Обробка зображень і мультимедіа

Олег Гутік

Лекція 8: Стиснення зображень, II

Стиснення зображень. Коди Грея

Методи стиснення зображень, розроблені для конкретних типів графічних образів, можуть успішно застосовуватися для компресії інших типів. Наприклад, будь-який метод стиснення дворівневих зображень підходить для напівтонових образів, якщо попередньо розшарувати напівтоновий образ на декілька дворівневих і кожен шар стискати незалежно. Уявімо собі зображення з 16 градаціями сірого кольору. Кожен піксель визначається 4 бітами, а тому зображення поділяється на 4 дворівневі. Однак за такого підходу може порушитися основний принцип стиснення зображень. Уявімо два прилеглих 4-бітних пікселя зі значеннями $7 = 0111_2$ і $8 = 1000_2$. Ці пікселі мають близькі значення, але при їхньому поділі на 4 шари, вони будуть різнитися у всіх 4 шарах! Відбувається це через те, що у двійковому поданні числа 7 і 8 різняться у всіх чотирьох розрядах. Для того, щоб успішно застосувати тут метод стиснення дворівневих зображень необхідно, щоб у двійковому поданні два послідовні числа мали двійкові коди, що відрізняються лише на один біт. Таке уявлення чисел існує і називається *рефлексними кодами Грея* (RGC, *reflected Gray code*). Їх легко побудувати, дотримуючись запропонованого рекурсивного правила.

Стиснення зображень. Коди Грея

Методи стиснення зображень, розроблені для конкретних типів графічних образів, можуть успішно застосовуватися для компресії інших типів. Наприклад, будь-який метод стиснення дворівневих зображень підходить для напівтонових образів, якщо попередньо розшарувати напівтоновий образ на декілька дворівневих і кожен шар стискати незалежно. Уявімо собі зображення з 16 градаціями сірого кольору. Кожен піксель визначається 4 бітами, а тому зображення поділяється на 4 дворівневі. Однак за такого підходу може порушитися основний принцип стиснення зображень. Уявімо два прилеглих 4-бітних пікселя зі значеннями $7 = 0111_2$ і $8 = 1000_2$. Ці пікселі мають близькі значення, але при їхньому поділі на 4 шари, вони будуть різнитися у всіх 4 шарах! Відбувається це через те, що у двійковому поданні числа 7 і 8 різняться у всіх чотирьох розрядах. Для того, щоб успішно застосувати тут метод стиснення дворівневих зображень необхідно, щоб у двійковому поданні два послідовні числа мали двійкові коди, що відрізняються лише на один біт. Таке уявлення чисел існує і називається *рефлексними кодами Грея* (RGC, *reflected Gray code*). Їх легко побудувати, дотримуючись запропонованого рекурсивного правила.

Стиснення зображень. Коди Грея

Методи стиснення зображень, розроблені для конкретних типів графічних образів, можуть успішно застосовуватися для компресії інших типів. Наприклад, будь-який метод стиснення дворівневих зображень підходить для напівтонових образів, якщо попередньо розшарувати напівтоновий образ на декілька дворівневих і кожен шар стискати незалежно. Уявімо собі зображення з 16 градаціями сірого кольору. Кожен піксель визначається 4 бітами, а тому зображення поділяється на 4 дворівневі. Однак за такого підходу може порушитися основний принцип стиснення зображень. Уявімо два прилеглих 4-бітних пікселя зі значеннями $7 = 0111_2$ і $8 = 1000_2$. Ці пікселі мають близькі значення, але при їхньому поділі на 4 шари, вони будуть різнитися у всіх 4 шарах! Відбувається це через те, що у двійковому поданні числа 7 і 8 різняться у всіх чотирьох розрядах. Для того, щоб успішно застосувати тут метод стиснення дворівневих зображень необхідно, щоб у двійковому поданні два послідовні числа мали двійкові коди, що відрізняються лише на один біт. Таке уявлення чисел існує і називається *рефлексними кодами Грея* (RGC, *reflected Gray code*). Їх легко побудувати, дотримуючись запропонованого рекурсивного правила.

Стиснення зображень. Коди Грея

Методи стиснення зображень, розроблені для конкретних типів графічних образів, можуть успішно застосовуватися для компресії інших типів. Наприклад, будь-який метод стиснення дворівневих зображень підходить для напівтонових образів, якщо попередньо розшарувати напівтоновий образ на декілька дворівневих і кожен шар стискати незалежно. Уявімо собі зображення з 16 градаціями сірого кольору. Кожен піксель визначається 4 бітами, а тому зображення поділяється на 4 дворівневі. Однак за такого підходу може порушитися основний принцип стиснення зображень. Уявімо два прилеглих 4-бітних пікселя зі значеннями $7 = 0111_2$ і $8 = 1000_2$. Ці пікселі мають близькі значення, але при їхньому поділі на 4 шари, вони будуть різнитися у всіх 4 шарах! Відбувається це через те, що у двійковому поданні числа 7 і 8 різняться у всіх чотирьох розрядах. Для того, щоб успішно застосувати тут метод стиснення дворівневих зображень необхідно, щоб у двійковому поданні два послідовні числа мали двійкові коди, що відрізняються лише на один біт. Таке уявлення чисел існує і називається *рефлексними кодами Грея* (RGC, *reflected Gray code*). Їх легко побудувати, дотримуючись запропонованого рекурсивного правила.

Стиснення зображень. Коди Грея

Методи стиснення зображень, розроблені для конкретних типів графічних образів, можуть успішно застосовуватися для компресії інших типів. Наприклад, будь-який метод стиснення дворівневих зображень підходить для напівтонових образів, якщо попередньо розшарувати напівтоновий образ на декілька дворівневих і кожен шар стискати незалежно. Уявімо собі зображення з 16 градаціями сірого кольору. Кожен піксель визначається 4 бітами, а тому зображення поділяється на 4 дворівневі. Однак за такого підходу може порушитися основний принцип стиснення зображень. Уявімо два прилеглих 4-бітних пікселя зі значеннями $7 = 0111_2$ і $8 = 1000_2$. Ці пікселі мають близькі значення, але при їхньому поділі на 4 шари, вони будуть різнитися у всіх 4 шарах! Відбувається це через те, що у двійковому поданні числа 7 і 8 різняться у всіх чотирьох розрядах. Для того, щоб успішно застосувати тут метод стиснення дворівневих зображень необхідно, щоб у двійковому поданні два послідовні числа мали двійкові коди, що відрізняються лише на один біт. Таке уявлення чисел існує і називається *рефлексними кодами Грея* (RGC, *reflected Gray code*). Їх легко побудувати, дотримуючись запропонованого рекурсивного правила.

Стиснення зображень. Коди Грея

Методи стиснення зображень, розроблені для конкретних типів графічних образів, можуть успішно застосовуватися для компресії інших типів. Наприклад, будь-який метод стиснення дворівневих зображень підходить для напівтонових образів, якщо попередньо розшарувати напівтоновий образ на декілька дворівневих і кожен шар стискати незалежно. Уявімо собі зображення з 16 градаціями сірого кольору. Кожен піксель визначається 4 бітами, а тому зображення поділяється на 4 дворівневі. Однак за такого підходу може порушитися основний принцип стиснення зображень. Уявімо два прилеглих 4-бітних пікселя зі значеннями $7 = 0111_2$ і $8 = 1000_2$. Ці пікселі мають близькі значення, але при їхньому поділі на 4 шари, вони будуть різнитися у всіх 4 шарах! Відбувається це через те, що у двійковому поданні числа 7 і 8 різняться у всіх чотирьох розрядах. Для того, щоб успішно застосувати тут метод стиснення дворівневих зображень необхідно, щоб у двійковому поданні два послідовні числа мали двійкові коди, що відрізняються лише на один біт. Таке уявлення чисел існує і називається *рефлексними кодами Грея* (RGC, *reflected Gray code*). Їх легко побудувати, дотримуючись запропонованого рекурсивного правила.

Стиснення зображень. Коди Грея

Методи стиснення зображень, розроблені для конкретних типів графічних образів, можуть успішно застосовуватися для компресії інших типів. Наприклад, будь-який метод стиснення дворівневих зображень підходить для напівтонових образів, якщо попередньо розшарувати напівтоновий образ на декілька дворівневих і кожен шар стискати незалежно. Уявімо собі зображення з 16 градаціями сірого кольору. Кожен піксель визначається 4 бітами, а тому зображення поділяється на 4 дворівневі. Однак за такого підходу може порушитися основний принцип стиснення зображень. Уявімо два прилеглих 4-бітних пікселя зі значеннями $7 = 0111_2$ і $8 = 1000_2$. Ці пікселі мають близькі значення, але при їхньому поділі на 4 шари, вони будуть різнитися у всіх 4 шарах! Відбувається це через те, що у двійковому поданні числа 7 і 8 різняться у всіх чотирьох розрядах. Для того, щоб успішно застосувати тут метод стиснення дворівневих зображень необхідно, щоб у двійковому поданні два послідовні числа мали двійкові коди, що відрізняються лише на один біт. Таке уявлення чисел існує і називається *рефлексними кодами Грея* (RGC, *reflected Gray code*). Їх легко побудувати, дотримуючись запропонованого рекурсивного правила.

Стиснення зображень. Коди Грея

Методи стиснення зображень, розроблені для конкретних типів графічних образів, можуть успішно застосовуватися для компресії інших типів. Наприклад, будь-який метод стиснення дворівневих зображень підходить для напівтонових образів, якщо попередньо розшарувати напівтоновий образ на декілька дворівневих і кожен шар стискати незалежно. Уявімо собі зображення з 16 градаціями сірого кольору. Кожен піксель визначається 4 бітами, а тому зображення поділяється на 4 дворівневі. Однак за такого підходу може порушитися основний принцип стиснення зображень. Уявімо два прилеглих 4-бітних пікселя зі значеннями $7 = 0111_2$ і $8 = 1000_2$. Ці пікселі мають близькі значення, але при їхньому поділі на 4 шари, вони будуть різнитися у всіх 4 шарах! Відбувається це через те, що у двійковому поданні числа 7 і 8 різняться у всіх чотирьох розрядах. Для того, щоб успішно застосувати тут метод стиснення дворівневих зображень необхідно, щоб у двійковому поданні два послідовні числа мали двійкові коди, що відрізняються лише на один біт. Таке уявлення чисел існує і називається *рефлексними кодами Грея* (RGC, *reflected Gray code*). Їх легко побудувати, дотримуючись запропонованого рекурсивного правила.

Стиснення зображень. Коди Грея

Методи стиснення зображень, розроблені для конкретних типів графічних образів, можуть успішно застосовуватися для компресії інших типів. Наприклад, будь-який метод стиснення дворівневих зображень підходить для напівтонових образів, якщо попередньо розшарувати напівтоновий образ на декілька дворівневих і кожен шар стискати незалежно. Уявімо собі зображення з 16 градаціями сірого кольору. Кожен піксель визначається 4 бітами, а тому зображення поділяється на 4 дворівневі. Однак за такого підходу може порушитися основний принцип стиснення зображень. Уявімо два прилеглих 4-бітних пікселя зі значеннями $7 = 0111_2$ і $8 = 1000_2$. Ці пікселі мають близькі значення, але при їхньому поділі на 4 шари, вони будуть різнитися у всіх 4 шарах! Відбувається це через те, що у двійковому поданні числа 7 і 8 різняться у всіх чотирьох розрядах. Для того, щоб успішно застосувати тут метод стиснення дворівневих зображень необхідно, щоб у двійковому поданні два послідовні числа мали двійкові коди, що відрізняються лише на один біт. Таке уявлення чисел існує і називається *рефлексними кодами Грея* (RGC, *reflected Gray code*). Їх легко побудувати, дотримуючись запропонованого рекурсивного правила.

Стиснення зображень. Коди Грея

Методи стиснення зображень, розроблені для конкретних типів графічних образів, можуть успішно застосовуватися для компресії інших типів. Наприклад, будь-який метод стиснення дворівневих зображень підходить для напівтонових образів, якщо попередньо розшарувати напівтоновий образ на декілька дворівневих і кожен шар стискати незалежно. Уявімо собі зображення з 16 градаціями сірого кольору. Кожен піксель визначається 4 бітами, а тому зображення поділяється на 4 дворівневі. Однак за такого підходу може порушитися основний принцип стиснення зображень. Уявімо два прилеглих 4-бітних пікселя зі значеннями $7 = 0111_2$ і $8 = 1000_2$. Ці пікселі мають близькі значення, але при їхньому поділі на 4 шари, вони будуть різнитися у всіх 4 шарах! Відбувається це через те, що у двійковому поданні числа 7 і 8 різняться у всіх чотирьох розрядах. Для того, щоб успішно застосувати тут метод стиснення дворівневих зображень необхідно, щоб у двійковому поданні два послідовні числа мали двійкові коди, що відрізняються лише на один біт. Таке уявлення чисел існує і називається *рефлексними кодами Грея* (RGC, *reflected Gray code*). Їх легко побудувати, дотримуючись запропонованого рекурсивного правила.

Стиснення зображень. Коди Грея

Методи стиснення зображень, розроблені для конкретних типів графічних образів, можуть успішно застосовуватися для компресії інших типів. Наприклад, будь-який метод стиснення дворівневих зображень підходить для напівтонових образів, якщо попередньо розшарувати напівтоновий образ на декілька дворівневих і кожен шар стискати незалежно. Уявімо собі зображення з 16 градаціями сірого кольору. Кожен піксель визначається 4 бітами, а тому зображення поділяється на 4 дворівневі. Однак за такого підходу може порушитися основний принцип стиснення зображень. Уявімо два прилеглих 4-бітних пікселя зі значеннями $7 = 0111_2$ і $8 = 1000_2$. Ці пікселі мають близькі значення, але при їхньому поділі на 4 шари, вони будуть різнитися у всіх 4 шарах! Відбувається це через те, що у двійковому поданні числа 7 і 8 різняться у всіх чотирьох розрядах. Для того, щоб успішно застосувати тут метод стиснення дворівневих зображень необхідно, щоб у двійковому поданні два послідовні числа мали двійкові коди, що відрізняються лише на один біт. Таке уявлення чисел існує і називається *рефлексними кодами Грея* (RGC, *reflected Gray code*). Їх легко побудувати, дотримуючись запропонованого рекурсивного правила.

Стиснення зображень. Коди Грея

Методи стиснення зображень, розроблені для конкретних типів графічних образів, можуть успішно застосовуватися для компресії інших типів. Наприклад, будь-який метод стиснення дворівневих зображень підходить для напівтонових образів, якщо попередньо розшарувати напівтоновий образ на декілька дворівневих і кожен шар стискати незалежно. Уявімо собі зображення з 16 градаціями сірого кольору. Кожен піксель визначається 4 бітами, а тому зображення поділяється на 4 дворівневі. Однак за такого підходу може порушитися основний принцип стиснення зображень. Уявімо два прилеглих 4-бітних пікселя зі значеннями $7 = 0111_2$ і $8 = 1000_2$. Ці пікселі мають близькі значення, але при їхньому поділі на 4 шари, вони будуть різнитися у всіх 4 шарах! Відбувається це через те, що у двійковому поданні числа 7 і 8 різняться у всіх чотирьох розрядах. Для того, щоб успішно застосувати тут метод стиснення дворівневих зображень необхідно, щоб у двійковому поданні два послідовні числа мали двійкові коди, що відрізняються лише на один біт. Таке уявлення чисел існує і називається *рефлексними кодами Грея* (RGC, *reflected Gray code*). Їх легко побудувати, дотримуючись запропонованого рекурсивного правила.

Стиснення зображень. Коди Грея

Методи стиснення зображень, розроблені для конкретних типів графічних образів, можуть успішно застосовуватися для компресії інших типів. Наприклад, будь-який метод стиснення дворівневих зображень підходить для напівтонових образів, якщо попередньо розшарувати напівтоновий образ на декілька дворівневих і кожен шар стискати незалежно. Уявімо собі зображення з 16 градаціями сірого кольору. Кожен піксель визначається 4 бітами, а тому зображення поділяється на 4 дворівневі. Однак за такого підходу може порушитися основний принцип стиснення зображень. Уявімо два прилеглих 4-бітних пікселя зі значеннями $7 = 0111_2$ і $8 = 1000_2$. Ці пікселі мають близькі значення, але при їхньому поділі на 4 шари, вони будуть різнитися у всіх 4 шарах! Відбувається це через те, що у двійковому поданні числа 7 і 8 різняться у всіх чотирьох розрядах. Для того, щоб успішно застосувати тут метод стиснення дворівневих зображень необхідно, щоб у двійковому поданні два послідовні числа мали двійкові коди, що відрізняються лише на один біт. Таке уявлення чисел існує і називається *рефлексними кодами Грея* (RGC, *reflected Gray code*). Їх легко побудувати, дотримуючись запропонованого рекурсивного правила.

Стиснення зображень. Коди Грея

Методи стиснення зображень, розроблені для конкретних типів графічних образів, можуть успішно застосовуватися для компресії інших типів. Наприклад, будь-який метод стиснення дворівневих зображень підходить для напівтонових образів, якщо попередньо розшарувати напівтоновий образ на декілька дворівневих і кожен шар стискати незалежно. Уявімо собі зображення з 16 градаціями сірого кольору. Кожен піксель визначається 4 бітами, а тому зображення поділяється на 4 дворівневі. Однак за такого підходу може порушитися основний принцип стиснення зображень. Уявімо два прилеглих 4-бітних пікселя зі значеннями $7 = 0111_2$ і $8 = 1000_2$. Ці пікселі мають близькі значення, але при їхньому поділі на 4 шари, вони будуть різнитися у всіх 4 шарах! Відбувається це через те, що у двійковому поданні числа 7 і 8 різняться у всіх чотирьох розрядах. Для того, щоб успішно застосувати тут метод стиснення дворівневих зображень необхідно, щоб у двійковому поданні два послідовні числа мали двійкові коди, що відрізняються лише на один біт. Таке уявлення чисел існує і називається *рефлексними кодами Грея* (RGC, *reflected Gray code*). Їх легко побудувати, дотримуючись запропонованого рекурсивного правила.

Стиснення зображень. Коди Грея

Методи стиснення зображень, розроблені для конкретних типів графічних образів, можуть успішно застосовуватися для компресії інших типів. Наприклад, будь-який метод стиснення дворівневих зображень підходить для напівтонових образів, якщо попередньо розшарувати напівтоновий образ на декілька дворівневих і кожен шар стискати незалежно. Уявімо собі зображення з 16 градаціями сірого кольору. Кожен піксель визначається 4 бітами, а тому зображення поділяється на 4 дворівневі. Однак за такого підходу може порушитися основний принцип стиснення зображень. Уявімо два прилеглих 4-бітних пікселя зі значеннями $7 = 0111_2$ і $8 = 1000_2$. Ці пікселі мають близькі значення, але при їхньому поділі на 4 шари, вони будуть різнитися у всіх 4 шарах! Відбувається це через те, що у двійковому поданні числа 7 і 8 різняться у всіх чотирьох розрядах. Для того, щоб успішно застосувати тут метод стиснення дворівневих зображень необхідно, щоб у двійковому поданні два послідовні числа мали двійкові коди, що відрізняються лише на один біт. Таке уявлення чисел існує і називається *рефлексними кодами Грея* (RGC, *reflected Gray code*). Їх легко побудувати, дотримуючись запропонованого рекурсивного правила.

Стиснення зображень. Коди Грея

Методи стиснення зображень, розроблені для конкретних типів графічних образів, можуть успішно застосовуватися для компресії інших типів. Наприклад, будь-який метод стиснення дворівневих зображень підходить для напівтонових образів, якщо попередньо розшарувати напівтоновий образ на декілька дворівневих і кожен шар стискати незалежно. Уявімо собі зображення з 16 градаціями сірого кольору. Кожен піксель визначається 4 бітами, а тому зображення поділяється на 4 дворівневі. Однак за такого підходу може порушитися основний принцип стиснення зображень. Уявімо два прилеглих 4-бітних пікселя зі значеннями $7 = 0111_2$ і $8 = 1000_2$. Ці пікселі мають близькі значення, але при їхньому поділі на 4 шари, вони будуть різнитися у всіх 4 шарах! Відбувається це через те, що у двійковому поданні числа 7 і 8 різняться у всіх чотирьох розрядах. Для того, щоб успішно застосувати тут метод стиснення дворівневих зображень необхідно, щоб у двійковому поданні два послідовні числа мали двійкові коди, що відрізняються лише на один біт. Таке уявлення чисел існує і називається *рефлексними кодами Грея* (RGC, *reflected Gray code*). Їх легко побудувати, дотримуючись запропонованого рекурсивного правила.

Стиснення зображень. Коди Грея

Методи стиснення зображень, розроблені для конкретних типів графічних образів, можуть успішно застосовуватися для компресії інших типів. Наприклад, будь-який метод стиснення дворівневих зображень підходить для напівтонових образів, якщо попередньо розшарувати напівтоновий образ на декілька дворівневих і кожен шар стискати незалежно. Уявімо собі зображення з 16 градаціями сірого кольору. Кожен піксель визначається 4 бітами, а тому зображення поділяється на 4 дворівневі. Однак за такого підходу може порушитися основний принцип стиснення зображень. Уявімо два прилеглих 4-бітних пікселя зі значеннями $7 = 0111_2$ і $8 = 1000_2$. Ці пікселі мають близькі значення, але при їхньому поділі на 4 шари, вони будуть різнитися у всіх 4 шарах! Відбувається це через те, що у двійковому поданні числа 7 і 8 різняться у всіх чотирьох розрядах. Для того, щоб успішно застосувати тут метод стиснення дворівневих зображень необхідно, щоб у двійковому поданні два послідовні числа мали двійкові коди, що відрізняються лише на один біт. Таке уявлення чисел існує і називається *рефлексними кодами Грея* (RGC, *reflected Gray code*). Їх легко побудувати, дотримуючись запропонованого рекурсивного правила.

Стиснення зображень. Коди Грея

Методи стиснення зображень, розроблені для конкретних типів графічних образів, можуть успішно застосовуватися для компресії інших типів. Наприклад, будь-який метод стиснення дворівневих зображень підходить для напівтонових образів, якщо попередньо розшарувати напівтоновий образ на декілька дворівневих і кожен шар стискати незалежно. Уявімо собі зображення з 16 градаціями сірого кольору. Кожен піксель визначається 4 бітами, а тому зображення поділяється на 4 дворівневі. Однак за такого підходу може порушитися основний принцип стиснення зображень. Уявімо два прилеглих 4-бітних пікселя зі значеннями $7 = 0111_2$ і $8 = 1000_2$. Ці пікселі мають близькі значення, але при їхньому поділі на 4 шари, вони будуть різнитися у всіх 4 шарах! Відбувається це через те, що у двійковому поданні числа 7 і 8 різняться у всіх чотирьох розрядах. Для того, щоб успішно застосувати тут метод стиснення дворівневих зображень необхідно, щоб у двійковому поданні два послідовні числа мали двійкові коди, що відрізняються лише на один біт. Таке уявлення чисел існує і називається *рефлексними кодами Грея* (RGC, *reflected Gray code*). Їх легко побудувати, дотримуючись запропонованого рекурсивного правила.

Стиснення зображень. Коди Грея

Почнемо із двох однобітних кодів $(0, 1)$. Побудуємо дві сім'ї двобітних кодів, повторивши $(0, 1)$, а потім додавши зліва (або праворуч), спочатку 0, а потім 1 до вихідних сім'ї. В результаті отримаємо $(00, 01)$ та $(10, 11)$. Тепер слід переставити (відобразити симетрично) коди другої сім'ї у зворотному порядку та приписати до першої сім'ї. Тоді вийдуть 4 двобітні коди RGC $(00, 01, 11, 10)$, за допомогою яких можна кодувати цілі числа від 0 до 3, причому послідовні числа будуть відрізнятися рівно в одному двійковому розряді. Застосовуємо цю процедуру ще раз і отримуємо дві сім'ї кодів довжини 3: $(000, 001, 011, 010)$ та $(110, 111, 101, 100)$. Об'єднуємо їх та отримуємо 3-бітові коди RGC. Виконаємо перші три кроки для утворення 4-бітних кодів Грея:

$$\begin{aligned}
\text{додати } 0: &\quad (0000, 0001, 0011, 0010, 0110, 0111, 0101, 0100),\\
\text{додати } 1: &\quad (1000, 1001, 1011, 1010, 1110, 1111, 1101, 1100),\\
\text{відобразити: } &\quad (1100, 1101, 1111, 1110, 1010, 1011, 1001, 1000).
\end{aligned}$$

Стиснення зображень. Коди Грея

Почнемо із двох однобітних кодів $(0, 1)$. Побудуємо дві сім'ї двобітних кодів, повторивши $(0, 1)$, а потім додавши зліва (або праворуч), спочатку 0, а потім 1 до вихідних сім'ї. В результаті отримаємо $(00, 01)$ та $(10, 11)$. Тепер слід переставити (відобразити симетрично) коди другої сім'ї у зворотному порядку та приписати до першої сім'ї. Тоді вийдуть 4 двобітні коди RGC $(00, 01, 11, 10)$, за допомогою яких можна кодувати цілі числа від 0 до 3, причому послідовні числа будуть відрізнятися рівно в одному двійковому розряді. Застосовуємо цю процедуру ще раз і отримуємо дві сім'ї кодів довжини 3: $(000, 001, 011, 010)$ та $(110, 111, 101, 100)$. Об'єднуємо їх та отримуємо 3-бітові коди RGC. Виконаємо перші три кроки для утворення 4-бітних кодів Грея:

$$\begin{aligned}
\text{додати } 0: &\quad (0000, 0001, 0011, 0010, 0110, 0111, 0101, 0100),\\
\text{додати } 1: &\quad (1000, 1001, 1011, 1010, 1110, 1111, 1101, 1100),\\
\text{відобразити: } &\quad (1100, 1101, 1111, 1110, 1010, 1011, 1001, 1000).
\end{aligned}$$

Стиснення зображень. Коди Грея

Почнемо із двох однобітних кодів $(0, 1)$. Побудуємо дві сім'ї двобітних кодів, повторивши $(0, 1)$, а потім додавши зліва (або праворуч), спочатку 0, а потім 1 до вихідних сім'ї. В результаті отримаємо $(00, 01)$ та $(10, 11)$. Тепер слід переставити (відобразити симетрично) коди другої сім'ї у зворотному порядку та приписати до першої сім'ї. Тоді вийдуть 4 двобітні коди RGC $(00, 01, 11, 10)$, за допомогою яких можна кодувати цілі числа від 0 до 3, причому послідовні числа будуть відрізнятися рівно в одному двійковому розряді. Застосовуємо цю процедуру ще раз і отримуємо дві сім'ї кодів довжини 3: $(000, 001, 011, 010)$ та $(110, 111, 101, 100)$. Об'єднуємо їх та отримуємо 3-бітові коди RGC. Виконаємо перші три кроки для утворення 4-бітних кодів Грея:

$$\begin{aligned}
\text{додати } 0: &\quad (0000, 0001, 0011, 0010, 0110, 0111, 0101, 0100),\\
\text{додати } 1: &\quad (1000, 1001, 1011, 1010, 1110, 1111, 1101, 1100),\\
\text{відобразити: } &\quad (1100, 1101, 1111, 1110, 1010, 1011, 1001, 1000).
\end{aligned}$$

Стиснення зображень. Коди Грея

Почнемо із двох однобітних кодів $(0, 1)$. Побудуємо дві сім'ї двобітних кодів, повторивши $(0, 1)$, а потім додавши зліва (або праворуч), спочатку 0, а потім 1 до вихідних сім'ї. В результаті отримаємо $(00, 01)$ та $(10, 11)$. Тепер слід переставити (відобразити симетрично) коди другої сім'ї у зворотному порядку та приписати до першої сім'ї. Тоді вийдуть 4 двобітні коди RGC $(00, 01, 11, 10)$, за допомогою яких можна кодувати цілі числа від 0 до 3, причому послідовні числа будуть відрізнятися рівно в одному двійковому розряді. Застосовуємо цю процедуру ще раз і отримуємо дві сім'ї кодів довжини 3: $(000, 001, 011, 010)$ та $(110, 111, 101, 100)$. Об'єднуємо їх та отримуємо 3-бітові коди RGC. Виконаємо перші три кроки для утворення 4-бітних кодів Грея:

$$\begin{aligned} \text{додати } 0: &\quad (0000, 0001, 0011, 0010, 0110, 0111, 0101, 0100), \\ \text{додати } 1: &\quad (1000, 1001, 1011, 1010, 1110, 1111, 1101, 1100), \\ \text{відобразити: } &\quad (1100, 1101, 1111, 1110, 1010, 1011, 1001, 1000). \end{aligned}$$

Стиснення зображень. Коди Грея

Почнемо із двох однобітних кодів $(0, 1)$. Побудуємо дві сім'ї двобітних кодів, повторивши $(0, 1)$, а потім додавши зліва (або праворуч), спочатку 0, а потім 1 до вихідних сім'ї. В результаті отримаємо $(00, 01)$ та $(10, 11)$. Тепер слід переставити (відобразити симетрично) коди другої сім'ї у зворотному порядку та приписати до першої сім'ї. Тоді вийдуть 4 двобітні коди RGC $(00, 01, 11, 10)$, за допомогою яких можна кодувати цілі числа від 0 до 3, причому послідовні числа будуть відрізнятися рівно в одному двійковому розряді. Застосовуємо цю процедуру ще раз і отримуємо дві сім'ї кодів довжини 3: $(000, 001, 011, 010)$ та $(110, 111, 101, 100)$. Об'єднуємо їх та отримуємо 3-бітові коди RGC. Виконаємо перші три кроки для утворення 4-бітних кодів Грея:

$$\begin{aligned} \text{додати } 0: &\quad (0000, 0001, 0011, 0010, 0110, 0111, 0101, 0100), \\ \text{додати } 1: &\quad (1000, 1001, 1011, 1010, 1110, 1111, 1101, 1100), \\ \text{відобразити: } &\quad (1100, 1101, 1111, 1110, 1010, 1011, 1001, 1000). \end{aligned}$$

Стиснення зображень. Коди Грея

Почнемо із двох однобітних кодів $(0, 1)$. Побудуємо дві сім'ї двобітних кодів, повторивши $(0, 1)$, а потім додавши зліва (або праворуч), спочатку 0, а потім 1 до вихідних сім'ї. В результаті отримаємо $(00, 01)$ та $(10, 11)$. Тепер слід переставити (відобразити симетрично) коди другої сім'ї у зворотному порядку та приписати до першої сім'ї. Тоді вийдуть 4 двобітні коди RGC $(00, 01, 11, 10)$, за допомогою яких можна кодувати цілі числа від 0 до 3, причому послідовні числа будуть відрізнятися рівно в одному двійковому розряді. Застосовуємо цю процедуру ще раз і отримуємо дві сім'ї кодів довжини 3: $(000, 001, 011, 010)$ та $(110, 111, 101, 100)$. Об'єднуємо їх та отримуємо 3-бітові коди RGC. Виконаємо перші три кроки для утворення 4-бітних кодів Грея:

$$\begin{aligned} \text{додати } 0: &\quad (0000, 0001, 0011, 0010, 0110, 0111, 0101, 0100), \\ \text{додати } 1: &\quad (1000, 1001, 1011, 1010, 1110, 1111, 1101, 1100), \\ \text{відобразити: } &\quad (1100, 1101, 1111, 1110, 1010, 1011, 1001, 1000). \end{aligned}$$

Стиснення зображень. Коди Грея

Почнемо із двох однобітних кодів $(0, 1)$. Побудуємо дві сім'ї двобітних кодів, повторивши $(0, 1)$, а потім додавши зліва (або праворуч), спочатку 0, а потім 1 до вихідних сім'ї. В результаті отримаємо $(00, 01)$ та $(10, 11)$. Тепер слід переставити (відобразити симетрично) коди другої сім'ї у зворотному порядку та приписати до першої сім'ї. Тоді вийдуть 4 двобітні коди RGC $(00, 01, 11, 10)$, за допомогою яких можна кодувати цілі числа від 0 до 3, причому послідовні числа будуть відрізнятися рівно в одному двійковому розряді. Застосовуємо цю процедуру ще раз і отримуємо дві сім'ї кодів довжини 3: $(000, 001, 011, 010)$ та $(110, 111, 101, 100)$. Об'єднуємо їх та отримуємо 3-бітові коди RGC. Виконаємо перші три кроки для утворення 4-бітних кодів Грея:

$$\begin{aligned}
\text{додати } 0: &\quad (0000, 0001, 0011, 0010, 0110, 0111, 0101, 0100),\\
\text{додати } 1: &\quad (1000, 1001, 1011, 1010, 1110, 1111, 1101, 1100),\\
\text{відобразити: } &\quad (1100, 1101, 1111, 1110, 1010, 1011, 1001, 1000).
\end{aligned}$$

Стиснення зображень. Коди Грея

Почнемо із двох однобітних кодів $(0, 1)$. Побудуємо дві сім'ї двобітних кодів, повторивши $(0, 1)$, а потім додавши зліва (або праворуч), спочатку 0, а потім 1 до вихідних сім'ї. В результаті отримаємо $(00, 01)$ та $(10, 11)$. Тепер слід переставити (відобразити симетрично) коди другої сім'ї у зворотному порядку та приписати до першої сім'ї. Тоді вийдуть 4 двобітні коди RGC $(00, 01, 11, 10)$, за допомогою яких можна кодувати цілі числа від 0 до 3, причому послідовні числа будуть відрізнятися рівно в одному двійковому розряді. Застосовуємо цю процедуру ще раз і отримуємо дві сім'ї кодів довжини 3: $(000, 001, 011, 010)$ та $(110, 111, 101, 100)$. Об'єднуємо їх та отримуємо 3-бітові коди RGC. Виконаємо перші три кроки для утворення 4-бітних кодів Грея:

$$\begin{aligned}
\text{додати } 0: &\quad (0000, 0001, 0011, 0010, 0110, 0111, 0101, 0100),\\
\text{додати } 1: &\quad (1000, 1001, 1011, 1010, 1110, 1111, 1101, 1100),\\
\text{відобразити: } &\quad (1100, 1101, 1111, 1110, 1010, 1011, 1001, 1000).
\end{aligned}$$

Стиснення зображень. Коди Грея

Почнемо із двох однобітних кодів $(0, 1)$. Побудуємо дві сім'ї двобітних кодів, повторивши $(0, 1)$, а потім додавши зліва (або праворуч), спочатку 0, а потім 1 до вихідних сім'ї. В результаті отримаємо $(00, 01)$ та $(10, 11)$. Тепер слід переставити (відобразити симетрично) коди другої сім'ї у зворотному порядку та приписати до першої сім'ї. Тоді вийдуть 4 двобітні коди RGC $(00, 01, 11, 10)$, за допомогою яких можна кодувати цілі числа від 0 до 3, причому послідовні числа будуть відрізнятися рівно в одному двійковому розряді. Застосовуємо цю процедуру ще раз і отримуємо дві сім'ї кодів довжини 3: $(000, 001, 011, 010)$ та $(110, 111, 101, 100)$. Об'єднуємо їх та отримуємо 3-бітові коди RGC. Виконаємо перші три кроки для утворення 4-бітних кодів Грея:

$$\begin{aligned} \text{додати } 0: &\quad (0000, 0001, 0011, 0010, 0110, 0111, 0101, 0100), \\ \text{додати } 1: &\quad (1000, 1001, 1011, 1010, 1110, 1111, 1101, 1100), \\ \text{відобразити: } &\quad (1100, 1101, 1111, 1110, 1010, 1011, 1001, 1000). \end{aligned}$$

Стиснення зображень. Коди Грея

Почнемо із двох однобітних кодів $(0, 1)$. Побудуємо дві сім'ї двобітних кодів, повторивши $(0, 1)$, а потім додавши зліва (або праворуч), спочатку 0, а потім 1 до вихідних сім'ї. В результаті отримаємо $(00, 01)$ та $(10, 11)$. Тепер слід переставити (відобразити симетрично) коди другої сім'ї у зворотному порядку та приписати до першої сім'ї. Тоді вийдуть 4 двобітні коди RGC $(00, 01, 11, 10)$, за допомогою яких можна кодувати цілі числа від 0 до 3, причому послідовні числа будуть відрізнятися рівно в одному двійковому розряді. Застосовуємо цю процедуру ще раз і отримуємо дві сім'ї кодів довжини 3: $(000, 001, 011, 010)$ та $(110, 111, 101, 100)$. Об'єднуємо їх та отримуємо 3-бітові коди RGC. Виконаємо перші три кроки для утворення 4-бітних кодів Грея:

$$\begin{aligned} \text{додати } 0: &\quad (0000, 0001, 0011, 0010, 0110, 0111, 0101, 0100), \\ \text{додати } 1: &\quad (1000, 1001, 1011, 1010, 1110, 1111, 1101, 1100), \\ \text{відобразити: } &\quad (1100, 1101, 1111, 1110, 1010, 1011, 1001, 1000). \end{aligned}$$

Стиснення зображень. Коди Грея

Почнемо із двох однобітних кодів $(0, 1)$. Побудуємо дві сім'ї двобітних кодів, повторивши $(0, 1)$, а потім додавши зліва (або праворуч), спочатку 0, а потім 1 до вихідних сім'ї. В результаті отримаємо $(00, 01)$ та $(10, 11)$. Тепер слід переставити (відобразити симетрично) коди другої сім'ї у зворотному порядку та приписати до першої сім'ї. Тоді вийдуть 4 двобітні коди RGC $(00, 01, 11, 10)$, за допомогою яких можна кодувати цілі числа від 0 до 3, причому послідовні числа будуть відрізнятися рівно в одному двійковому розряді. Застосовуємо цю процедуру ще раз і отримуємо дві сім'ї кодів довжини 3: $(000, 001, 011, 010)$ та $(110, 111, 101, 100)$. Об'єднуємо їх та отримуємо 3-бітові коди RGC. Виконаємо перші три кроки для утворення 4-бітних кодів Грея:

$$\begin{aligned}
\text{додати } 0: &\quad (0000, 0001, 0011, 0010, 0110, 0111, 0101, 0100),\\
\text{додати } 1: &\quad (1000, 1001, 1011, 1010, 1110, 1111, 1101, 1100),\\
\text{відобразити: } &\quad (1100, 1101, 1111, 1110, 1010, 1011, 1001, 1000).
\end{aligned}$$

Стиснення зображень. Коди Грея

Почнемо із двох однобітних кодів $(0, 1)$. Побудуємо дві сім'ї двобітних кодів, повторивши $(0, 1)$, а потім додавши зліва (або праворуч), спочатку 0, а потім 1 до вихідних сім'ї. В результаті отримаємо $(00, 01)$ та $(10, 11)$. Тепер слід переставити (відобразити симетрично) коди другої сім'ї у зворотному порядку та приписати до першої сім'ї. Тоді вийдуть 4 двобітні коди RGC $(00, 01, 11, 10)$, за допомогою яких можна кодувати цілі числа від 0 до 3, причому послідовні числа будуть відрізнятися рівно в одному двійковому розряді. Застосовуємо цю процедуру ще раз і отримуємо дві сім'ї кодів довжини 3: $(000, 001, 011, 010)$ та $(110, 111, 101, 100)$. Об'єднуємо їх та отримуємо 3-бітові коди RGC. Виконаємо перші три кроки для утворення 4-бітних кодів Грея:

$$\begin{aligned} \text{додати } 0: &\quad (0000, 0001, 0011, 0010, 0110, 0111, 0101, 0100), \\ \text{додати } 1: &\quad (1000, 1001, 1011, 1010, 1110, 1111, 1101, 1100), \\ \text{відобразити: } &\quad (1100, 1101, 1111, 1110, 1010, 1011, 1001, 1000). \end{aligned}$$

Стиснення зображень. Коди Грея

Почнемо із двох однобітних кодів $(0, 1)$. Побудуємо дві сім'ї двобітних кодів, повторивши $(0, 1)$, а потім додавши зліва (або праворуч), спочатку 0, а потім 1 до вихідних сім'ї. В результаті отримаємо $(00, 01)$ та $(10, 11)$. Тепер слід переставити (відобразити симетрично) коди другої сім'ї у зворотному порядку та приписати до першої сім'ї. Тоді вийдуть 4 двобітні коди RGC $(00, 01, 11, 10)$, за допомогою яких можна кодувати цілі числа від 0 до 3, причому послідовні числа будуть відрізнятися рівно в одному двійковому розряді. Застосовуємо цю процедуру ще раз і отримуємо дві сім'ї кодів довжини 3: $(000, 001, 011, 010)$ та $(110, 111, 101, 100)$. Об'єднуємо їх та отримуємо 3-бітові коди RGC. Виконаємо перші три кроки для утворення 4-бітних кодів Грея:

$$\begin{aligned} \text{додати } 0: &\quad (0000, 0001, 0011, 0010, 0110, 0111, 0101, 0100), \\ \text{додати } 1: &\quad (1000, 1001, 1011, 1010, 1110, 1111, 1101, 1100), \\ \text{відобразити: } &\quad (1100, 1101, 1111, 1110, 1010, 1011, 1001, 1000). \end{aligned}$$

Стиснення зображень. Коди Грея

Почнемо із двох однобітних кодів $(0, 1)$. Побудуємо дві сім'ї двобітних кодів, повторивши $(0, 1)$, а потім додавши зліва (або праворуч), спочатку 0, а потім 1 до вихідних сім'ї. В результаті отримаємо $(00, 01)$ та $(10, 11)$. Тепер слід переставити (відобразити симетрично) коди другої сім'ї у зворотному порядку та приписати до першої сім'ї. Тоді вийдуть 4 двобітні коди RGC $(00, 01, 11, 10)$, за допомогою яких можна кодувати цілі числа від 0 до 3, причому послідовні числа будуть відрізнятися рівно в одному двійковому розряді. Застосовуємо цю процедуру ще раз і отримуємо дві сім'ї кодів довжини 3: $(000, 001, 011, 010)$ та $(110, 111, 101, 100)$. Об'єднуємо їх та отримуємо 3-бітові коди RGC. Виконаємо перші три кроки для утворення 4-бітних кодів Грея:

$$\begin{aligned}
\text{додати } 0: &\quad (0000, 0001, 0011, 0010, 0110, 0111, 0101, 0100),\\
\text{додати } 1: &\quad (1000, 1001, 1011, 1010, 1110, 1111, 1101, 1100),\\
\text{відобразити: } &\quad (1100, 1101, 1111, 1110, 1010, 1011, 1001, 1000).
\end{aligned}$$

Почнемо із двох однобітних кодів $(0, 1)$. Побудуємо дві сім'ї двобітних кодів, повторивши $(0, 1)$, а потім додавши зліва (або праворуч), спочатку 0, а потім 1 до вихідних сім'ї. В результаті отримаємо $(00, 01)$ та $(10, 11)$. Тепер слід переставити (відобразити симетрично) коди другої сім'ї у зворотному порядку та приписати до першої сім'ї. Тоді вийдуть 4 двобітні коди RGC $(00, 01, 11, 10)$, за допомогою яких можна кодувати цілі числа від 0 до 3, причому послідовні числа будуть відрізнятися рівно в одному двійковому розряді. Застосовуємо цю процедуру ще раз і отримуємо дві сім'ї кодів довжини 3: $(000, 001, 011, 010)$ та $(110, 111, 101, 100)$. Об'єднуємо їх та отримуємо 3-бітові коди RGC. Виконаємо перші три кроки для утворення 4-бітних кодів Грея:

$$\begin{aligned}
\text{додати } 0: &\quad (0000, 0001, 0011, 0010, 0110, 0111, 0101, 0100),\\
\text{додати } 1: &\quad (1000, 1001, 1011, 1010, 1110, 1111, 1101, 1100),\\
\text{відобразити: } &\quad (1100, 1101, 1111, 1110, 1010, 1011, 1001, 1000).
\end{aligned}$$

Стиснення зображень. Коди Грея

43210	Gray	43210	Gray	43210	Gray	43210	Gray
00000	00000	**01000**	01100	**10000**	11000	**11000**	10100
0000**1**	0000**1**	0100**1**	0110**1**	1000**1**	1100**1**	1100**1**	1010**1**
000**10**	0001**1**	010**10**	0111**1**	100**10**	1101**1**	110**10**	1011**1**
0001**1**	0001**0**	0101**1**	0111**0**	1001**1**	1101**0**	1101**1**	1011**0**
00**100**	0011**0**	01**100**	0101**0**	10**100**	1111**0**	11**100**	1001**0**
0010**1**	0011**1**	0110**1**	0101**1**	1010**1**	1111**1**	1110**1**	1001**1**
001**10**	0010**1**	011**10**	0100**1**	101**10**	1110**1**	111**10**	1000**1**
0011**1**	0010**0**	0111**1**	0100**0**	1011**1**	1110**0**	1111**1**	1000**0**

```
function b=rgc(a,i)
[r,c]=size(a);
b= [zeros(r,1),a; ones(r,1),flipud(a)];
if i>1, b=rgc(b,i-1); end;
```

У табл. зображено, як змінюються окремі біти двійкових кодів, що становлять числа від 0 до 32. У цій таблиці в непарних стовпцях наведено звичайні двійкові коди цілих чисел. Жирним шрифтом виділено біти числа i, які відрізняються від відповідних бітів числа $i-1$. Видно, що біт наймолодшого (нульового) розряду (біт b_0) змінюється щоразу, біт b_1 змінюється кожен другий раз, а в загальному випадку, біт b_k змінюється у кожного 2^k-го цілого числа. У парних стовпцях таблиці розміщено всі послідовні коди Грея. Під таблицею наведено рекурсивну функцію Matlab, яка обчислює коди Грея.

Стиснення зображень. Коди Грея

43210	Gray	43210	Gray	43210	Gray	43210	Gray
00000	00000	**01000**	01100	**10000**	11000	**11000**	10100
0000**1**	00001	0100**1**	01101	1000**1**	11001	1100**1**	10101
000**10**	00011	010**10**	01111	100**10**	11011	110**10**	10111
0001**1**	00010	0101**1**	01110	1001**1**	11010	1101**1**	10110
00**100**	00110	01**100**	01010	10**100**	11110	11**100**	10010
0010**1**	00111	0110**1**	01011	1010**1**	11111	1110**1**	10011
0011**0**	00101	0111**0**	01001	1011**0**	11101	1111**0**	10001
0011**1**	00100	0111**1**	01000	1011**1**	11100	1111**1**	10000

```
function b=rgc(a,i)
[r,c]=size(a);
b= [zeros(r,1),a; ones(r,1),flipud(a)];
if i>1, b=rgc(b,i-1); end;
```

У табл. зображено, як змінюються окремі біти двійкових кодів, що становлять числа від 0 до 32. У цій таблиці в непарних стовпцях наведено звичайні двійкові коди цілих чисел. Жирним шрифтом виділено біти числа i, які відрізняються від відповідних бітів числа $i-1$. Видно, що біт наймолодшого (нульового) розряду (біт b_0) змінюється щоразу, біт b_1 змінюється кожен другий раз, а в загальному випадку, біт b_k змінюється у кожного 2^k-го цілого числа. У парних стовпцях таблиці розміщено всі послідовні коди Грея. Під таблицею наведено рекурсивну функцію Matlab, яка обчислює коди Грея.

Стиснення зображень. Коди Грея

43210	Gray	43210	Gray	43210	Gray	43210	Gray
00000	00000	**01000**	01100	**10000**	11000	**11000**	10100
0000**1**	0000**1**	0100**1**	0110**1**	1000**1**	1100**1**	1100**1**	1010**1**
000**10**	0001**1**	010**10**	0111**1**	100**10**	1101**1**	110**10**	1011**1**
0001**1**	0001**0**	0101**1**	0111**0**	1001**1**	1101**0**	1101**1**	1011**0**
00**100**	0011**0**	01**100**	0101**0**	10**100**	1111**0**	11**100**	1001**0**
0010**1**	0011**1**	0110**1**	0101**1**	1010**1**	1111**1**	1110**1**	1001**1**
001**10**	0010**1**	011**10**	0100**1**	101**10**	1110**1**	111**10**	1000**1**
0011**1**	0010**0**	0111**1**	0100**0**	1011**1**	1110**0**	1111**1**	1000**0**

```
function b=rgc(a,i)
[r,c]=size(a);
b= [zeros(r,1),a; ones(r,1),flipud(a)];
if i>1, b=rgc(b,i-1); end;
```

У табл. зображено, як змінюються окремі біти двійкових кодів, що становлять числа від 0 до 32. У цій таблиці в непарних стовпцях наведено звичайні двійкові коди цілих чисел. Жирним шрифтом виділено біти числа i, які відрізняються від відповідних бітів числа $i-1$. Видно, що біт наймолодшого (нульового) розряду (біт b_0) змінюється щоразу, біт b_1 змінюється кожен другий раз, а в загальному випадку, біт b_k змінюється у кожного 2^k-го цілого числа. У парних стовпцях таблиці розміщено всі послідовні коди Грея. Під таблицею наведено рекурсивну функцію Matlab, яка обчислює коди Грея.

Стиснення зображень. Коди Грея

43210	Gray	43210	Gray	43210	Gray	43210	Gray
00000	00000	**01000**	01100	**10000**	11000	**11000**	10100
0000**1**	0000**1**	0100**1**	0110**1**	1000**1**	1100**1**	1100**1**	1010**1**
000**10**	0001**1**	010**10**	0111**1**	100**10**	1101**1**	110**10**	1011**1**
0001**1**	0001**0**	0101**1**	0111**0**	1001**1**	1101**0**	1101**1**	1011**0**
00**100**	0011**0**	01**100**	0101**0**	10**100**	1111**0**	11**100**	1001**0**
0010**1**	0011**1**	0110**1**	0101**1**	1010**1**	1111**1**	1110**1**	1001**1**
001**10**	0010**1**	011**10**	0100**1**	101**10**	1110**1**	111**10**	1000**1**
0011**1**	0010**0**	0111**1**	0100**0**	1011**1**	1110**0**	1111**1**	1000**0**

```
function b=rgc(a,i)
[r,c]=size(a);
b= [zeros(r,1),a; ones(r,1),flipud(a)];
if i>1, b=rgc(b,i-1); end;
```

У табл. зображено, як змінюються окремі біти двійкових кодів, що становлять числа від 0 до 32. У цій таблиці в непарних стовпцях наведено звичайні двійкові коди цілих чисел. Жирним шрифтом виділено біти числа i, які відрізняються від відповідних бітів числа $i-1$. Видно, що біт наймолодшого (нульового) розряду (біт b_0) змінюється щоразу, біт b_1 змінюється кожен другий раз, а в загальному випадку, біт b_k змінюється у кожного 2^k-го цілого числа. У парних стовпцях таблиці розміщено всі послідовні коди Грея. Під таблицею наведено рекурсивну функцію Matlab, яка обчислює коди Грея.

Стиснення зображень. Коди Грея

43210	Gray	43210	Gray	43210	Gray	43210	Gray
00000	00000	**01000**	01100	**10000**	11000	**11000**	10100
0000**1**	0000**1**	0100**1**	0110**1**	1000**1**	1100**1**	1100**1**	1010**1**
000**10**	0001**1**	010**10**	0111**1**	100**10**	1101**1**	110**10**	1011**1**
0001**1**	0001**0**	0101**1**	0111**0**	1001**1**	1101**0**	1101**1**	1011**0**
00**100**	0011**0**	01**100**	0101**0**	10**100**	1111**0**	11**100**	1001**0**
0010**1**	0011**1**	0110**1**	0101**1**	1010**1**	1111**1**	1110**1**	1001**1**
001**10**	0010**1**	011**10**	0100**1**	101**10**	1110**1**	111**10**	1000**1**
0011**1**	0010**0**	0111**1**	0100**0**	1011**1**	1110**0**	1111**1**	1000**0**

```
function b=rgc(a,i)
[r,c]=size(a);
b= [zeros(r,1),a; ones(r,1),flipud(a)];
if i>1, b=rgc(b,i-1); end;
```

У табл. зображено, як змінюються окремі біти двійкових кодів, що становлять числа від 0 до 32. У цій таблиці в непарних стовпцях наведено звичайні двійкові коди цілих чисел. Жирним шрифтом виділено біти числа i, які відрізняються від відповідних бітів числа $i-1$. Видно, що біт наймолодшого (нульового) розряду (біт b_0) змінюється щоразу, біт b_1 змінюється кожен другий раз, а в загальному випадку, біт b_k змінюється у кожного 2^k-го цілого числа. У парних стовпцях таблиці розміщено всі послідовні коди Грея. Під таблицею наведено рекурсивну функцію Matlab, яка обчислює коди Грея.

Стиснення зображень. Коди Грея

43210	Gray	43210	Gray	43210	Gray	43210	Gray
00000	00000	**01000**	01100	**10000**	11000	**11000**	10100
0000**1**	0000**1**	0100**1**	0110**1**	1000**1**	1100**1**	1100**1**	1010**1**
000**10**	0001**1**	010**10**	0111**1**	100**10**	1101**1**	110**10**	1011**1**
0001**1**	0001**0**	0101**1**	0111**0**	1001**1**	1101**0**	1101**1**	1011**0**
00**100**	0011**0**	01**100**	0101**0**	10**100**	1111**0**	11**100**	1001**0**
0010**1**	0011**1**	0110**1**	0101**1**	1010**1**	1111**1**	1110**1**	1001**1**
001**10**	0010**1**	011**10**	0100**1**	101**10**	1110**1**	111**10**	1000**1**
0011**1**	0010**0**	0111**1**	0100**0**	1011**1**	1110**0**	1111**1**	1000**0**

```
function b=rgc(a,i)
[r,c]=size(a);
b= [zeros(r,1),a; ones(r,1),flipud(a)];
if i>1, b=rgc(b,i-1); end;
```

У табл. зображено, як змінюються окремі біти двійкових кодів, що становлять числа від 0 до 32. У цій таблиці в непарних стовпцях наведено звичайні двійкові коди цілих чисел. Жирним шрифтом виділено біти числа i, які відрізняються від відповідних бітів числа $i-1$. Видно, що біт наймолодшого (нульового) розряду (біт b_0) змінюється щоразу, біт b_1 змінюється кожен другий раз, а в загальному випадку, біт b_k змінюється у кожного 2^k-го цілого числа. У парних стовпцях таблиці розміщено всі послідовні коди Грея. Під таблицею наведено рекурсивну функцію Matlab, яка обчислює коди Грея.

Стиснення зображень. Коди Грея

43210	Gray	43210	Gray	43210	Gray	43210	Gray
00000	00000	**01000**	01100	**10000**	11000	**11000**	10100
0000**1**	00001	0100**1**	01101	1000**1**	11001	1100**1**	10101
000**10**	00011	010**10**	01111	100**10**	11011	110**10**	10111
0001**1**	00010	0101**1**	01110	1001**1**	11010	1101**1**	10110
00**100**	00110	01**100**	01010	10**100**	11110	11**100**	10010
0010**1**	00111	0110**1**	01011	1010**1**	11111	1110**1**	10011
001**10**	00101	011**10**	01001	101**10**	11101	111**10**	10001
0011**1**	00100	0111**1**	01000	1011**1**	11100	1111**1**	10000

```
function b=rgc(a,i)
[r,c]=size(a);
b= [zeros(r,1),a; ones(r,1),flipud(a)];
if i>1, b=rgc(b,i-1); end;
```

У табл. зображено, як змінюються окремі біти двійкових кодів, що становлять числа від 0 до 32. У цій таблиці в непарних стовпцях наведено звичайні двійкові коди цілих чисел. Жирним шрифтом виділено біти числа i, які відрізняються від відповідних бітів числа $i-1$. Видно, що біт наймолодшого (нульового) розряду (біт b_0) змінюється щоразу, біт b_1 змінюється кожен другий раз, а в загальному випадку, біт b_k змінюється у кожного 2^k-го цілого числа. У парних стовпцях таблиці розміщено всі послідовні коди Грея. Під таблицею наведено рекурсивну функцію Matlab, яка обчислює коди Грея.

Стиснення зображень. Коди Грея

43210	Gray	43210	Gray	43210	Gray	43210	Gray
00000	00000	**01000**	01100	**10000**	11000	**11000**	10100
0000**1**	0000**1**	0100**1**	0110**1**	1000**1**	1100**1**	1100**1**	1010**1**
000**10**	0001**1**	010**10**	0111**1**	100**10**	1101**1**	110**10**	1011**1**
0001**1**	0001**0**	0101**1**	0111**0**	1001**1**	1101**0**	1101**1**	1011**0**
00**100**	0011**0**	01**100**	0101**0**	10**100**	1111**0**	11**100**	1001**0**
0010**1**	0011**1**	0110**1**	0101**1**	1010**1**	1111**1**	1110**1**	1001**1**
001**10**	0010**1**	011**10**	0100**1**	101**10**	1110**1**	111**10**	1000**1**
0011**1**	0010**0**	0111**1**	0100**0**	1011**1**	1110**0**	1111**1**	1000**0**

```
function b=rgc(a,i)
[r,c]=size(a);
b= [zeros(r,1),a; ones(r,1),flipud(a)];
if i>1, b=rgc(b,i-1); end;
```

У табл. зображено, як змінюються окремі біти двійкових кодів, що становлять числа від 0 до 32. У цій таблиці в непарних стовпцях наведено звичайні двійкові коди цілих чисел. Жирним шрифтом виділено біти числа i, які відрізняються від відповідних бітів числа $i-1$. Видно, що біт наймолодшого (нульового) розряду (біт b_0) змінюється щоразу, біт b_1 змінюється кожен другий раз, а в загальному випадку, біт b_k змінюється у кожного 2^k-го цілого числа. У парних стовпцях таблиці розміщено всі послідовні коди Грея. Під таблицею наведено рекурсивну функцію Matlab, яка обчислює коди Грея.

Стиснення зображень. Коди Грея

43210	Gray	43210	Gray	43210	Gray	43210	Gray
00000	00000	**01000**	01100	**10000**	11000	**11000**	10100
0000**1**	0000**1**	0100**1**	0110**1**	1000**1**	1100**1**	1100**1**	1010**1**
000**10**	0001**1**	010**10**	0111**1**	100**10**	1101**1**	110**10**	1011**1**
0001**1**	0001**0**	0101**1**	0111**0**	1001**1**	1101**0**	1101**1**	1011**0**
00**100**	0011**0**	01**100**	0101**0**	10**100**	1111**0**	11**100**	1001**0**
0010**1**	0011**1**	0110**1**	0101**1**	1010**1**	1111**1**	1110**1**	1001**1**
001**10**	0010**1**	011**10**	0100**1**	101**10**	1110**1**	111**10**	1000**1**
0011**1**	0010**0**	0111**1**	0100**0**	1011**1**	1110**0**	1111**1**	1000**0**

```
function b=rgc(a,i)
[r,c]=size(a);
b= [zeros(r,1),a; ones(r,1),flipud(a)];
if i>1, b=rgc(b,i-1); end;
```

У табл. зображено, як змінюються окремі біти двійкових кодів, що становлять числа від 0 до 32. У цій таблиці в непарних стовпцях наведено звичайні двійкові коди цілих чисел. Жирним шрифтом виділено біти числа i, які відрізняються від відповідних бітів числа $i-1$. Видно, що біт наймолодшого (нульового) розряду (біт b_0) змінюється щоразу, біт b_1 змінюється кожен другий раз, а в загальному випадку, біт b_k змінюється у кожного 2^k-го цілого числа. У парних стовпцях таблиці розміщено всі послідовні коди Грея. Під таблицею наведено рекурсивну функцію Matlab, яка обчислює коди Грея.

Стиснення зображень. Коди Грея

43210	Gray	43210	Gray	43210	Gray	43210	Gray
00000	00000	**01000**	01100	**10000**	11000	**11000**	10100
0000**1**	00001	0100**1**	01101	1000**1**	11001	1100**1**	10101
000**10**	00011	010**10**	01111	100**10**	11011	110**10**	10111
0001**1**	00010	0101**1**	01110	1001**1**	11010	1101**1**	10110
00**100**	00110	01**100**	01010	10**100**	11110	11**100**	10010
0010**1**	00111	0110**1**	01011	1010**1**	11111	1110**1**	10011
001**10**	00101	011**10**	01001	101**10**	11101	111**10**	10001
0011**1**	00100	0111**1**	01000	1011**1**	11100	1111**1**	10000

```
function b=rgc(a,i)
[r,c]=size(a);
b= [zeros(r,1),a; ones(r,1),flipud(a)];
if i>1, b=rgc(b,i-1); end;
```

У табл. зображено, як змінюються окремі біти двійкових кодів, що становлять числа від 0 до 32. У цій таблиці в непарних стовпцях наведено звичайні двійкові коди цілих чисел. Жирним шрифтом виділено біти числа i, які відрізняються від відповідних бітів числа $i-1$. Видно, що біт наймолодшого (нульового) розряду (біт b_0) змінюється щоразу, біт b_1 змінюється кожен другий раз, а в загальному випадку, біт b_k змінюється у кожного 2^k-го цілого числа. У парних стовпцях таблиці розміщено всі послідовні коди Грея. Під таблицею наведено рекурсивну функцію Matlab, яка обчислює коди Грея.

Стиснення зображень. Коди Грея

43210	Gray	43210	Gray	43210	Gray	43210	Gray
00000	00000	**01000**	01100	**10000**	11000	**11000**	10100
0000**1**	00001	0100**1**	01101	1000**1**	11001	1100**1**	10101
000**10**	00011	010**10**	01111	100**10**	11011	110**10**	10111
0001**1**	00010	0101**1**	01110	1001**1**	11010	1101**1**	10110
00**100**	00110	01**100**	01010	10**100**	11110	11**100**	10010
0010**1**	00111	0110**1**	01011	1010**1**	11111	1110**1**	10011
0011**0**	00101	0111**0**	01001	1011**0**	11101	1111**0**	10001
0011**1**	00100	0111**1**	01000	1011**1**	11100	1111**1**	10000

```
function b=rgc(a,i)
[r,c]=size(a);
b= [zeros(r,1),a; ones(r,1),flipud(a)];
if i>1, b=rgc(b,i-1); end;
```

У табл. зображено, як змінюються окремі біти двійкових кодів, що становлять числа від 0 до 32. У цій таблиці в непарних стовпцях наведено звичайні двійкові коди цілих чисел. Жирним шрифтом виділено біти числа i, які відрізняються від відповідних бітів числа $i-1$. Видно, що біт наймолодшого (нульового) розряду (біт b_0) змінюється щоразу, біт b_1 змінюється кожен другий раз, а в загальному випадку, біт b_k змінюється у кожного 2^k-го цілого числа. У парних стовпцях таблиці розміщено всі послідовні коди Грея. Під таблицею наведено рекурсивну функцію Matlab, яка обчислює коди Грея.

Стиснення зображень. Коди Грея

43210	Gray	43210	Gray	43210	Gray	43210	Gray
00000	00000	**01000**	01100	**10000**	11000	**11000**	10100
0000**1**	00001	0100**1**	01101	1000**1**	11001	1100**1**	10101
000**10**	00011	010**10**	01111	100**10**	11011	110**10**	10111
0001**1**	00010	0101**1**	01110	1001**1**	11010	1101**1**	10110
00**100**	00110	01**100**	01010	10**100**	11110	11**100**	10010
0010**1**	00111	0110**1**	01011	1010**1**	11111	1110**1**	10011
0011**0**	00101	0111**0**	01001	1011**0**	11101	1111**0**	10001
0011**1**	00100	0111**1**	01000	1011**1**	11100	1111**1**	10000

```
a=linspace(0,31,32); b=bitshift(a,-1);
b=bitxor(a,b); dec2bin(b)
```

Коди RGC можна побудувати для цілого числа n, і не вдаючись до рекурсивної процедури. Достатньо застосувати функцію "ВИКЛЮЧНЕ АБО" до n і до його логічного зсуву на один розряд вправо. Мовою C це запишеться наступними операторами:

```
n^(n»1).
```

Табл. була породжена саме цим виразом.

Стиснення зображень. Коди Грея

43210	Gray	43210	Gray	43210	Gray	43210	Gray
00000	00000	**01000**	01100	**10000**	11000	**11000**	10100
0000**1**	00001	0100**1**	01101	1000**1**	11001	1100**1**	10101
000**10**	00011	010**10**	01111	100**10**	11011	110**10**	10111
0001**1**	00010	0101**1**	01110	1001**1**	11010	1101**1**	10110
00**100**	00110	01**100**	01010	10**100**	11110	11**100**	10010
0010**1**	00111	0110**1**	01011	1010**1**	11111	1110**1**	10011
0011**0**	00101	0111**0**	01001	1011**0**	11101	1111**0**	10001
0011**1**	00100	0111**1**	01000	1011**1**	11100	1111**1**	10000

```
a=linspace(0,31,32); b=bitshift(a,-1);
b=bitxor(a,b); dec2bin(b)
```

Коди RGC можна побудувати для цілого числа n, і не вдаючись до рекурсивної процедури. Достатньо застосувати функцію "ВИКЛЮЧНЕ АБО" до n і до його логічного зсуву на один розряд вправо. Мовою C це запишеться наступними операторами:

```
n^(n»1).
```

Табл. була породжена саме цим виразом.

Стиснення зображень. Коди Грея

43210	Gray	43210	Gray	43210	Gray	43210	Gray
00000	00000	**01000**	01100	**10000**	11000	**11000**	10100
0000**1**	00001	0100**1**	01101	1000**1**	11001	1100**1**	10101
000**10**	00011	010**10**	01111	100**10**	11011	110**10**	10111
0001**1**	00010	0101**1**	01110	1001**1**	11010	1101**1**	10110
00**100**	00110	01**100**	01010	10**100**	11110	11**100**	10010
0010**1**	00111	0110**1**	01011	1010**1**	11111	1110**1**	10011
0011**0**	00101	0111**0**	01001	1011**0**	11101	1111**0**	10001
0011**1**	00100	0111**1**	01000	1011**1**	11100	1111**1**	10000

```
a=linspace(0,31,32); b=bitshift(a,-1);
b=bitxor(a,b); dec2bin(b)
```

Коди RGC можна побудувати для цілого числа n, і не вдаючись до рекурсивної процедури. Достатньо застосувати функцію "ВИКЛЮЧНЕ АБО" до n і до його логічного зсуву на один розряд вправо. Мовою C це запишеться наступними операторами:

```
n^(n»1).
```

Табл. була породжена саме цим виразом.

Стиснення зображень. Коди Грея

43210	Gray	43210	Gray	43210	Gray	43210	Gray
00000	00000	**01000**	01100	**10000**	11000	**11000**	10100
0000**1**	00001	0100**1**	01101	1000**1**	11001	1100**1**	10101
000**10**	00011	010**10**	01111	100**10**	11011	110**10**	10111
0001**1**	00010	0101**1**	01110	1001**1**	11010	1101**1**	10110
00**100**	00110	01**100**	01010	10**100**	11110	11**100**	10010
0010**1**	00111	0110**1**	01011	1010**1**	11111	1110**1**	10011
0011**0**	00101	0111**0**	01001	1011**0**	11101	1111**0**	10001
0011**1**	00100	0111**1**	01000	1011**1**	11100	1111**1**	10000

```
a=linspace(0,31,32); b=bitshift(a,-1);
b=bitxor(a,b); dec2bin(b)
```

Коди RGC можна побудувати для цілого числа n, і не вдаючись до рекурсивної процедури. Достатньо застосувати функцію "ВИКЛЮЧНЕ АБО" до n і до його логічного зсуву на один розряд вправо. Мовою C це запишеться наступними операторами:

```
n^(n»1).
```

Табл. була породжена саме цим виразом.

Стиснення зображень. Коди Грея

43210	Gray	43210	Gray	43210	Gray	43210	Gray
00000	00000	**0**1000	01100	**1**0000	11000	**1**1000	10100
0000**1**	00001	0100**1**	01101	1000**1**	11001	1100**1**	10101
000**10**	00011	010**10**	01111	100**10**	11011	110**10**	10111
0001**1**	00010	0101**1**	01110	1001**1**	11010	1101**1**	10110
00**100**	00110	01**100**	01010	10**100**	11110	11**100**	10010
00**1**0**1**	00111	01**1**0**1**	01011	10**1**0**1**	11111	11**1**0**1**	10011
00**11****0**	00101	01**11****0**	01001	10**11****0**	11101	11**11****0**	10001
0011**1**	00100	0111**1**	01000	1011**1**	11100	1111**1**	10000

```
a=linspace(0,31,32); b=bitshift(a,-1);
b=bitxor(a,b); dec2bin(b)
```

Коди RGC можна побудувати для цілого числа n, і не вдаючись до рекурсивної процедури. Достатньо застосувати функцію "ВИКЛЮЧНЕ АБО" до n і до його логічного зсуву на один розряд вправо. Мовою C це запишеться наступними операторами:

```
n^(n»1).
```

Табл. була породжена саме цим виразом.

Стиснення зображень. Коди Грея

43210	Gray	43210	Gray	43210	Gray	43210	Gray
00000	00000	**01000**	01100	**10000**	11000	**11000**	10100
0000**1**	00001	0100**1**	01101	1000**1**	11001	1100**1**	10101
000**10**	00011	010**10**	01111	100**10**	11011	110**10**	10111
0001**1**	00010	0101**1**	01110	1001**1**	11010	1101**1**	10110
00**100**	00110	01**100**	01010	10**100**	11110	11**100**	10010
0010**1**	00111	0110**1**	01011	1010**1**	11111	1110**1**	10011
0011**0**	00101	0111**0**	01001	1011**0**	11101	1111**0**	10001
0011**1**	00100	0111**1**	01000	1011**1**	11100	1111**1**	10000

```
a=linspace(0,31,32); b=bitshift(a,-1);
b=bitxor(a,b); dec2bin(b)
```

Коди RGC можна побудувати для цілого числа n, і не вдаючись до рекурсивної процедури. Достатньо застосувати функцію "ВИКЛЮЧНЕ АБО" до n і до його логічного зсуву на один розряд вправо. Мовою C це запишеться наступними операторами:

```
n^(n»1).
```

Табл. була породжена саме цим виразом.

Стиснення зображень. Коди Грея

43210	Gray	43210	Gray	43210	Gray	43210	Gray
00000	00000	**01000**	01100	**10000**	11000	**11000**	10100
0000**1**	00001	0100**1**	01101	1000**1**	11001	1100**1**	10101
000**10**	00011	010**10**	01111	100**10**	11011	110**10**	10111
0001**1**	00010	0101**1**	01110	1001**1**	11010	1101**1**	10110
00**100**	00110	01**100**	01010	10**100**	11110	11**100**	10010
0010**1**	00111	0110**1**	01011	1010**1**	11111	1110**1**	10011
0011**0**	00101	0111**0**	01001	1011**0**	11101	1111**0**	10001
0011**1**	00100	0111**1**	01000	1011**1**	11100	1111**1**	10000

```
a=linspace(0,31,32); b=bitshift(a,-1);
b=bitxor(a,b); dec2bin(b)
```

Коди RGC можна побудувати для цілого числа n, і не вдаючись до рекурсивної процедури. Достатньо застосувати функцію "ВИКЛЮЧНЕ АБО" до n і до його логічного зсуву на один розряд вправо. Мовою C це запишеться наступними операторами:

```
n^(n»1).
```

Табл. була породжена саме цим виразом.

Стиснення зображень. Коди Грея

Можна зробити такий висновок. Шар, який відповідає найстаршому двійковому розряду (у звичному уявленні) зображення, підпорядковується принципу стискування більшої, ніж шар наймолодшого розряду. Коли сусідні пікселі мають значення, які різняться на одиницю (рівні p і $p+1$), шанси на те, що їхні менш значущі або більш значущі розряди відрізняються, однакові. Отже, будь-який метод компресії, який окремо стискати шари повинен або враховувати цю особливість розрядності шарів, або використовувати коди Грея для представлення величин пікселів.

На наступних трьох рис. зображено вісім побітих шарів картинки “папуги” у звичайному двійковому зображенні (ліворуч) та у вигляді кодів Грея (праворуч).

Можна зробити такий висновок. Шар, який відповідає найстаршому двійковому розряду (у звичному уявленні) зображення, підпорядковується принципу стискування більшої, ніж шар наймолодшого розряду. Коли сусідні пікселі мають значення, які різняться на одиницю (рівні p і $p+1$), шанси на те, що їхні менш значущі або більш значущі розряди відрізняються, однакові. Отже, будь-який метод компресії, який окремо стискати шари повинен або враховувати цю особливість розрядності шарів, або використовувати коди Грея для представлення величин пікселів.

На наступних трьох рис. зображено вісім побітих шарів картинки “папуги” у звичайному двійковому зображенні (ліворуч) та у вигляді кодів Грея (праворуч).

Стиснення зображень. Коди Грея

Можна зробити такий висновок. Шар, який відповідає найстаршому двійковому розряду (у звичному уявленні) зображення, підпорядковується принципу стискування більшої, ніж шар наймолодшого розряду. Коли сусідні пікселі мають значення, які різняться на одиницю (рівні p і $p+1$), шанси на те, що їхні менш значущі або більш значущі розряди відрізняються, однакові. Отже, будь-який метод компресії, який окремо стискати шари повинен або враховувати цю особливість розрядності шарів, або використовувати коди Грея для представлення величин пікселів.

На наступних трьох рис. зображено вісім побітих шарів картинки “папуги” у звичайному двійковому зображенні (ліворуч) та у вигляді кодів Грея (праворуч).

Стиснення зображень. Коди Грея

Можна зробити такий висновок. Шар, який відповідає найстаршому двійковому розряду (у звичному уявленні) зображення, підпорядковується принципу стискування більшої, ніж шар наймолодшого розряду. Коли сусідні пікселі мають значення, які різняться на одиницю (рівні p і $p+1$), шанси на те, що їхні менш значущі або більш значущі розряди відрізняються, однакові. Отже, будь-який метод компресії, який окремо стискати шари повинен або враховувати цю особливість розрядності шарів, або використовувати коди Грея для представлення величин пікселів.

На наступних трьох рис. зображено вісім побітих шарів картинки “папуги” у звичайному двійковому зображенні (ліворуч) та у вигляді кодів Грея (праворуч).

Стиснення зображень. Коди Грея

Можна зробити такий висновок. Шар, який відповідає найстаршому двійковому розряду (у звичному уявленні) зображення, підпорядковується принципу стискування більшої, ніж шар наймолодшого розряду. Коли сусідні пікселі мають значення, які різняться на одиницю (рівні p і $p+1$), шанси на те, що їхні менш значущі або більш значущі розряди відрізняються, однакові. Отже, будь-який метод компресії, який окремо стискати шари повинен або враховувати цю особливість розрядності шарів, або використовувати коди Грея для представлення величин пікселів.

На наступних трьох рис. зображено вісім побітих шарів картинки “папуги” у звичайному двійковому зображенні (ліворуч) та у вигляді кодів Грея (праворуч).

Стиснення зображень. Коди Грея

Можна зробити такий висновок. Шар, який відповідає найстаршому двійковому розряду (у звичному уявленні) зображення, підпорядковується принципу стискування більшої, ніж шар наймолодшого розряду. Коли сусідні пікселі мають значення, які різняться на одиницю (рівні p і $p+1$), шанси на те, що їхні менш значущі або більш значущі розряди відрізняються, однакові. Отже, будь-який метод компресії, який окремо стискати шари повинен або враховувати цю особливість розрядності шарів, або використовувати коди Грея для представлення величин пікселів.

На наступних трьох рис. зображено вісім побітих шарів картинки “папуги” у звичайному двійковому зображенні (ліворуч) та у вигляді кодів Грея (праворуч).

Стиснення зображень. Коди Грея

Можна зробити такий висновок. Шар, який відповідає найстаршому двійковому розряду (у звичному уявленні) зображення, підпорядковується принципу стискування більшої, ніж шар наймолодшого розряду. Коли сусідні пікселі мають значення, які різняться на одиницю (рівні p і $p+1$), шанси на те, що їхні менш значущі або більш значущі розряди відрізняються, однакові. Отже, будь-який метод компресії, який окремо стискати шари повинен або враховувати цю особливість розрядності шарів, або використовувати коди Грея для представлення величин пікселів.

На наступних трьох рис. зображено вісім побітих шарів картинки "папуги" у звичайному двійковому зображенні (ліворуч) та у вигляді кодів Грея (праворуч).

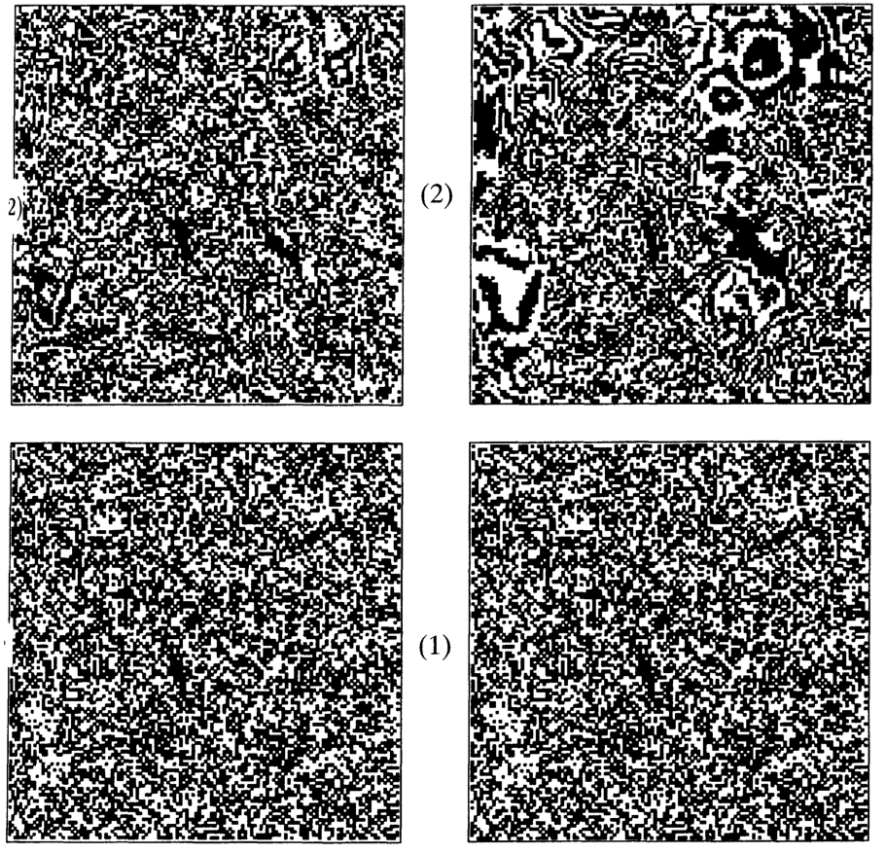
Можна зробити такий висновок. Шар, який відповідає найстаршому двійковому розряду (у звичному уявленні) зображення, підпорядковується принципу стискування більшої, ніж шар наймолодшого розряду. Коли сусідні пікселі мають значення, які різняться на одиницю (рівні p і $p+1$), шанси на те, що їхні менш значущі або більш значущі розряди відрізняються, однакові. Отже, будь-який метод компресії, який окремо стискати шари повинен або враховувати цю особливість розрядності шарів, або використовувати коди Грея для представлення величин пікселів.

На наступних трьох рис. зображено вісім побітих шарів картинки “папуги” у звичайному двійковому зображенні (ліворуч) та у вигляді кодів Грея (праворуч).

Можна зробити такий висновок. Шар, який відповідає найстаршому двійковому розряду (у звичному уявленні) зображення, підпорядковується принципу стискування більшої, ніж шар наймолодшого розряду. Коли сусідні пікселі мають значення, які різняться на одиницю (рівні p і $p+1$), шанси на те, що їхні менш значущі або більш значущі розряди відрізняються, однакові. Отже, будь-який метод компресії, який окремо стискати шари повинен або враховувати цю особливість розрядності шарів, або використовувати коди Грея для представлення величин пікселів.

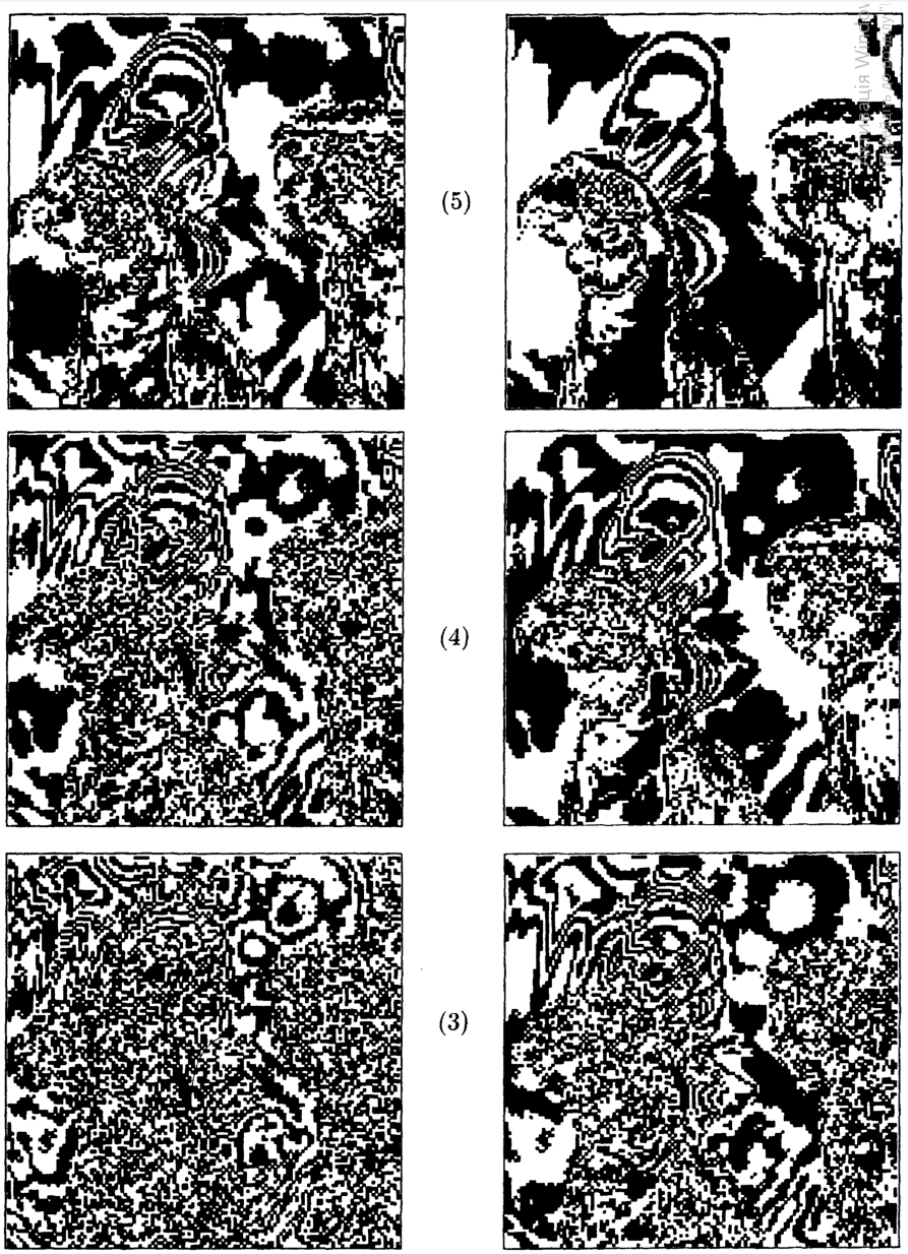
На наступних трьох рис. зображено вісім побітих шарів картинки “папуги” у звичайному двійковому зображенні (ліворуч) та у вигляді кодів Грея (праворуч).

Стиснення зображень. Коди Грея



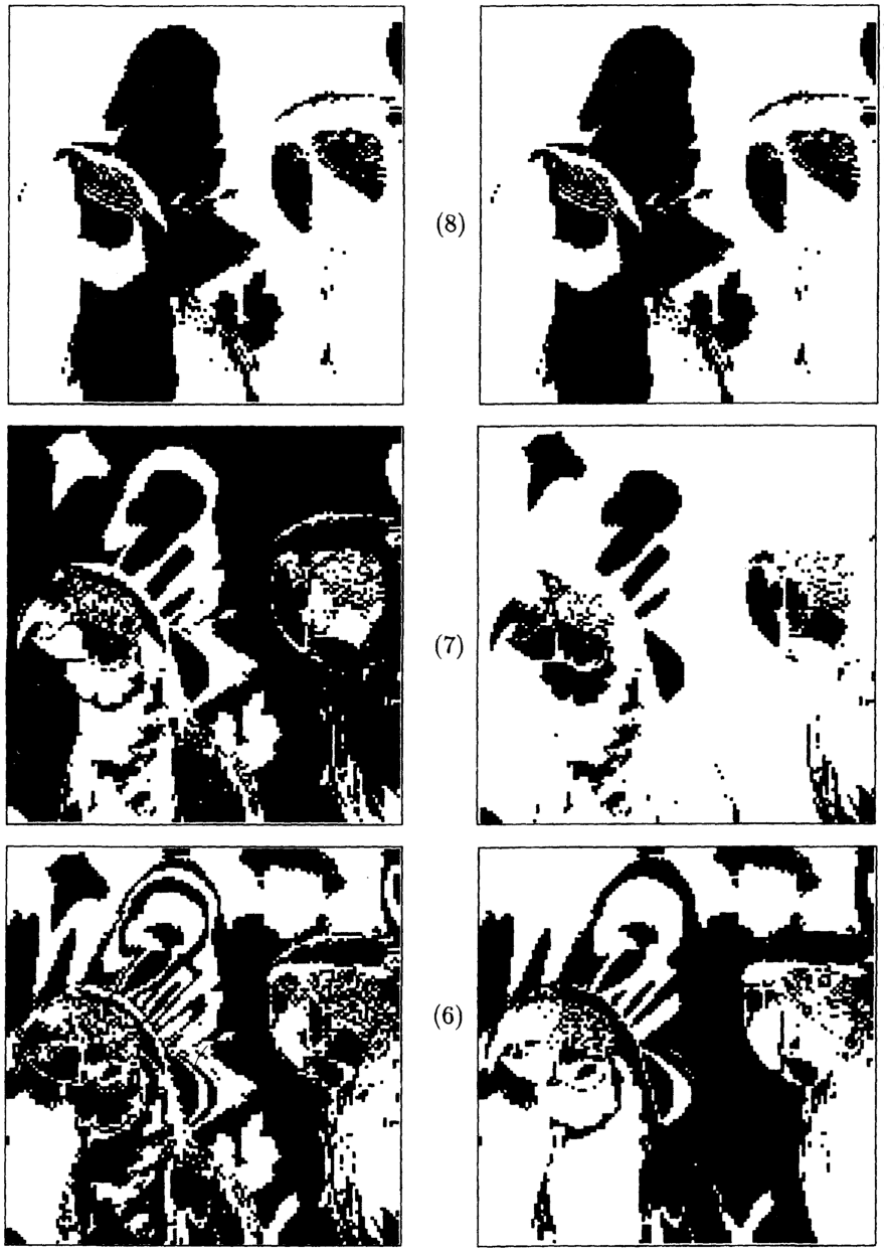
Двійковий код. Код Грея (Шари 1 та 2 зображення “папуги”)

Стиснення зображень. Коди Грея



Двійковий код Код Грея (Шари 3, 4 та 5 зображення “папуги”)

Стиснення зображень. Коди Грея



Двійковий код Код Грея (Шари 6, 7 та 8 зображення “папуги”)

Стиснення зображень. Коди Грея

Шари, побудовані за допомогою програми Matlab, наведено на рис.

```
clear;
filename='parrot si 28'; diin=128;
fid=fopen(filename,'r');
img=fread(fid,[dim,dim])';
mask=1; */. між 1 и 8


nimg=bitget(img,mask);
imagesc(nimg), colormap(gray)
```

Двійковий код

```
clear;
filename='parrotsi28'; dim=128;
fid=fopen(filename,'r');
img=fread(fid,[dim,dim])';
mask=1; */. між 1 и 8
a=bitshift(img,-1);
b=bitxor(img,a);
nimg=bitget(b,mask);
imagesc(nimg), colormap(gray)
```

Код Грея

Вони занумеровані числами від 8 (найстарший, найбільш значний біт) до 1 (наймолодший біт). Очевидно, що шар наймолодшого біта не показує жодної кореляції між пікселями; вони випадкові або близькі до випадкових для обох уявлень, двійкового та RGC. Однак шари від 2 до 5 демонструють велику кореляцію пікселів для кодів Грея. Шари з 6 по 8 виглядають по-різному для двійкових кодів та для кодів Грея, але здаються сильно корельованими в обох випадках.

Стиснення зображень. Коди Грея

Шари, побудовані за допомогою програми Matlab, наведено на рис.

```
clear;
filename='parrot si 28'; diin=128;
fid=fopen(filename,'r');
img=fread(fid,[dim,dim])';
mask=1; */. між 1 и 8


nimg=bitget(img,mask);
imagesc(nimg), colormap(gray)
```

Двійковий код

```
clear;
filename='parrotsi28'; dim=128;
fid=fopen(filename,'r');
img=fread(fid,[dim,dim])';
mask=1; */. між 1 и 8
a=bitshift(img,-1);
b=bitxor(img,a);
nimg=bitget(b,mask);
imagesc(nimg), colormap(gray)
```

Код Грея

Вони занумеровані числами від 8 (найстарший, найбільш значний біт) до 1 (наймолодший біт). Очевидно, що шар наймолодшого біта не показує жодної кореляції між пікселями; вони випадкові або близькі до випадкових для обох уявлень, двійкового та RGC. Однак шари від 2 до 5 демонструють велику кореляцію пікселів для кодів Грея. Шари з 6 по 8 виглядають по-різному для двійкових кодів та для кодів Грея, але здаються сильно корельованими в обох випадках.

Стиснення зображень. Коди Грея

Шари, побудовані за допомогою програми Matlab, наведено на рис.

```
clear;
filename='parrot si 28'; diin=128;
fid=fopen(filename,'r');
img=fread(fid,[dim,dim])';
mask=1; */. між 1 и 8


nimg=bitget(img,mask);
imagesc(nimg), colormap(gray)
```

Двійковий код

```
clear;
filename='parrotsi28'; dim=128;
fid=fopen(filename,'r');
img=fread(fid,[dim,dim])';
mask=1; */. між 1 и 8
a=bitshift(img,-1);
b=bitxor(img,a);
nimg=bitget(b,mask);
imagesc(nimg), colormap(gray)
```

Код Грея

Вони занумеровані числами від 8 (найстарший, найбільш значний біт) до 1 (наймолодший біт). Очевидно, що шар наймолодшого біта не показує жодної кореляції між пікселями; вони випадкові або близькі до випадкових для обох уявлень, двійкового та RGC. Однак шари від 2 до 5 демонструють велику кореляцію пікселів для кодів Грея. Шари з 6 по 8 виглядають по-різному для двійкових кодів та для кодів Грея, але здаються сильно корельованими в обох випадках.

Стиснення зображень. Коди Грея

Шари, побудовані за допомогою програми Matlab, наведено на рис.

```
clear;
filename='parrot si 28'; diin=128;
fid=fopen(filename,'r');
img=fread(fid,[dim,dim])';
mask=1; */. між 1 и 8


nimg=bitget(img,mask);
imagesc(nimg), colormap(gray)
```

Двійковий код

```
clear;
filename='parrotsi28'; dim=128;
fid=fopen(filename,'r');
img=fread(fid,[dim,dim])';
mask=1; */. між 1 и 8
a=bitshift(img,-1);
b=bitxor(img,a);
nimg=bitget(b,mask);
imagesc(nimg), colormap(gray)
```

Код Грея

Вони занумеровані числами від 8 (найстарший, найбільш значний біт) до 1 (наймолодший біт). Очевидно, що шар наймолодшого біта не показує жодної кореляції між пікселями; вони випадкові або близькі до випадкових для обох уявлень, двійкового та RGC. Однак шари від 2 до 5 демонструють велику кореляцію пікселів для кодів Грея. Шари з 6 по 8 виглядають по-різному для двійкових кодів та для кодів Грея, але здаються сильно корельованими в обох випадках.

Стиснення зображень. Коди Грея

Шари, побудовані за допомогою програми Matlab, наведено на рис.

```
clear;
filename='parrot si 28'; diin=128;
fid=fopen(filename,'r');
img=fread(fid,[dim,dim])';
mask=1; */. між 1 и 8


nimg=bitget(img,mask);
imagesc(nimg), colormap(gray)
```

Двійковий код

```
clear;
filename='parrotsi28'; dim=128;
fid=fopen(filename,'r');
img=fread(fid,[dim,dim])';
mask=1; */. між 1 и 8
a=bitshift(img,-1);
b=bitxor(img,a);
nimg=bitget(b,mask);
imagesc(nimg), colormap(gray)
```

Код Грея

Вони занумеровані числами від 8 (найстарший, найбільш значний біт) до 1 (наймолодший біт). Очевидно, що шар наймолодшого біта не показує жодної кореляції між пікселями; вони випадкові або близькі до випадкових для обох уявлень, двійкового та RGC. Однак шари від 2 до 5 демонструють велику кореляцію пікселів для кодів Грея. Шари з 6 по 8 виглядають по-різному для двійкових кодів та для кодів Грея, але здаються сильно корельованими в обох випадках.

Стиснення зображень. Коди Грея

Шари, побудовані за допомогою програми Matlab, наведено на рис.

```
clear;
filename='parrot si 28'; diin=128;
fid=fopen(filename,'r');
img=fread(fid,[dim,dim])';
mask=1; */. між 1 и 8


nimg=bitget(img,mask);
imagesc(nimg), colormap(gray)
```

Двійковий код

```
clear;
filename='parrotsi28'; dim=128;
fid=fopen(filename,'r');
img=fread(fid,[dim,dim])';
mask=1; */. між 1 и 8
a=bitshift(img,-1);
b=bitxor(img,a);
nimg=bitget(b,mask);
imagesc(nimg), colormap(gray)
```

Код Грея

Вони занумеровані числами від 8 (найстарший, найбільш значний біт) до 1 (наймолодший біт). Очевидно, що шар наймолодшого біта не показує жодної кореляції між пікселями; вони випадкові або близькі до випадкових для обох уявлень, двійкового та RGC. Однак шари від 2 до 5 демонструють велику кореляцію пікселів для кодів Грея. Шари з 6 по 8 виглядають по-різному для двійкових кодів та для кодів Грея, але здаються сильно корельованими в обох випадках.

Стиснення зображень. Коди Грея

Шари, побудовані за допомогою програми Matlab, наведено на рис.

```
clear;
filename='parrot si 28'; diin=128;
fid=fopen(filename,'r');
img=fread(fid,[dim,dim])';
mask=1; */. між 1 и 8


nimg=bitget(img,mask);
imagesc(nimg), colormap(gray)
```

Двійковий код

```
clear;
filename='parrotsi28'; dim=128;
fid=fopen(filename,'r');
img=fread(fid,[dim,dim])';
mask=1; */. між 1 и 8
a=bitshift(img,-1);
b=bitxor(img,a);
nimg=bitget(b,mask);
imagesc(nimg), colormap(gray)
```

Код Грея

Вони занумеровані числами від 8 (найстарший, найбільш значний біт) до 1 (наймолодший біт). Очевидно, що шар наймолодшого біта не показує жодної кореляції між пікселями; вони випадкові або близькі до випадкових для обох уявлень, двійкового та RGC. Однак шари від 2 до 5 демонструють велику кореляцію пікселів для кодів Грея. Шари з 6 по 8 виглядають по-різному для двійкових кодів та для кодів Грея, але здаються сильно корельованими в обох випадках.

Стиснення зображень. Коди Грея

Шари, побудовані за допомогою програми Matlab, наведено на рис.

```
clear;
filename='parrot si 28'; diin=128;
fid=fopen(filename,'r');
img=fread(fid,[dim,dim])';
mask=1; */. між 1 и 8


nimg=bitget(img,mask);
imagesc(nimg), colormap(gray)
```

Двійковий код

```
clear;
filename='parrotsi28'; dim=128;
fid=fopen(filename,'r');
img=fread(fid,[dim,dim])';
mask=1; */. між 1 и 8
a=bitshift(img,-1);
b=bitxor(img,a);
nimg=bitget(b,mask);
imagesc(nimg), colormap(gray)
```

Код Грея

Вони занумеровані числами від 8 (найстарший, найбільш значний біт) до 1 (наймолодший біт). Очевидно, що шар наймолодшого біта не показує жодної кореляції між пікселями; вони випадкові або близькі до випадкових для обох уявлень, двійкового та RGC. Однак шари від 2 до 5 демонструють велику кореляцію пікселів для кодів Грея. Шари з 6 по 8 виглядають по-різному для двійкових кодів та для кодів Грея, але здаються сильно корельованими в обох випадках.

Стиснення зображень. Коди Грея

Шари, побудовані за допомогою програми Matlab, наведено на рис.

```
clear;
filename='parrot si 28'; diin=128;
fid=fopen(filename,'r');
img=fread(fid,[dim,dim])';
mask=1; */. між 1 и 8


nimg=bitget(img,mask);
imagesc(nimg), colormap(gray)
```

Двійковий код

```
clear;
filename='parrotsi28'; dim=128;
fid=fopen(filename,'r');
img=fread(fid,[dim,dim])';
mask=1; */. між 1 и 8
a=bitshift(img,-1);
b=bitxor(img,a);
nimg=bitget(b,mask);
imagesc(nimg), colormap(gray)
```

Код Грея

Вони занумеровані числами від 8 (найстарший, найбільш значний біт) до 1 (наймолодший біт). Очевидно, що шар наймолодшого біта не показує жодної кореляції між пікселями; вони випадкові або близькі до випадкових для обох уявлень, двійкового та RGC. Однак шари від 2 до 5 демонструють велику кореляцію пікселів для кодів Грея. Шари з 6 по 8 виглядають по-різному для двійкових кодів та для кодів Грея, але здаються сильно корельованими в обох випадках.

Стиснення зображень. Коди Грея

Шари, побудовані за допомогою програми Matlab, наведено на рис.

```
clear;
filename='parrot si 28'; diin=128;
fid=fopen(filename,'r');
img=fread(fid,[dim,dim])';
mask=1; */. між 1 и 8


nimg=bitget(img,mask);
imagesc(nimg), colormap(gray)
```

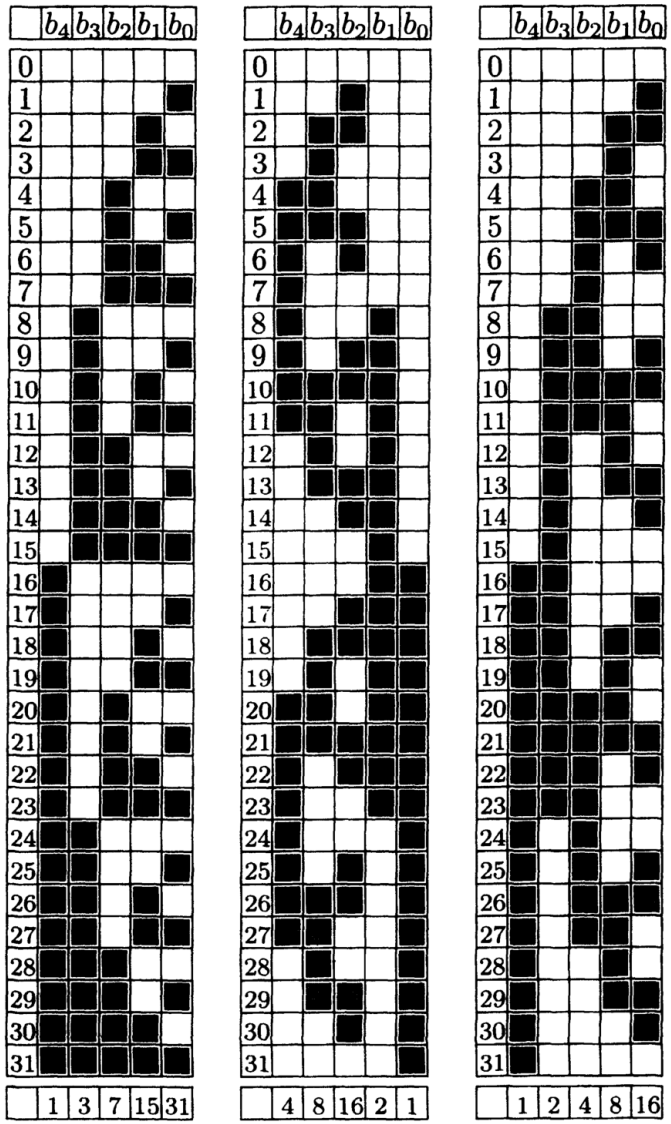
Двійковий код

```
clear;
filename='parrotsi28'; dim=128;
fid=fopen(filename,'r');
img=fread(fid,[dim,dim])';
mask=1; */. між 1 и 8
a=bitshift(img,-1);
b=bitxor(img,a);
nimg=bitget(b,mask);
imagesc(nimg), colormap(gray)
```

Код Грея

Вони занумеровані числами від 8 (найстарший, найбільш значний біт) до 1 (наймолодший біт). Очевидно, що шар наймолодшого біта не показує жодної кореляції між пікселями; вони випадкові або близькі до випадкових для обох уявлень, двійкового та RGC. Однак шари від 2 до 5 демонструють велику кореляцію пікселів для кодів Грея. Шари з 6 по 8 виглядають по-різному для двійкових кодів та для кодів Грея, але здаються сильно корельованими в обох випадках.

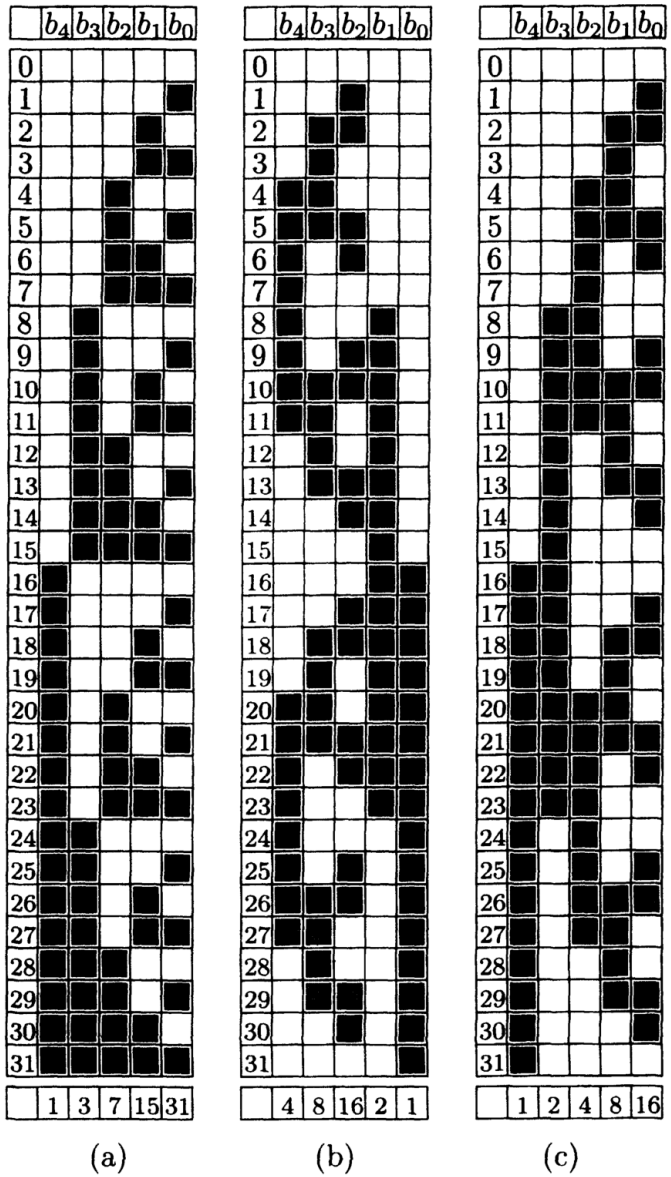
Стиснення зображень. Коди Грея



(a) (b) (c)

На рис. зображено дві версії перших 32 кодів Грея в їхньому графічному зображенні.

Стиснення зображень. Коди Грея



На рис. зображено дві версії перших 32 кодів Грея в їхньому графічному зображенні.

Стиснення зображень. Коди Грея

На рис. (b) показані коди із табл.,

43210	Gray
00000	00000
00001	00100
00010	01100
00011	01000
00100	11000
00101	11100
00110	10100
00111	10000

43210	Gray
01000	10010
01001	10110
01010	11110
01011	11010
01100	01010
01101	01110
01110	00110
01111	00010

43210	Gray
10000	00011
10001	00111
10010	01111
10011	01011
10100	11011
10101	11111
10110	10111
10111	10011

```
function b=rgc(a,i)
[r,c]=size(a);
b= [zeros(r,1),a; ones(r,1),flipud(a)];
if i>1, b=rgc(b,i-1); end;
```

На рис. (b) показані коди із табл.,

43210	Gray	43210	Gray	43210	Gray
00000	00000	**01000**	10010	**10000**	00011
00001	00100	01001	10110	10001	00111
00010	01100	01010	11110	10010	01111
00011	01000	01011	11010	10011	01011
00100	11000	01100	01010	10100	11011
00101	11100	01101	01110	10101	11111
00110	10100	01110	00110	10110	10111
00111	10000	01111	00010	10111	10011

```
function b=rgc(a,i)
[r,c]=size(a);
b= [zeros(r,1),a; ones(r,1),flipud(a)];
if i>1, b=rgc(b,i-1); end;
```

а на рис. (c) наведені коди з табл.

43210	Gray	43210	Gray	43210	Gray
00000	00000	**01000**	01100	**10000**	11000
0000**1**	00001	0100**1**	01101	1000**1**	11001
000**10**	00011	010**10**	01111	100**10**	11011
0001**1**	00010	0101**1**	01110	1001**1**	11010
00**100**	00110	01**100**	01010	10**100**	11110
00101	00111	01101	01011	10101	11111
00**110**	00101	01**110**	01001	10**110**	11101
0011**1**	00100	0111**1**	01000	1011**1**	11100

```
a=linspace(0,31,32); b=bitshift(a,-1);
b=bitxor(a,b); dec2bin(b)
```

а на рис. (c) наведені коди з табл.

43210	Gray	43210	Gray	43210	Gray
00000	00000	**01000**	01100	**10000**	11000
0000**1**	00001	0100**1**	01101	1000**1**	11001
000**10**	00011	010**10**	01111	100**10**	11011
0001**1**	00010	0101**1**	01110	1001**1**	11010
00**100**	00110	01**100**	01010	10**100**	11110
00101	00111	01101	01011	10101	11111
00**110**	00101	01**110**	01001	10**110**	11101
0011**1**	00100	0111**1**	01000	1011**1**	11100

```
a=linspace(0,31,32); b=bitshift(a,-1);
b=bitxor(a,b); dec2bin(b)
```

Стиснення зображень. Коди Грея

Незважаючи на те, що обидві сім'ї утворюють коди Грея, в них по-різному чергуються біти 0 і 1 у різних побитих шарах. На рис. (b) найважливіший біт змінюється 4 рази. Другий значний біт змінюється 8 разів, а біти трьох молодших розрядів змінюються, відповідно, 16, 2 і один раз. Якщо використовувати цю множину кодів Грея для розшарування зображення, то середні шари, очевидно, утворюють найменшу кореляцію між сусідніми пікселями.

На противагу цьому коди рис. (c) демонструватимуть велику кореляцію в шарах старших розрядів, оскільки там зміна між 0 і 1 відбувається, відповідно, 1, 2 і 4 рази (за старшинством розрядів). А молодші розряди покажуть слабку кореляцію пікселів, яка наближається до нуля, як і при випадковому розподілі пікселів. Для порівняння на рис. (a) зображено звичайні двійкові коди. Видно, що в цих кодах розряди змінюються суттєво частіше.

Стиснення зображень. Коди Грея

Незважаючи на те, що обидві сім'ї утворюють коди Грея, в них по-різному чергуються біти 0 і 1 у різних побитих шарах. На рис. (b) найважливіший біт змінюється 4 рази. Другий значний біт змінюється 8 разів, а біти трьох молодших розрядів змінюються, відповідно, 16, 2 і один раз. Якщо використовувати цю множину кодів Грея для розшарування зображення, то середні шари, очевидно, утворюють найменшу кореляцію між сусідніми пікселями.

На противагу цьому коди рис. (c) демонструватимуть велику кореляцію в шарах старших розрядів, оскільки там зміна між 0 і 1 відбувається, відповідно, 1, 2 і 4 рази (за старшинством розрядів). А молодші розряди покажуть слабку кореляцію пікселів, яка наближається до нуля, як і при випадковому розподілі пікселів. Для порівняння на рис. (a) зображено звичайні двійкові коди. Видно, що в цих кодах розряди змінюються суттєво частіше.

Стиснення зображень. Коди Грея

Незважаючи на те, що обидві сім'ї утворюють коди Грея, в них по-різному чергуються біти 0 і 1 у різних побитих шарах. На рис. (b) найважливіший біт змінюється 4 рази. Другий значний біт змінюється 8 разів, а біти трьох молодших розрядів змінюються, відповідно, 16, 2 і один раз. Якщо використовувати цю множину кодів Грея для розшарування зображення, то середні шари, очевидно, утворюють найменшу кореляцію між сусідніми пікселями.

На противагу цьому коди рис. (c) демонструватимуть велику кореляцію в шарах старших розрядів, оскільки там зміна між 0 і 1 відбувається, відповідно, 1, 2 і 4 рази (за старшинством розрядів). А молодші розряди покажуть слабку кореляцію пікселів, яка наближається до нуля, як і при випадковому розподілі пікселів. Для порівняння на рис. (a) зображено звичайні двійкові коди. Видно, що в цих кодах розряди змінюються суттєво частіше.

Стиснення зображень. Коди Грея

Незважаючи на те, що обидві сім'ї утворюють коди Грея, в них по-різному чергуються біти 0 і 1 у різних побитих шарах. На рис. (b) найважливіший біт змінюється 4 рази. Другий значний біт змінюється 8 разів, а біти трьох молодших розрядів змінюються, відповідно, 16, 2 і один раз. Якщо використовувати цю множину кодів Грея для розшарування зображення, то середні шари, очевидно, утворюють найменшу кореляцію між сусідніми пікселями.

На противагу цьому коди рис. (c) демонструватимуть велику кореляцію в шарах старших розрядів, оскільки там зміна між 0 і 1 відбувається, відповідно, 1, 2 і 4 рази (за старшинством розрядів). А молодші розряди покажуть слабку кореляцію пікселів, яка наближається до нуля, як і при випадковому розподілі пікселів. Для порівняння на рис. (a) зображено звичайні двійкові коди. Видно, що в цих кодах розряди змінюються суттєво частіше.

Стиснення зображень. Коди Грея

Незважаючи на те, що обидві сім'ї утворюють коди Грея, в них по-різному чергуються біти 0 і 1 у різних побитих шарах. На рис. (b) найважливіший біт змінюється 4 рази. Другий значний біт змінюється 8 разів, а біти трьох молодших розрядів змінюються, відповідно, 16, 2 і один раз. Якщо використовувати цю множину кодів Грея для розшарування зображення, то середні шари, очевидно, утворюють найменшу кореляцію між сусідніми пікселями.

На противагу цьому коди рис. (c) демонструватимуть велику кореляцію в шарах старших розрядів, оскільки там зміна між 0 і 1 відбувається, відповідно, 1, 2 і 4 рази (за старшинством розрядів). А молодші розряди покажуть слабку кореляцію пікселів, яка наближається до нуля, як і при випадковому розподілі пікселів. Для порівняння на рис. (a) зображено звичайні двійкові коди. Видно, що в цих кодах розряди змінюються суттєво частіше.

Стиснення зображень. Коди Грея

Незважаючи на те, що обидві сім'ї утворюють коди Грея, в них по-різному чергуються біти 0 і 1 у різних побитих шарах. На рис. (b) найважливіший біт змінюється 4 рази. Другий значний біт змінюється 8 разів, а біти трьох молодших розрядів змінюються, відповідно, 16, 2 і один раз. Якщо використовувати цю множину кодів Грея для розшарування зображення, то середні шари, очевидно, утворюють найменшу кореляцію між сусідніми пікселями.

На противагу цьому коди рис. (c) демонструватимуть велику кореляцію в шарах старших розрядів, оскільки там зміна між 0 і 1 відбувається, відповідно, 1, 2 і 4 рази (за старшинством розрядів). А молодші розряди покажуть слабку кореляцію пікселів, яка наближається до нуля, як і при випадковому розподілі пікселів. Для порівняння на рис. (a) зображено звичайні двійкові коди. Видно, що в цих кодах розряди змінюються суттєво частіше.

Стиснення зображень. Коди Грея

Незважаючи на те, що обидві сім'ї утворюють коди Грея, в них по-різному чергуються біти 0 і 1 у різних побитих шарах. На рис. (b) найважливіший біт змінюється 4 рази. Другий значний біт змінюється 8 разів, а біти трьох молодших розрядів змінюються, відповідно, 16, 2 і один раз. Якщо використовувати цю множину кодів Грея для розшарування зображення, то середні шари, очевидно, утворюють найменшу кореляцію між сусідніми пікселями.

На противагу цьому коди рис. (c) демонструватимуть велику кореляцію в шарах старших розрядів, оскільки там зміна між 0 і 1 відбувається, відповідно, 1, 2 і 4 рази (за старшинством розрядів). А молодші розряди покажуть слабку кореляцію пікселів, яка наближається до нуля, як і при випадковому розподілі пікселів. Для порівняння на рис. (a) зображено звичайні двійкові коди. Видно, що в цих кодах розряди змінюються суттєво частіше.

Стиснення зображень. Коди Грея

Незважаючи на те, що обидві сім'ї утворюють коди Грея, в них по-різному чергуються біти 0 і 1 у різних побитих шарах. На рис. (b) найважливіший біт змінюється 4 рази. Другий значний біт змінюється 8 разів, а біти трьох молодших розрядів змінюються, відповідно, 16, 2 і один раз. Якщо використовувати цю множину кодів Грея для розшарування зображення, то середні шари, очевидно, утворюють найменшу кореляцію між сусідніми пікселями.

На противагу цьому коди рис. (c) демонструватимуть велику кореляцію в шарах старших розрядів, оскільки там зміна між 0 і 1 відбувається, відповідно, 1, 2 і 4 рази (за старшинством розрядів). А молодші розряди покажуть слабку кореляцію пікселів, яка наближається до нуля, як і при випадковому розподілі пікселів. Для порівняння на рис. (a) зображено звичайні двійкові коди. Видно, що в цих кодах розряди змінюються суттєво частіше.

Стиснення зображень. Коди Грея

Незважаючи на те, що обидві сім'ї утворюють коди Грея, в них по-різному чергуються біти 0 і 1 у різних побитих шарах. На рис. (b) найважливіший біт змінюється 4 рази. Другий значний біт змінюється 8 разів, а біти трьох молодших розрядів змінюються, відповідно, 16, 2 і один раз. Якщо використовувати цю множину кодів Грея для розшарування зображення, то середні шари, очевидно, утворюють найменшу кореляцію між сусідніми пікселями.

На противагу цьому коди рис. (c) демонструватимуть велику кореляцію в шарах старших розрядів, оскільки там зміна між 0 і 1 відбувається, відповідно, 1, 2 і 4 рази (за старшинством розрядів). А молодші розряди покажуть слабку кореляцію пікселів, яка наближається до нуля, як і при випадковому розподілі пікселів. Для порівняння на рис. (a) зображено звичайні двійкові коди. Видно, що в цих кодах розряди змінюються суттєво частіше.

Незважаючи на те, що обидві сім'ї утворюють коди Грея, в них по-різному чергуються біти 0 і 1 у різних побитих шарах. На рис. (b) найважливіший біт змінюється 4 рази. Другий значний біт змінюється 8 разів, а біти трьох молодших розрядів змінюються, відповідно, 16, 2 і один раз. Якщо використовувати цю множину кодів Грея для розшарування зображення, то середні шари, очевидно, утворюють найменшу кореляцію між сусідніми пікселями.

На противагу цьому коди рис. (c) демонструватимуть велику кореляцію в шарах старших розрядів, оскільки там зміна між 0 і 1 відбувається, відповідно, 1, 2 і 4 рази (за старшинством розрядів). А молодші розряди покажуть слабку кореляцію пікселів, яка наближається до нуля, як і при випадковому розподілі пікселів. Для порівняння на рис. (a) зображено звичайні двійкові коди. Видно, що в цих кодах розряди змінюються суттєво частіше.

Стиснення зображень. Коди Грея

Незважаючи на те, що обидві сім'ї утворюють коди Грея, в них по-різному чергуються біти 0 і 1 у різних побитих шарах. На рис. (b) найважливіший біт змінюється 4 рази. Другий значний біт змінюється 8 разів, а біти трьох молодших розрядів змінюються, відповідно, 16, 2 і один раз. Якщо використовувати цю множину кодів Грея для розшарування зображення, то середні шари, очевидно, утворюють найменшу кореляцію між сусідніми пікселями.

На противагу цьому коди рис. (c) демонструватимуть велику кореляцію в шарах старших розрядів, оскільки там зміна між 0 і 1 відбувається, відповідно, 1, 2 і 4 рази (за старшинством розрядів). А молодші розряди покажуть слабку кореляцію пікселів, яка наближається до нуля, як і при випадковому розподілі пікселів. Для порівняння на рис. (a) зображено звичайні двійкові коди. Видно, що в цих кодах розряди змінюються суттєво частіше.

Стиснення зображень. Коди Грея

Незважаючи на те, що обидві сім'ї утворюють коди Грея, в них по-різному чергуються біти 0 і 1 у різних побитих шарах. На рис. (b) найважливіший біт змінюється 4 рази. Другий значний біт змінюється 8 разів, а біти трьох молодших розрядів змінюються, відповідно, 16, 2 і один раз. Якщо використовувати цю множину кодів Грея для розшарування зображення, то середні шари, очевидно, утворюють найменшу кореляцію між сусідніми пікселями.

На противагу цьому коди рис. (c) демонструватимуть велику кореляцію в шарах старших розрядів, оскільки там зміна між 0 і 1 відбувається, відповідно, 1, 2 і 4 рази (за старшинством розрядів). А молодші розряди покажуть слабку кореляцію пікселів, яка наближається до нуля, як і при випадковому розподілі пікселів. Для порівняння на рис. (a) зображено звичайні двійкові коди. Видно, що в цих кодах розряди змінюються суттєво частіше.

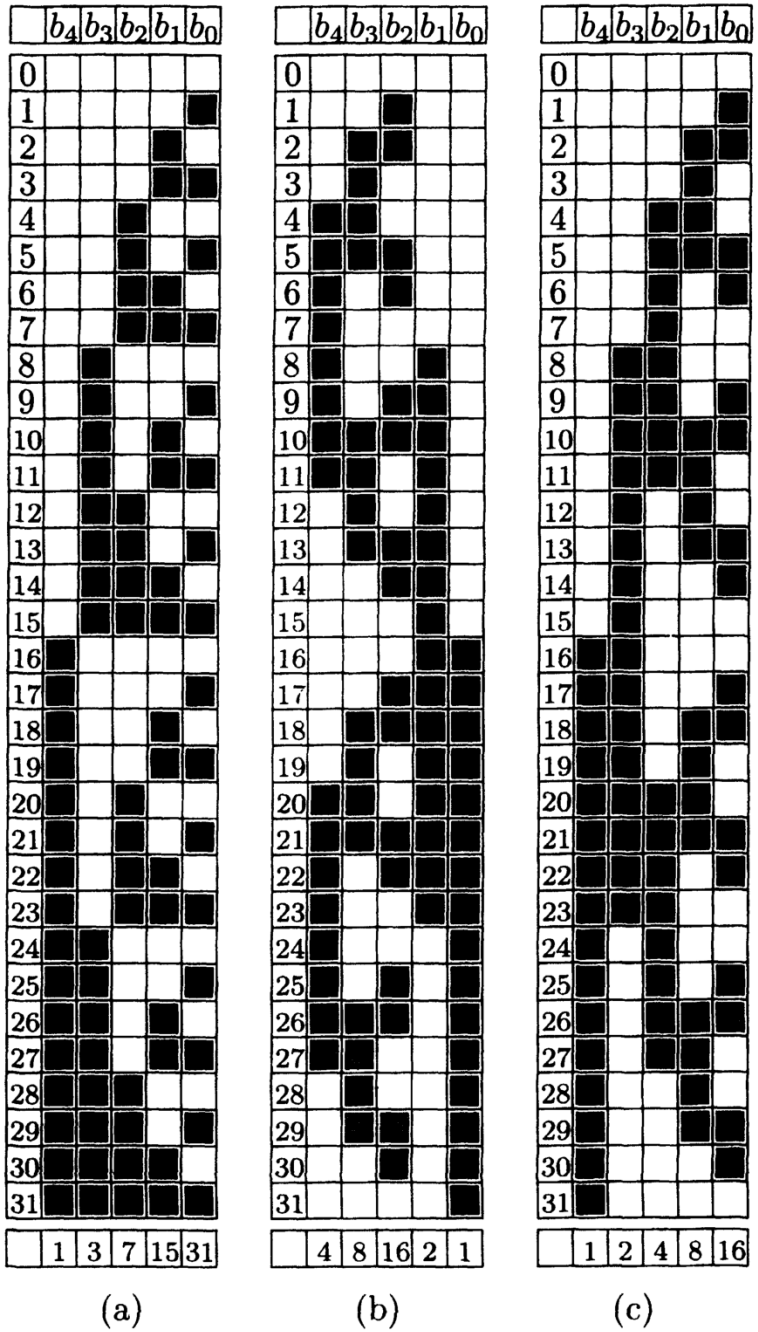
Незважаючи на те, що обидві сім'ї утворюють коди Грея, в них по-різному чергуються біти 0 і 1 у різних побитих шарах. На рис. (b) найважливіший біт змінюється 4 рази. Другий значний біт змінюється 8 разів, а біти трьох молодших розрядів змінюються, відповідно, 16, 2 і один раз. Якщо використовувати цю множину кодів Грея для розшарування зображення, то середні шари, очевидно, утворюють найменшу кореляцію між сусідніми пікселями.

На противагу цьому коди рис. (c) демонструватимуть велику кореляцію в шарах старших розрядів, оскільки там зміна між 0 і 1 відбувається, відповідно, 1, 2 і 4 рази (за старшинством розрядів). А молодші розряди покажуть слабку кореляцію пікселів, яка наближається до нуля, як і при випадковому розподілі пікселів. Для порівняння на рис. (a) зображено звичайні двійкові коди. Видно, що в цих кодах розряди змінюються суттєво частіше.

Незважаючи на те, що обидві сім'ї утворюють коди Грея, в них по-різному чергуються біти 0 і 1 у різних побитих шарах. На рис. (b) найважливіший біт змінюється 4 рази. Другий значний біт змінюється 8 разів, а біти трьох молодших розрядів змінюються, відповідно, 16, 2 і один раз. Якщо використовувати цю множину кодів Грея для розшарування зображення, то середні шари, очевидно, утворюють найменшу кореляцію між сусідніми пікселями.

На противагу цьому коди рис. (c) демонструватимуть велику кореляцію в шарах старших розрядів, оскільки там зміна між 0 і 1 відбувається, відповідно, 1, 2 і 4 рази (за старшинством розрядів). А молодші розряди покажуть слабку кореляцію пікселів, яка наближається до нуля, як і при випадковому розподілі пікселів. Для порівняння на рис. (a) зображено звичайні двійкові коди. Видно, що в цих кодах розряди змінюються суттєво частіше.

Стиснення зображень. Коди Грея



Стиснення зображень. Коди Грея

Навіть поверховий погляд на коди Грея рис. (c) виявляє в них певну регулярність. Більш детальне дослідження виявляє дві важливі особливості цих кодів, які можна використовувати при їхній побудові. Перша особливість полягає у періодичній зміні значень усіх п'яти розрядів. Легко написати програму, яка спочатку встановить усі п'ять розрядів в 0, а потім мінятиме найправіший розряд у кожного другого коду, а решта чотирьох розрядів мінятиме в середині періоду свого правого сусіднього розряду.

Інша особливість кодів Грея полягає в тому, що друга половина таблиці є дзеркальним відображенням першої половини, якщо значний біт встановити в 1. Якщо перша половина таблиці вже обчислена, то друга легко виходить за допомогою цієї властивості.

Стиснення зображень. Коди Грея

Навіть поверховий погляд на коди Грея рис. (c) виявляє в них певну регулярність. Більш детальне дослідження виявляє дві важливі особливості цих кодів, які можна використовувати при їхній побудові. Перша особливість полягає у періодичній зміні значень усіх п'яти розрядів. Легко написати програму, яка спочатку встановить усі п'ять розрядів в 0, а потім мінятиме найправіший розряд у кожного другого коду, а решта чотирьох розрядів мінятиме в середині періоду свого правого сусіднього розряду.

Інша особливість кодів Грея полягає в тому, що друга половина таблиці є дзеркальним відображенням першої половини, якщо значний біт встановити в 1. Якщо перша половина таблиці вже обчислена, то друга легко виходить за допомогою цієї властивості.

Стиснення зображень. Коди Грея

Навіть поверховий погляд на коди Грея рис. (c) виявляє в них певну регулярність. Більш детальне дослідження виявляє дві важливі особливості цих кодів, які можна використовувати при їхній побудові. Перша особливість полягає у періодичній зміні значень усіх п'яти розрядів. Легко написати програму, яка спочатку встановить усі п'ять розрядів в 0, а потім мінятиме найправіший розряд у кожного другого коду, а решта чотирьох розрядів мінятиме в середині періоду свого правого сусіднього розряду.

Інша особливість кодів Грея полягає в тому, що друга половина таблиці є дзеркальним відображенням першої половини, якщо значний біт встановити в 1. Якщо перша половина таблиці вже обчислена, то друга легко виходить за допомогою цієї властивості.

Стиснення зображень. Коди Грея

Навіть поверховий погляд на коди Грея рис. (c) виявляє в них певну регулярність. Більш детальне дослідження виявляє дві важливі особливості цих кодів, які можна використовувати при їхній побудові.

Перша особливість полягає у періодичній зміні значень усіх п'яти розрядів. Легко написати програму, яка спочатку встановить усі п'ять розрядів в 0, а потім мінятиме найправіший розряд у кожного другого коду, а решта чотирьох розрядів мінятиме в середині періоду свого правого сусіднього розряду.

Інша особливість кодів Грея полягає в тому, що друга половина таблиці є дзеркальним відображенням першої половини, якщо значний біт встановити в 1. Якщо перша половина таблиці вже обчислена, то друга легко виходить за допомогою цієї властивості.

Стиснення зображень. Коди Грея

Навіть поверховий погляд на коди Грея рис. (c) виявляє в них певну регулярність. Більш детальне дослідження виявляє дві важливі особливості цих кодів, які можна використовувати при їхній побудові. Перша особливість полягає у періодичній зміні значень усіх п'яти розрядів. Легко написати програму, яка спочатку встановить усі п'ять розрядів в 0, а потім мінятиме найправіший розряд у кожного другого коду, а решта чотирьох розрядів мінятиме в середині періоду свого правого сусіднього розряду.

Інша особливість кодів Грея полягає в тому, що друга половина таблиці є дзеркальним відображенням першої половини, якщо значний біт встановити в 1. Якщо перша половина таблиці вже обчислена, то друга легко виходить за допомогою цієї властивості.

Стиснення зображень. Коди Грея

Навіть поверховий погляд на коди Грея рис. (c) виявляє в них певну регулярність. Більш детальне дослідження виявляє дві важливі особливості цих кодів, які можна використовувати при їхній побудові. Перша особливість полягає у періодичній зміні значень усіх п'яти розрядів. Легко написати програму, яка спочатку встановить усі п'ять розрядів в 0, а потім мінятиме найправіший розряд у кожного другого коду, а решта чотирьох розрядів мінятиме в середині періоду свого правого сусіднього розряду.

Інша особливість кодів Грея полягає в тому, що друга половина таблиці є дзеркальним відображенням першої половини, якщо значний біт встановити в 1. Якщо перша половина таблиці вже обчислена, то друга легко виходить за допомогою цієї властивості.

Стиснення зображень. Коди Грея

Навіть поверховий погляд на коди Грея рис. (c) виявляє в них певну регулярність. Більш детальне дослідження виявляє дві важливі особливості цих кодів, які можна використовувати при їхній побудові. Перша особливість полягає у періодичній зміні значень усіх п'яти розрядів. Легко написати програму, яка спочатку встановить усі п'ять розрядів в 0, а потім мінятиме найправіший розряд у кожного другого коду, а решта чотирьох розрядів мінятиме в середині періоду свого правого сусіднього розряду.

Інша особливість кодів Грея полягає в тому, що друга половина таблиці є дзеркальним відображенням першої половини, якщо значний біт встановити в 1. Якщо перша половина таблиці вже обчислена, то друга легко виходить за допомогою цієї властивості.

Стиснення зображень. Коди Грея

Навіть поверховий погляд на коди Грея рис. (c) виявляє в них певну регулярність. Більш детальне дослідження виявляє дві важливі особливості цих кодів, які можна використовувати при їхній побудові. Перша особливість полягає у періодичній зміні значень усіх п'яти розрядів. Легко написати програму, яка спочатку встановить усі п'ять розрядів в 0, а потім мінятиме найправіший розряд у кожного другого коду, а решта чотирьох розрядів мінятиме в середині періоду свого правого сусіднього розряду.

Інша особливість кодів Грея полягає в тому, що друга половина таблиці є дзеркальним відображенням першої половини, якщо значний біт встановити в 1. Якщо перша половина таблиці вже обчислена, то друга легко виходить за допомогою цієї властивості.

Стиснення зображень. Коди Грея

Навіть поверховий погляд на коди Грея рис. (c) виявляє в них певну регулярність. Більш детальне дослідження виявляє дві важливі особливості цих кодів, які можна використовувати при їхній побудові. Перша особливість полягає у періодичній зміні значень усіх п'яти розрядів. Легко написати програму, яка спочатку встановить усі п'ять розрядів в 0, а потім мінятиме найправіший розряд у кожного другого коду, а решта чотирьох розрядів мінятиме в середині періоду свого правого сусіднього розряду.

Інша особливість кодів Грея полягає в тому, що друга половина таблиці є дзеркальним відображенням першої половини, якщо значний біт встановити в 1. Якщо перша половина таблиці вже обчислена, то друга легко виходить за допомогою цієї властивості.

Стиснення зображень. Коди Грея

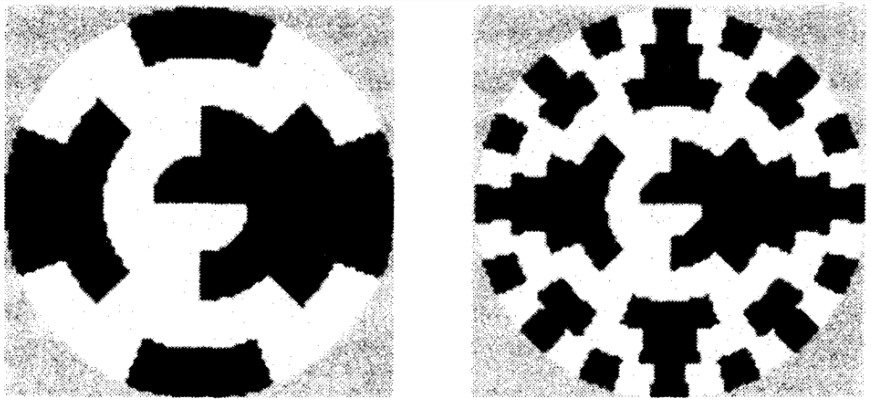
Навіть поверховий погляд на коди Грея рис. (c) виявляє в них певну регулярність. Більш детальне дослідження виявляє дві важливі особливості цих кодів, які можна використовувати при їхній побудові. Перша особливість полягає у періодичній зміні значень усіх п'яти розрядів. Легко написати програму, яка спочатку встановить усі п'ять розрядів в 0, а потім мінятиме найправіший розряд у кожного другого коду, а решта чотирьох розрядів мінятиме в середині періоду свого правого сусіднього розряду.

Інша особливість кодів Грея полягає в тому, що друга половина таблиці є дзеркальним відображенням першої половини, якщо значний біт встановити в 1. Якщо перша половина таблиці вже обчислена, то друга легко виходить за допомогою цієї властивості.

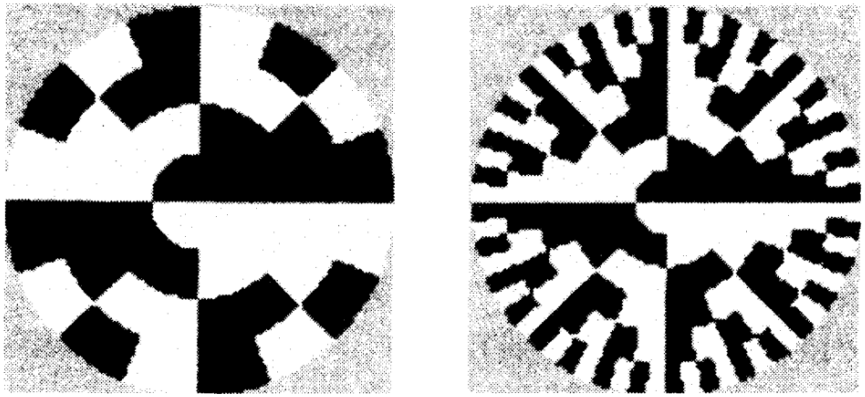
Навіть поверховий погляд на коди Грея рис. (c) виявляє в них певну регулярність. Більш детальне дослідження виявляє дві важливі особливості цих кодів, які можна використовувати при їхній побудові. Перша особливість полягає у періодичній зміні значень усіх п'яти розрядів. Легко написати програму, яка спочатку встановить усі п'ять розрядів в 0, а потім мінятиме найправіший розряд у кожного другого коду, а решта чотирьох розрядів мінятиме в середині періоду свого правого сусіднього розряду.

Інша особливість кодів Грея полягає в тому, що друга половина таблиці є дзеркальним відображенням першої половини, якщо значний біт встановити в 1. Якщо перша половина таблиці вже обчислена, то друга легко виходить за допомогою цієї властивості.

Стиснення зображень. Коди Грея



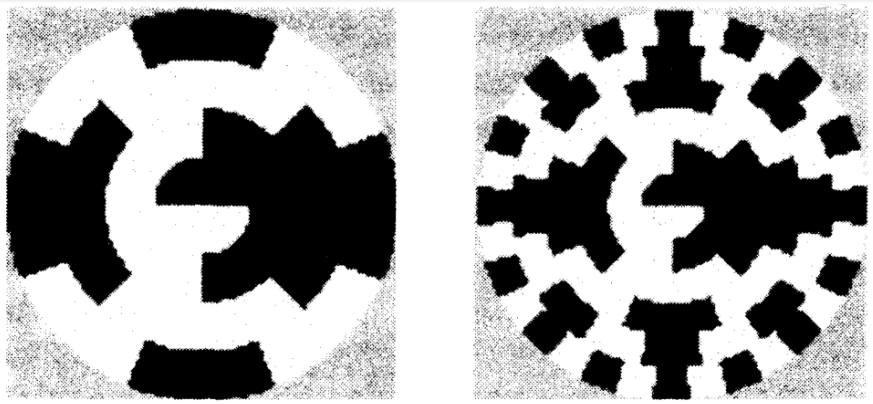
(a)



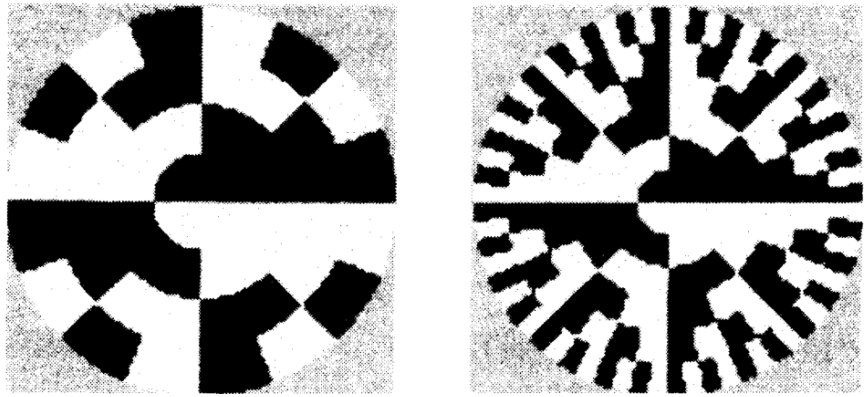
(b)

На рис. зображено кругові діаграми, що зображають 4-х та 6-ти бітні коди RGC (a), та звичайні двійкові коди (b). Шари розрядів показані як кілець, причому шар найзначнішого біта перебуває всередині. Видно, що максимальна кутова частота коду RGC вдвічі менша, ніж у двійкового коду. Таке циклічне подання кодів Грея не порушує їхню структуру, оскільки перший і останній код також відрізняються рівно в одному біті.

Стиснення зображень. Коди Грея



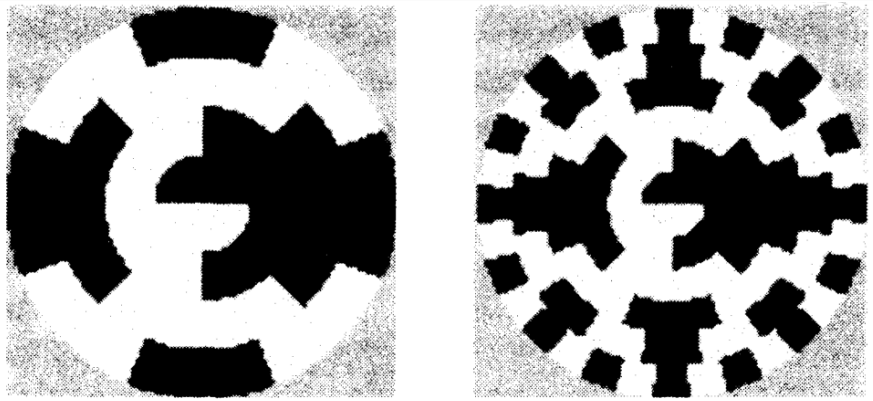
(a)



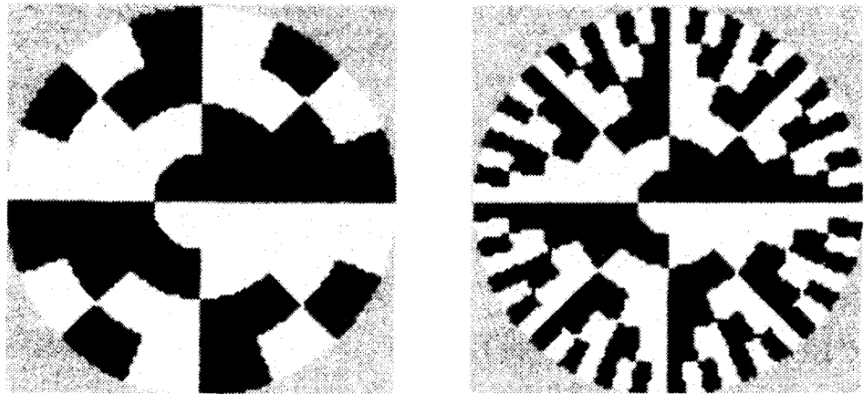
(b)

На рис. зображено кругові діаграми, що зображають 4-х та 6-ти бітні коди RGC (a), та звичайні двійкові коди (b). Шари розрядів показані як кілець, причому шар найзначнішого біта перебуває всередині. Видно, що максимальна кутова частота коду RGC вдвічі менша, ніж у двійкового коду. Таке циклічне подання кодів Грея не порушує їхню структуру, оскільки перший і останній код також відрізняються рівно в одному біті.

Стиснення зображень. Коди Грея



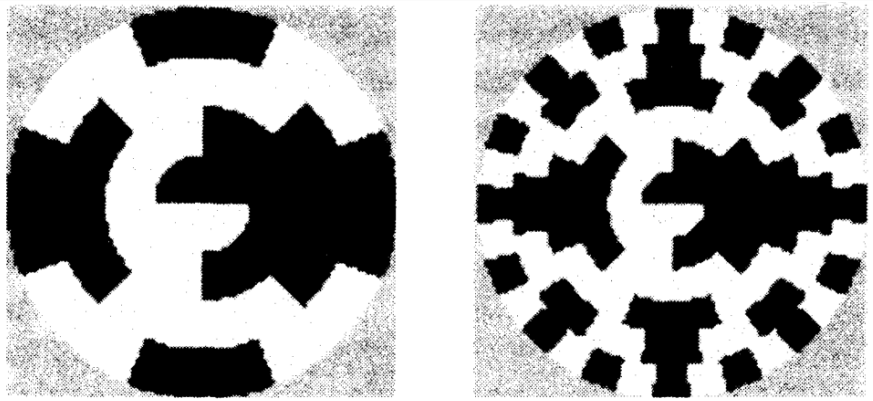
(a)



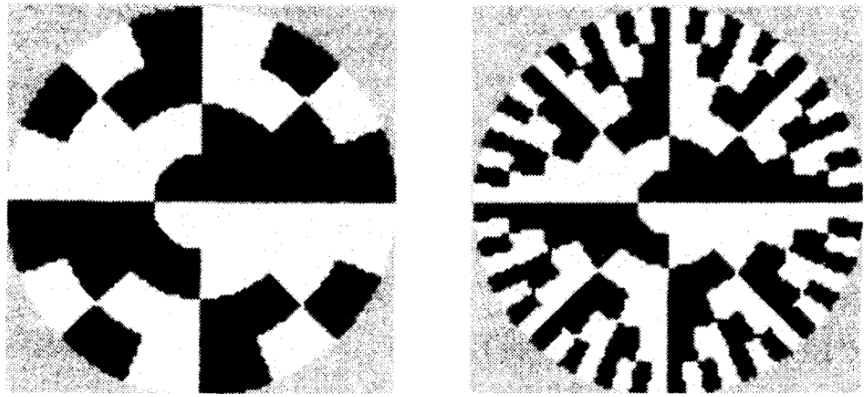
(b)

На рис. зображено кругові діаграми, що зображають 4-х та 6-ти бітні коди RGC (a), та звичайні двійкові коди (b). Шари розрядів показані як кілець, причому шар найзначнішого біта перебуває всередині. Видно, що максимальна кутова частота коду RGC вдвічі менша, ніж у двійкового коду. Таке циклічне подання кодів Грея не порушує їхню структуру, оскільки перший і останній код також відрізняються рівно в одному біті.

Стиснення зображень. Коди Грея



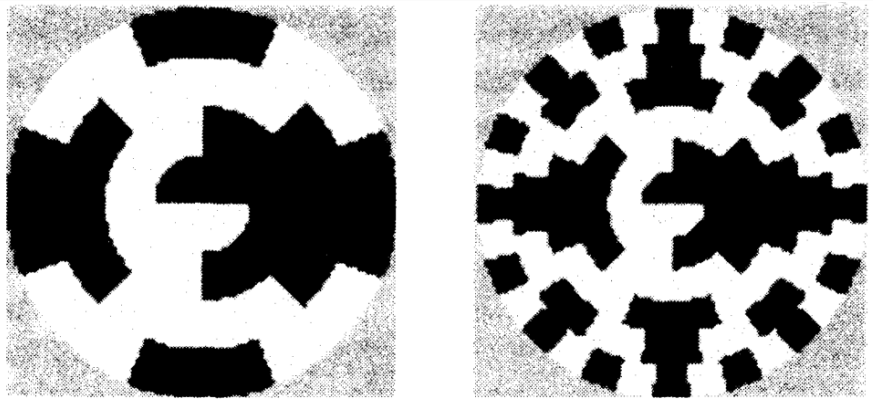
(a)



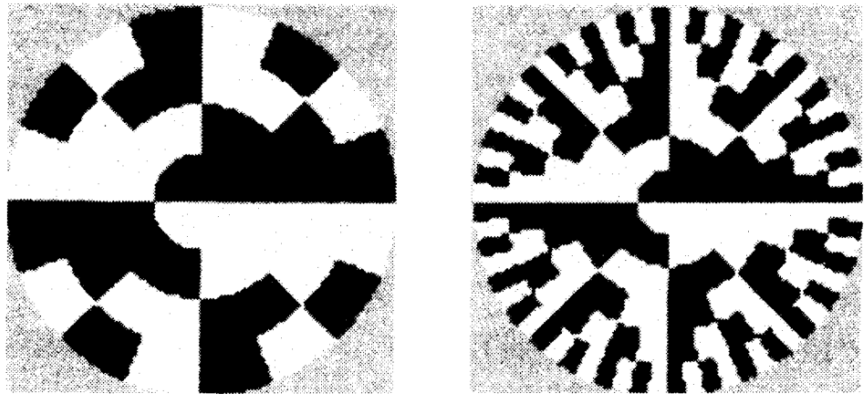
(b)

На рис. зображено кругові діаграми, що зображають 4-х та 6-ти бітні коди RGC (a), та звичайні двійкові коди (b). Шари розрядів показані як кілець, причому шар найзначнішого біта перебуває всередині. Видно, що максимальна кутова частота коду RGC вдвічі менша, ніж у двійкового коду. Таке циклічне подання кодів Грея не порушує їхню структуру, оскільки перший і останній код також відрізняються рівно в одному біті.

Стиснення зображень. Коди Грея



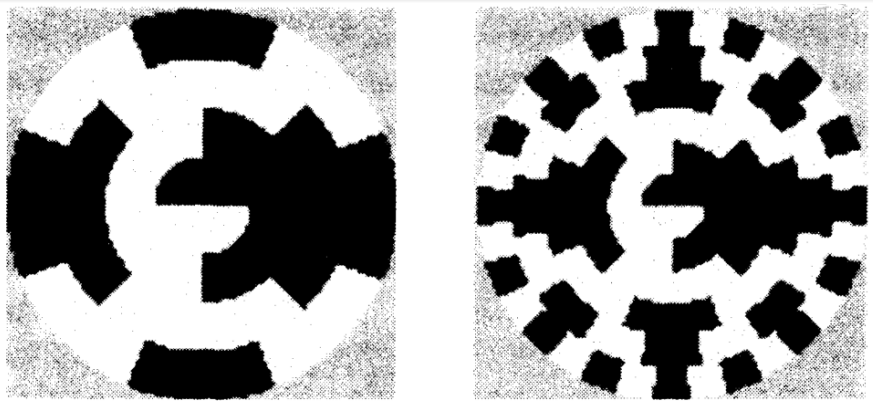
(a)



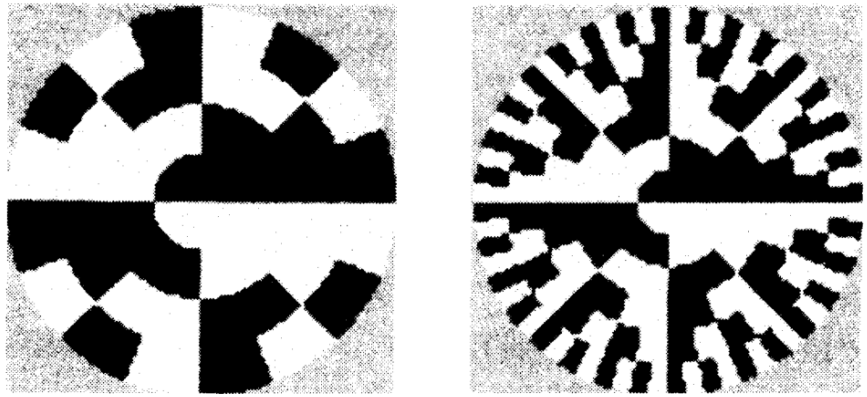
(b)

На рис. зображено кругові діаграми, що зображають 4-х та 6-ти бітні коди RGC (a), та звичайні двійкові коди (b). Шари розрядів показані як кілець, причому шар найзначнішого біта перебуває всередині. Видно, що максимальна кутова частота коду RGC вдвічі менша, ніж у двійкового коду. Таке циклічне подання кодів Грея не порушує їхню структуру, оскільки перший і останній код також відрізняються рівно в одному біті.

Стиснення зображень. Коди Грея



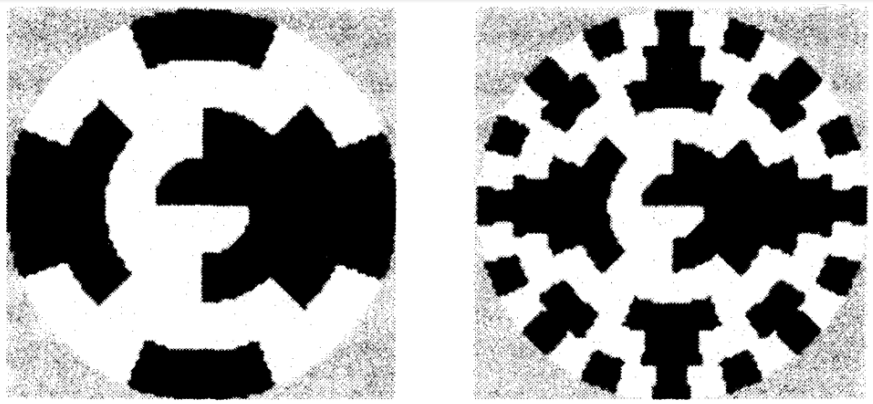
(a)



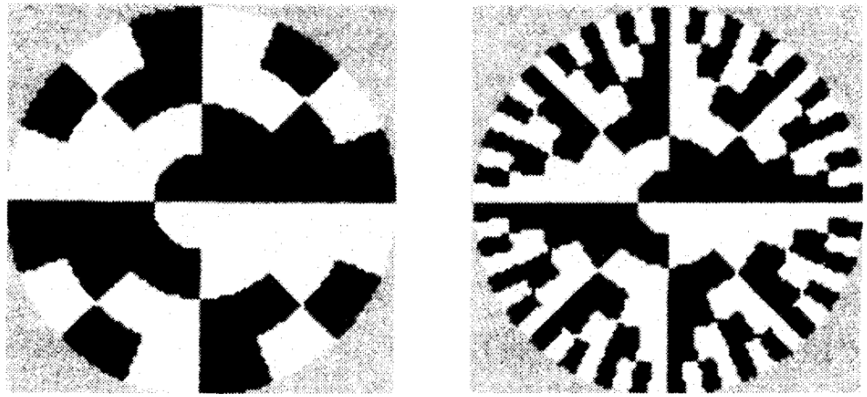
(b)

На рис. зображено кругові діаграми, що зображають 4-х та 6-ти бітні коди RGC (a), та звичайні двійкові коди (b). Шари розрядів показані як кілець, причому шар найзначнішого біта перебуває всередині. Видно, що максимальна кутова частота коду RGC вдвічі менша, ніж у двійкового коду. Таке циклічне подання кодів Грея не порушує їхню структуру, оскільки перший і останній код також відрізняються рівно в одному біті.

Стиснення зображень. Коди Грея



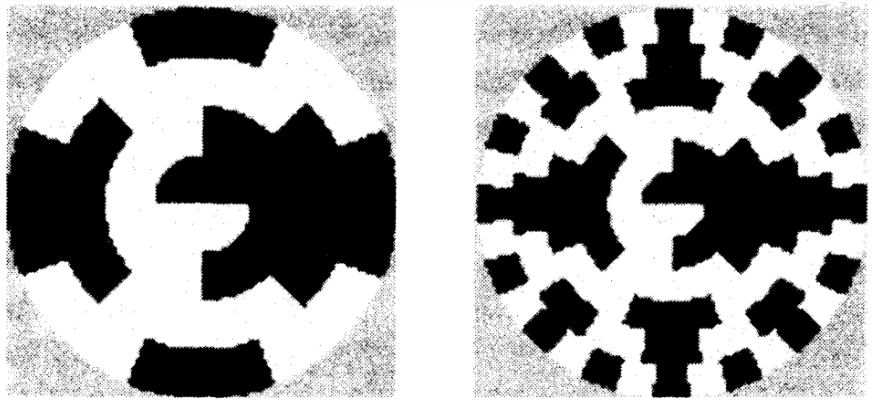
(a)



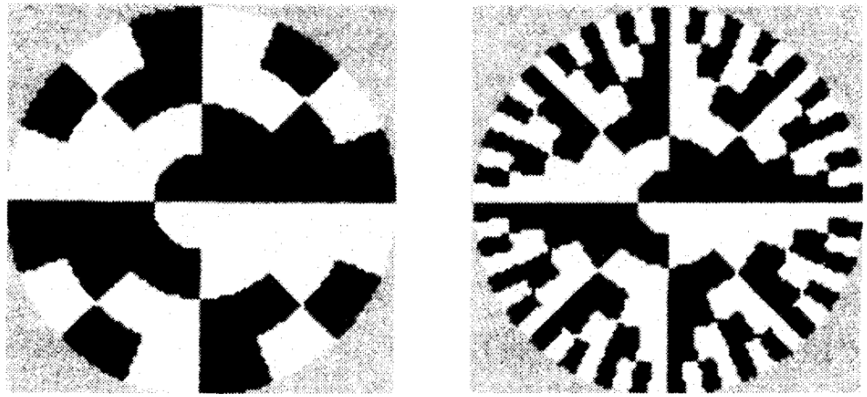
(b)

На рис. зображено кругові діаграми, що зображають 4-х та 6-ти бітні коди RGC (a), та звичайні двійкові коди (b). Шари розрядів показані як кілець, причому шар найзначнішого біта перебуває всередині. Видно, що максимальна кутова частота коду RGC вдвічі менша, ніж у двійкового коду. Таке циклічне подання кодів Грея не порушує їхню структуру, оскільки перший і останній код також відрізняються рівно в одному біті.

Стиснення зображень. Коди Грея



(a)



(b)

На рис. зображено кругові діаграми, що зображають 4-х та 6-ти бітні коди RGC (a), та звичайні двійкові коди (b). Шари розрядів показані як кілець, причому шар найзначнішого біта перебуває всередині. Видно, що максимальна кутова частота коду RGC вдвічі менша, ніж у двійкового коду. Таке циклічне подання кодів Грея не порушує їхню структуру, оскільки перший і останній код також відрізняються рівно в одному біті.

Стиснення зображень. Коди Грея

До кольорових зображень можна застосовувати методи компресії, розроблені для інших типів зображень. Будь-який метод стиснення напівтонових образів дозволяє стискати кольорові зображення. У кольоровому зображенні кожен піксель складається з трьох кольорових компонентів (наприклад, RGB). Уявімо кольорове зображення, в якому всі компоненти складаються з одного байта. Піксель описується трьома байтами чи 24 бітами, але ці біти не можна розглядати як одне число. Три пікселі

$$118|206|12 \qquad \text{та} \qquad 117|206|12$$

розрізняються тільки на одиницю в першій компоненті, а тому вони мають близькі кольори. Якщо ж їх розглядати у вигляді 24-бітних чисел, то вони сильно відрізнятимуться, оскільки вони відрізняються у старшому біті. Будь-який метод компресії, який ґрунтуватиметься на такому розрізненні пікселів, буде дуже неоптимальним. У цьому сенсі найбільш підходящим є метод поділу зображення на кольорові компоненти з їхнім подальшим незалежним стиском методом напівтонової компресії.

Метод зважених контекстних дерев зображень є прикладом використання кодів RGC для стиснення образів.

Стиснення зображень. Коди Грея

До кольорових зображень можна застосовувати методи компресії, розроблені для інших типів зображень. Будь-який метод стиснення напівтонових образів дозволяє стискати кольорові зображення. У кольоровому зображенні кожен піксель складається з трьох кольорових компонентів (наприклад, RGB). Уявімо кольорове зображення, в якому всі компоненти складаються з одного байта. Піксель описується трьома байтами чи 24 бітами, але ці біти не можна розглядати як одне число. Три пікселі

$$118|206|12 \qquad \text{та} \qquad 117|206|12$$

розрізняються тільки на одиницю в першій компоненті, а тому вони мають близькі кольори. Якщо ж їх розглядати у вигляді 24-бітних чисел, то вони сильно відрізнятимуться, оскільки вони відрізняються у старшому біті. Будь-який метод компресії, який ґрунтуватиметься на такому розрізненні пікселів, буде дуже неоптимальним. У цьому сенсі найбільш підходящим є метод поділу зображення на кольорові компоненти з їхнім подальшим незалежним стиском методом напівтонової компресії.

Метод зважених контекстних дерев зображень є прикладом використання кодів RGC для стиснення образів.

Стиснення зображень. Коди Грея

До кольорових зображень можна застосовувати методи компресії, розроблені для інших типів зображень. Будь-який метод стиснення напівтонових образів дозволяє стискати кольорові зображення. У кольоровому зображенні кожен піксель складається з трьох кольорових компонентів (наприклад, RGB). Уявімо кольорове зображення, в якому всі компоненти складаються з одного байта. Піксель описується трьома байтами чи 24 бітами, але ці біти не можна розглядати як одне число. Три пікселі

$$118|206|12 \qquad \text{та} \qquad 117|206|12$$

розрізняються тільки на одиницю в першій компоненті, а тому вони мають близькі кольори. Якщо ж їх розглядати у вигляді 24-бітних чисел, то вони сильно відрізнятимуться, оскільки вони відрізняються у старшому біті. Будь-який метод компресії, який ґрунтуватиметься на такому розрізненні пікселів, буде дуже неоптимальним. У цьому сенсі найбільш підходящим є метод поділу зображення на кольорові компоненти з їхнім подальшим незалежним стиском методом напівтонової компресії.

Метод зважених контекстних дерев зображень є прикладом використання кодів RGC для стиснення образів.

Стиснення зображень. Коди Грея

До кольорових зображень можна застосовувати методи компресії, розроблені для інших типів зображень. Будь-який метод стиснення напівтонових образів дозволяє стискати кольорові зображення. У кольоровому зображенні кожен піксель складається з трьох кольорових компонентів (наприклад, RGB). Уявімо кольорове зображення, в якому всі компоненти складаються з одного байта. Піксель описується трьома байтами чи 24 бітами, але ці біти не можна розглядати як одне число. Три пікселі

$$118|206|12 \qquad \text{та} \qquad 117|206|12$$

розрізняються тільки на одиницю в першій компоненті, а тому вони мають близькі кольори. Якщо ж їх розглядати у вигляді 24-бітних чисел, то вони сильно відрізнятимуться, оскільки вони відрізняються у старшому біті. Будь-який метод компресії, який ґрунтуватиметься на такому розрізненні пікселів, буде дуже неоптимальним. У цьому сенсі найбільш підходящим є метод поділу зображення на кольорові компоненти з їхнім подальшим незалежним стиском методом напівтонової компресії.

Метод зважених контекстних дерев зображень є прикладом використання кодів RGC для стиснення образів.

Стиснення зображень. Коди Грея

До кольорових зображень можна застосовувати методи компресії, розроблені для інших типів зображень. Будь-який метод стиснення напівтонових образів дозволяє стискати кольорові зображення. У кольоровому зображенні кожен піксель складається з трьох кольорових компонентів (наприклад, RGB). Уявімо кольорове зображення, в якому всі компоненти складаються з одного байта. Піксель описується трьома байтами чи 24 бітами, але ці біти не можна розглядати як одне число. Три пікселі

$$118|206|12 \qquad \text{та} \qquad 117|206|12$$

розрізняються тільки на одиницю в першій компоненті, а тому вони мають близькі кольори. Якщо ж їх розглядати у вигляді 24-бітних чисел, то вони сильно відрізнятимуться, оскільки вони відрізняються у старшому біті. Будь-який метод компресії, який ґрунтуватиметься на такому розрізненні пікселів, буде дуже неоптимальним. У цьому сенсі найбільш підходящим є метод поділу зображення на кольорові компоненти з їхнім подальшим незалежним стиском методом напівтонової компресії.

Метод зважених контекстних дерев зображень є прикладом використання кодів RGC для стиснення образів.

Стиснення зображень. Коди Грея

До кольорових зображень можна застосовувати методи компресії, розроблені для інших типів зображень. Будь-який метод стиснення напівтонових образів дозволяє стискати кольорові зображення. У кольоровому зображенні кожен піксель складається з трьох кольорових компонентів (наприклад, RGB). Уявімо кольорове зображення, в якому всі компоненти складаються з одного байта. Піксель описується трьома байтами чи 24 бітами, але ці біти не можна розглядати як одне число. Три пікселі

$$118|206|12 \qquad \text{та} \qquad 117|206|12$$

розрізняються тільки на одиницю в першій компоненті, а тому вони мають близькі кольори. Якщо ж їх розглядати у вигляді 24-бітних чисел, то вони сильно відрізнятимуться, оскільки вони відрізняються у старшому біті. Будь-який метод компресії, який ґрунтуватиметься на такому розрізненні пікселів, буде дуже неоптимальним. У цьому сенсі найбільш підходящим є метод поділу зображення на кольорові компоненти з їхнім подальшим незалежним стиском методом напівтонової компресії.

Метод зважених контекстних дерев зображень є прикладом використання кодів RGC для стиснення образів.

Стиснення зображень. Коди Грея

До кольорових зображень можна застосовувати методи компресії, розроблені для інших типів зображень. Будь-який метод стиснення напівтонових образів дозволяє стискати кольорові зображення. У кольоровому зображенні кожен піксель складається з трьох кольорових компонентів (наприклад, RGB). Уявімо кольорове зображення, в якому всі компоненти складаються з одного байта. Піксель описується трьома байтами чи 24 бітами, але ці біти не можна розглядати як одне число. Три пікселі

$$118|206|12 \qquad \text{та} \qquad 117|206|12$$

розрізняються тільки на одиницю в першій компоненті, а тому вони мають близькі кольори. Якщо ж їх розглядати у вигляді 24-бітних чисел, то вони сильно відрізнятимуться, оскільки вони відрізняються у старшому біті. Будь-який метод компресії, який ґрунтуватиметься на такому розрізненні пікселів, буде дуже неоптимальним. У цьому сенсі найбільш підходящим є метод поділу зображення на кольорові компоненти з їхнім подальшим незалежним стиском методом напівтонової компресії.

Метод зважених контекстних дерев зображень є прикладом використання кодів RGC для стиснення образів.

Стиснення зображень. Коди Грея

До кольорових зображень можна застосовувати методи компресії, розроблені для інших типів зображень. Будь-який метод стиснення напівтонових образів дозволяє стискати кольорові зображення. У кольоровому зображенні кожен піксель складається з трьох кольорових компонентів (наприклад, RGB). Уявімо кольорове зображення, в якому всі компоненти складаються з одного байта. Піксель описується трьома байтами чи 24 бітами, але ці біти не можна розглядати як одне число. Три пікселі

$$118|206|12 \qquad \text{та} \qquad 117|206|12$$

розрізняються тільки на одиницю в першій компоненті, а тому вони мають близькі кольори. Якщо ж їх розглядати у вигляді 24-бітних чисел, то вони сильно відрізнятимуться, оскільки вони відрізняються у старшому біті. Будь-який метод компресії, який ґрунтуватиметься на такому розрізненні пікселів, буде дуже неоптимальним. У цьому сенсі найбільш підходящим є метод поділу зображення на кольорові компоненти з їхнім подальшим незалежним стиском методом напівтонової компресії.

Метод зважених контекстних дерев зображень є прикладом використання кодів RGC для стиснення образів.

Стиснення зображень. Коди Грея

До кольорових зображень можна застосовувати методи компресії, розроблені для інших типів зображень. Будь-який метод стиснення напівтонових образів дозволяє стискати кольорові зображення. У кольоровому зображенні кожен піксель складається з трьох кольорових компонентів (наприклад, RGB). Уявімо кольорове зображення, в якому всі компоненти складаються з одного байта. Піксель описується трьома байтами чи 24 бітами, але ці біти не можна розглядати як одне число. Три пікселі

$$118|206|12 \qquad \text{та} \qquad 117|206|12$$

розрізняються тільки на одиницю в першій компоненті, а тому вони мають близькі кольори. Якщо ж їх розглядати у вигляді 24-бітних чисел, то вони сильно відрізнятимуться, оскільки вони відрізняються у старшому біті. Будь-який метод компресії, який ґрунтуватиметься на такому розрізненні пікселів, буде дуже неоптимальним. У цьому сенсі найбільш підходящим є метод поділу зображення на кольорові компоненти з їхнім подальшим незалежним стиском методом напівтонової компресії.

Метод зважених контекстних дерев зображень є прикладом використання кодів RGC для стиснення образів.

Стиснення зображень. Коди Грея

До кольорових зображень можна застосовувати методи компресії, розроблені для інших типів зображень. Будь-який метод стиснення напівтонових образів дозволяє стискати кольорові зображення. У кольоровому зображенні кожен піксель складається з трьох кольорових компонентів (наприклад, RGB). Уявімо кольорове зображення, в якому всі компоненти складаються з одного байта. Піксель описується трьома байтами чи 24 бітами, але ці біти не можна розглядати як одне число. Три пікселі

$$118|206|12 \qquad \text{та} \qquad 117|206|12$$

розрізняються тільки на одиницю в першій компоненті, а тому вони мають близькі кольори. Якщо ж їх розглядати у вигляді 24-бітних чисел, то вони сильно відрізнятимуться, оскільки вони відрізняються у старшому біті. Будь-який метод компресії, який ґрунтуватиметься на такому розрізненні пікселів, буде дуже неоптимальним. У цьому сенсі найбільш підходящим є метод поділу зображення на кольорові компоненти з їхнім подальшим незалежним стиском методом напівтонової компресії.

Метод зважених контекстних дерев зображень є прикладом використання кодів RGC для стиснення образів.

Стиснення зображень. Коди Грея

До кольорових зображень можна застосовувати методи компресії, розроблені для інших типів зображень. Будь-який метод стиснення напівтонових образів дозволяє стискати кольорові зображення. У кольоровому зображенні кожен піксель складається з трьох кольорових компонентів (наприклад, RGB). Уявімо кольорове зображення, в якому всі компоненти складаються з одного байта. Піксель описується трьома байтами чи 24 бітами, але ці біти не можна розглядати як одне число. Три пікселі

$$118|206|12 \qquad \text{та} \qquad 117|206|12$$

розрізняються тільки на одиницю в першій компоненті, а тому вони мають близькі кольори. Якщо ж їх розглядати у вигляді 24-бітних чисел, то вони сильно відрізнятимуться, оскільки вони відрізняються у старшому біті. Будь-який метод компресії, який ґрунтуватиметься на такому розрізненні пікселів, буде дуже неоптимальним. У цьому сенсі найбільш підходящим є метод поділу зображення на кольорові компоненти з їхнім подальшим незалежним стиском методом напівтонової компресії.

Метод зважених контекстних дерев зображень є прикладом використання кодів RGC для стиснення образів.

Стиснення зображень. Коди Грея

До кольорових зображень можна застосовувати методи компресії, розроблені для інших типів зображень. Будь-який метод стиснення напівтонових образів дозволяє стискати кольорові зображення. У кольоровому зображенні кожен піксель складається з трьох кольорових компонентів (наприклад, RGB). Уявімо кольорове зображення, в якому всі компоненти складаються з одного байта. Піксель описується трьома байтами чи 24 бітами, але ці біти не можна розглядати як одне число. Три пікселі

$$118|206|12 \qquad \text{та} \qquad 117|206|12$$

розрізняються тільки на одиницю в першій компоненті, а тому вони мають близькі кольори. Якщо ж їх розглядати у вигляді 24-бітних чисел, то вони сильно відрізнятимуться, оскільки вони відрізняються у старшому біті. Будь-який метод компресії, який ґрунтуватиметься на такому розрізненні пікселів, буде дуже неоптимальним. У цьому сенсі найбільш підходящим є метод поділу зображення на кольорові компоненти з їхнім подальшим незалежним стиском методом напівтонової компресії.

Метод зважених контекстних дерев зображень є прикладом використання кодів RGC для стиснення образів.

Стиснення зображень. Коди Грея

До кольорових зображень можна застосовувати методи компресії, розроблені для інших типів зображень. Будь-який метод стиснення напівтонових образів дозволяє стискати кольорові зображення. У кольоровому зображенні кожен піксель складається з трьох кольорових компонентів (наприклад, RGB). Уявімо кольорове зображення, в якому всі компоненти складаються з одного байта. Піксель описується трьома байтами чи 24 бітами, але ці біти не можна розглядати як одне число. Три пікселі

$$118|206|12 \qquad \text{та} \qquad 117|206|12$$

розрізняються тільки на одиницю в першій компоненті, а тому вони мають близькі кольори. Якщо ж їх розглядати у вигляді 24-бітних чисел, то вони сильно відрізнятимуться, оскільки вони відрізняються у старшому біті. Будь-який метод компресії, який ґрунтуватиметься на такому розрізненні пікселів, буде дуже неоптимальним. У цьому сенсі найбільш підходящим є метод поділу зображення на кольорові компоненти з їхнім подальшим незалежним стиском методом напівтонової компресії.

Метод зважених контекстних дерев зображень є прикладом використання кодів RGC для стиснення образів.

Стиснення зображень. Коди Грея

До кольорових зображень можна застосовувати методи компресії, розроблені для інших типів зображень. Будь-який метод стиснення напівтонових образів дозволяє стискати кольорові зображення. У кольоровому зображенні кожен піксель складається з трьох кольорових компонентів (наприклад, RGB). Уявімо кольорове зображення, в якому всі компоненти складаються з одного байта. Піксель описується трьома байтами чи 24 бітами, але ці біти не можна розглядати як одне число. Три пікселі

$$118|206|12 \qquad \text{та} \qquad 117|206|12$$

розрізняються тільки на одиницю в першій компоненті, а тому вони мають близькі кольори. Якщо ж їх розглядати у вигляді 24-бітних чисел, то вони сильно відрізнятимуться, оскільки вони відрізняються у старшому біті. Будь-який метод компресії, який ґрунтуватиметься на такому розрізненні пікселів, буде дуже неоптимальним. У цьому сенсі найбільш підходящим є метод поділу зображення на кольорові компоненти з їхнім подальшим незалежним стиском методом напівтонової компресії.

Метод зважених контекстних дерев зображень є прикладом використання кодів RGC для стиснення образів.

Стиснення зображень. Коди Грея

До кольорових зображень можна застосовувати методи компресії, розроблені для інших типів зображень. Будь-який метод стиснення напівтонових образів дозволяє стискати кольорові зображення. У кольоровому зображенні кожен піксель складається з трьох кольорових компонентів (наприклад, RGB). Уявімо кольорове зображення, в якому всі компоненти складаються з одного байта. Піксель описується трьома байтами чи 24 бітами, але ці біти не можна розглядати як одне число. Три пікселі

$$118|206|12 \qquad \text{та} \qquad 117|206|12$$

розрізняються тільки на одиницю в першій компоненті, а тому вони мають близькі кольори. Якщо ж їх розглядати у вигляді 24-бітних чисел, то вони сильно відрізнятимуться, оскільки вони відрізняються у старшому біті. Будь-який метод компресії, який ґрунтуватиметься на такому розрізненні пікселів, буде дуже неоптимальним. У цьому сенсі найбільш підходящим є метод поділу зображення на кольорові компоненти з їхнім подальшим незалежним стиском методом напівтонової компресії.

Метод зважених контекстних дерев зображень є прикладом використання кодів RGC для стиснення образів.

Стиснення зображень. Коди Грея

До кольорових зображень можна застосовувати методи компресії, розроблені для інших типів зображень. Будь-який метод стиснення напівтонових образів дозволяє стискати кольорові зображення. У кольоровому зображенні кожен піксель складається з трьох кольорових компонентів (наприклад, RGB). Уявімо кольорове зображення, в якому всі компоненти складаються з одного байта. Піксель описується трьома байтами чи 24 бітами, але ці біти не можна розглядати як одне число. Три пікселі

$$118|206|12 \qquad \text{та} \qquad 117|206|12$$

розрізняються тільки на одиницю в першій компоненті, а тому вони мають близькі кольори. Якщо ж їх розглядати у вигляді 24-бітних чисел, то вони сильно відрізнятимуться, оскільки вони відрізняються у старшому біті. Будь-який метод компресії, який ґрунтуватиметься на такому розрізненні пікселів, буде дуже неоптимальним. У цьому сенсі найбільш підходящим є метод поділу зображення на кольорові компоненти з їхнім подальшим незалежним стиском методом напівтонової компресії.

Метод зважених контекстних дерев зображень є прикладом використання кодів RGC для стиснення образів.

Історія кодів Грея

Ці коди названі на честь Франка Грея (Prank Gray), який запатентував їхнє використання у кодерах у 1953 році. Однак ця робота була виконана суттєво раніше: він її подав для патентування вже 1947 року. Грей працював дослідником у лабораторії Белла. Протягом 1930-х та 1940-х років він отримав кілька патентів у галузі телебачення. Якщо вірити деяким історичним дослідженням, то ці коди вже застосовувалися Баудотом (J. M. E. Baudot) в телеграфії в 1870-х роках, але тільки після винаходу комп'ютерів ці коди стали широко відомі.

Коди Баудота складалися із п'яти біт на символ. З їхньою допомогою можна представити $32 \times 2 - 2 = 62$ символи (кожен код має два зміст або значення, зміст позначався LS і FS кодами). Вони стали дуже популярними, і в 1950 році вони були прийняті як стандарт N1 міжнародного телеграфу. Вони також використовувалися у багатьох комп'ютерах першого та другого покоління.

Історія кодів Грея

Ці коди названі на честь Франка Грея (Prank Gray), який запатентував їхнє використання у кодерах у 1953 році. Однак ця робота була виконана суттєво раніше: він її подав для патентування вже 1947 року. Грей працював дослідником у лабораторії Белла. Протягом 1930-х та 1940-х років він отримав кілька патентів у галузі телебачення. Якщо вірити деяким історичним дослідженням, то ці коди вже застосовувалися Баудотом (J. M. E. Baudot) в телеграфії в 1870-х роках, але тільки після винаходу комп'ютерів ці коди стали широко відомі.

Коди Баудота складалися із п'яти біт на символ. З їхньою допомогою можна представити $32 \times 2 - 2 = 62$ символи (кожен код має два зміст або значення, зміст позначався LS і FS кодами). Вони стали дуже популярними, і в 1950 році вони були прийняті як стандарт N1 міжнародного телеграфу. Вони також використовувалися у багатьох комп'ютерах першого та другого покоління.

Історія кодів Грея

Ці коди названі на честь Франка Грея (Prank Gray), який запатентував їхнє використання у кодерах у 1953 році. Однак ця робота була виконана суттєво раніше: він її подав для патентування вже 1947 року. Грей працював дослідником у лабораторії Белла. Протягом 1930-х та 1940-х років він отримав кілька патентів у галузі телебачення. Якщо вірити деяким історичним дослідженням, то ці коди вже застосовувалися Баудотом (J. M. E. Baudot) в телеграфії в 1870-х роках, але тільки після винаходу комп'ютерів ці коди стали широко відомі.

Коди Баудота складалися із п'яти біт на символ. З їхньою допомогою можна представити $32 \times 2 - 2 = 62$ символи (кожен код має два зміст або значення, зміст позначався LS і FS кодами). Вони стали дуже популярними, і в 1950 році вони були прийняті як стандарт N1 міжнародного телеграфу. Вони також використовувалися у багатьох комп'ютерах першого та другого покоління.

Історія кодів Грея

Ці коди названі на честь Франка Грея (Prank Gray), який запатентував їхнє використання у кодерах у 1953 році. Однак ця робота була виконана суттєво раніше: він її подав для патентування вже 1947 року. Грей працював дослідником у лабораторії Белла. Протягом 1930-х та 1940-х років він отримав кілька патентів у галузі телебачення. Якщо вірити деяким історичним дослідженням, то ці коди вже застосовувалися Баудотом (J. M. E. Baudot) в телеграфії в 1870-х роках, але тільки після винаходу комп'ютерів ці коди стали широко відомі.

Коди Баудота складалися із п'яти біт на символ. З їхньою допомогою можна представити $32 \times 2 - 2 = 62$ символи (кожен код має два зміст або значення, зміст позначався LS і FS кодами). Вони стали дуже популярними, і в 1950 році вони були прийняті як стандарт N1 міжнародного телеграфу. Вони також використовувалися у багатьох комп'ютерах першого та другого покоління.

Історія кодів Грея

Ці коди названі на честь Франка Грея (Prank Gray), який запатентував їхнє використання у кодерах у 1953 році. Однак ця робота була виконана суттєво раніше: він її подав для патентування вже 1947 року. Грей працював дослідником у лабораторії Белла. Протягом 1930-х та 1940-х років він отримав кілька патентів у галузі телебачення. Якщо вірити деяким історичним дослідженням, то ці коди вже застосовувалися Баудотом (J. M. E. Baudot) в телеграфії в 1870-х роках, але тільки після винаходу комп'ютерів ці коди стали широко відомі.

Коди Баудота складалися із п'яти біт на символ. З їхньою допомогою можна представити $32 \times 2 - 2 = 62$ символи (кожен код має два зміст або значення, зміст позначався LS і FS кодами). Вони стали дуже популярними, і в 1950 році вони були прийняті як стандарт N1 міжнародного телеграфу. Вони також використовувалися у багатьох комп'ютерах першого та другого покоління.

Історія кодів Грея

Ці коди названі на честь Франка Грея (Prank Gray), який запатентував їхнє використання у кодерах у 1953 році. Однак ця робота була виконана суттєво раніше: він її подав для патентування вже 1947 року. Грей працював дослідником у лабораторії Белла. Протягом 1930-х та 1940-х років він отримав кілька патентів у галузі телебачення. Якщо вірити деяким історичним дослідженням, то ці коди вже застосовувалися Баудотом (J. M. E. Baudot) в телеграфії в 1870-х роках, але тільки після винаходу комп'ютерів ці коди стали широко відомі.

Коди Баудота складалися із п'яти біт на символ. З їхньою допомогою можна представити $32 \times 2 - 2 = 62$ символи (кожен код має два зміст або значення, зміст позначався LS і FS кодами). Вони стали дуже популярними, і в 1950 році вони були прийняті як стандарт N1 міжнародного телеграфу. Вони також використовувалися у багатьох комп'ютерах першого та другого покоління.

Історія кодів Грея

Ці коди названі на честь Франка Грея (Prank Gray), який запатентував їхнє використання у кодерах у 1953 році. Однак ця робота була виконана суттєво раніше: він її подав для патентування вже 1947 року. Грей працював дослідником у лабораторії Белла. Протягом 1930-х та 1940-х років він отримав кілька патентів у галузі телебачення. Якщо вірити деяким історичним дослідженням, то ці коди вже застосовувалися Баудотом (J. M. E. Baudot) в телеграфії в 1870-х роках, але тільки після винаходу комп'ютерів ці коди стали широко відомі.

Коди Баудота складалися із п'яти біт на символ. З їхньою допомогою можна представити $32 \times 2 - 2 = 62$ символи (кожен код має два зміст або значення, зміст позначався LS і FS кодами). Вони стали дуже популярними, і в 1950 році вони були прийняті як стандарт N1 міжнародного телеграфу. Вони також використовувалися у багатьох комп'ютерах першого та другого покоління.

Історія кодів Грея

Ці коди названі на честь Франка Грея (Prank Gray), який запатентував їхнє використання у кодерах у 1953 році. Однак ця робота була виконана суттєво раніше: він її подав для патентування вже 1947 року. Грей працював дослідником у лабораторії Белла. Протягом 1930-х та 1940-х років він отримав кілька патентів у галузі телебачення. Якщо вірити деяким історичним дослідженням, то ці коди вже застосовувалися Баудотом (J. M. E. Baudot) в телеграфії в 1870-х роках, але тільки після винаходу комп'ютерів ці коди стали широко відомі.

Коди Баудота складалися із п'яти біт на символ. З їхньою допомогою можна представити $32 \times 2 - 2 = 62$ символи (кожен код має два зміст або значення, зміст позначався LS і FS кодами). Вони стали дуже популярними, і в 1950 році вони були прийняті як стандарт N1 міжнародного телеграфу. Вони також використовувалися у багатьох комп'ютерах першого та другого покоління.

Історія кодів Грея

Ці коди названі на честь Франка Грея (Prank Gray), який запатентував їхнє використання у кодерах у 1953 році. Однак ця робота була виконана суттєво раніше: він її подав для патентування вже 1947 року. Грей працював дослідником у лабораторії Белла. Протягом 1930-х та 1940-х років він отримав кілька патентів у галузі телебачення. Якщо вірити деяким історичним дослідженням, то ці коди вже застосовувалися Баудотом (J. M. E. Baudot) в телеграфії в 1870-х роках, але тільки після винаходу комп'ютерів ці коди стали широко відомі.

Коди Баудота складалися із п'яти біт на символ. З їхньою допомогою можна представити $32 \times 2 - 2 = 62$ символи (кожен код має два зміст або значення, зміст позначався LS і FS кодами). Вони стали дуже популярними, і в 1950 році вони були прийняті як стандарт N1 міжнародного телеграфу. Вони також використовувалися у багатьох комп'ютерах першого та другого покоління.

Історія кодів Грея

Ці коди названі на честь Франка Грея (Prank Gray), який запатентував їхнє використання у кодерах у 1953 році. Однак ця робота була виконана суттєво раніше: він її подав для патентування вже 1947 року. Грей працював дослідником у лабораторії Белла. Протягом 1930-х та 1940-х років він отримав кілька патентів у галузі телебачення. Якщо вірити деяким історичним дослідженням, то ці коди вже застосовувалися Баудотом (J. M. E. Baudot) в телеграфії в 1870-х роках, але тільки після винаходу комп'ютерів ці коди стали широко відомі.

Коди Баудота складалися із п'яти біт на символ. З їхньою допомогою можна представити $32 \times 2 - 2 = 62$ символи (кожен код має два зміст або значення, зміст позначався LS і FS кодами). Вони стали дуже популярними, і в 1950 році вони були прийняті як стандарт N1 міжнародного телеграфу. Вони також використовувалися у багатьох комп'ютерах першого та другого покоління.

Історія кодів Грея

Ці коди названі на честь Франка Грея (Prank Gray), який запатентував їхнє використання у кодерах у 1953 році. Однак ця робота була виконана суттєво раніше: він її подав для патентування вже 1947 року. Грей працював дослідником у лабораторії Белла. Протягом 1930-х та 1940-х років він отримав кілька патентів у галузі телебачення. Якщо вірити деяким історичним дослідженням, то ці коди вже застосовувалися Баудотом (J. M. E. Baudot) в телеграфії в 1870-х роках, але тільки після винаходу комп'ютерів ці коди стали широко відомі.

Коди Баудота складалися із п'яти біт на символ. З їхньою допомогою можна представити $32 \times 2 - 2 = 62$ символи (кожен код має два зміст або значення, зміст позначався LS і FS кодами). Вони стали дуже популярними, і в 1950 році вони були прийняті як стандарт N1 міжнародного телеграфу. Вони також використовувалися у багатьох комп'ютерах першого та другого покоління.

Історія кодів Грея

Ці коди названі на честь Франка Грея (Prank Gray), який запатентував їхнє використання у кодерах у 1953 році. Однак ця робота була виконана суттєво раніше: він її подав для патентування вже 1947 року. Грей працював дослідником у лабораторії Белла. Протягом 1930-х та 1940-х років він отримав кілька патентів у галузі телебачення. Якщо вірити деяким історичним дослідженням, то ці коди вже застосовувалися Баудотом (J. M. E. Baudot) в телеграфії в 1870-х роках, але тільки після винаходу комп'ютерів ці коди стали широко відомі.

Коди Баудота складалися із п'яти біт на символ. З їхньою допомогою можна представити $32 \times 2 - 2 = 62$ символи (кожен код має два зміст або значення, зміст позначався LS і FS кодами). Вони стали дуже популярними, і в 1950 році вони були прийняті як стандарт N1 міжнародного телеграфу. Вони також використовувалися у багатьох комп'ютерах першого та другого покоління.

Історія кодів Грея

Ці коди названі на честь Франка Грея (Prank Gray), який запатентував їхнє використання у кодерах у 1953 році. Однак ця робота була виконана суттєво раніше: він її подав для патентування вже 1947 року. Грей працював дослідником у лабораторії Белла. Протягом 1930-х та 1940-х років він отримав кілька патентів у галузі телебачення. Якщо вірити деяким історичним дослідженням, то ці коди вже застосовувалися Баудотом (J. M. E. Baudot) в телеграфії в 1870-х роках, але тільки після винаходу комп'ютерів ці коди стали широко відомі.

Коди Баудота складалися із п'яти біт на символ. З їхньою допомогою можна представити $32 \times 2 - 2 = 62$ символи (кожен код має два зміст або значення, зміст позначався LS і FS кодами). Вони стали дуже популярними, і в 1950 році вони були прийняті як стандарт N1 міжнародного телеграфу. Вони також використовувалися у багатьох комп'ютерах першого та другого покоління.

Історія кодів Грея

Ці коди названі на честь Франка Грея (Prank Gray), який запатентував їхнє використання у кодерах у 1953 році. Однак ця робота була виконана суттєво раніше: він її подав для патентування вже 1947 року. Грей працював дослідником у лабораторії Белла. Протягом 1930-х та 1940-х років він отримав кілька патентів у галузі телебачення. Якщо вірити деяким історичним дослідженням, то ці коди вже застосовувалися Баудотом (J. M. E. Baudot) в телеграфії в 1870-х роках, але тільки після винаходу комп'ютерів ці коди стали широко відомі.

Коди Баудота складалися із п'яти біт на символ. З їхньою допомогою можна представити $32 \times 2 - 2 = 62$ символи (кожен код має два зміст або значення, зміст позначався LS і FS кодами). Вони стали дуже популярними, і в 1950 році вони були прийняті як стандарт N1 міжнародного телеграфу. Вони також використовувалися у багатьох комп'ютерах першого та другого покоління.

Дякую за увагу!