НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. Ігоря СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Комп'ютерний практикум №5
з курсу алгоритми кодування двійкових даних

РЕАЛІЗАЦІЯ АЛГОРИТМУ СТИСКАННЯ ДАНИХ LZW

Виконав студент
групи ФІ-42мн
Бєш Радомир Андрійович

Київ — 2026

Зміст

1 Мета

Опанувати методи підвищення ефективності алгоритмів стискання даних: перетворення Барроуза-Вілера та MTF-перетворення.

2 Постановка задачі

У даній рботі необхідно реалізувати дві окремі процедури:

- перетворення Барроуза-Вілера;
- перетворення Move-to-Front («Стопки книг»).

Дані процедури повинні бути реалізовані у вигляді окремих функій, які повинні бути легко інтегровані із нашими реалізаціями алгоритмів стискання даних Хаффмана та LZW, - тобто, вони повинні викликатись (разово чи неперервно) між функціями зчитування/запису даних файлів та функціями безпосередньо стиснення.
Утиліти стискання даних повинні передбачати окремі налаштування, які вмикають чи вимикають використання методів покращення.
Результати дослідження:
`https://github.com/Radomir21/Encoding-Algorithms/tree/main/lab_5`.

3 Хід роботи

3.1 Особливості реалізації перетворень Барроуза-Вілера та MTF

1. **Реалізація перетворення Барроуза–Вілера (BWT):**
   Перетворення Барроуза–Вілера реалізовано блоковим способом. Вхідні дані розбиваються на блоки фіксованого розміру (4096 байтів), і для кожного блоку перетворення виконується незалежно.
   Для кожного блоку:
   - Формується множина всіх циклічних зсувів блоку.
   - Зсуви сортуються в лексикографічному порядку.
   - Результатом є послідовність байтів, що відповідає останньому стовпцю відсортованої матриці зсувів.
   - Результатом перетворення є послідовність байтів, що відповідає останньому стовпцю відсортованої матриці зсувів.

   Для коректного зворотного перетворення BWT необхідно зберегти primary index для кожного блоку. У реалізації ці значення: зберігаються у заголовку стиснутого файлу, записуються як масив 32-бітних цілих чисел (uint32), кількість збережених індексів відповідає кількості блоків, на які був розбитий файл. При декодуванні ці індекси використовуються для відновлення початкового порядку символів у кожному блоці за допомогою Last-First - відображення.

2. **Реалізація перетворення Move-to-Front (MTF):**
   Перетворення Move-to-Front реалізовано для байтового алфавіту, що складається з 256 можливих значень (0–255). На початку кодування алфавіт ініціалізується у вигляді впорядкованого списку всіх байтів у зростаючому порядку.
   Принцип роботи MTF:

- Для кожного вхідного байта визначається його позиція у поточному алфавіті.
- У вихідний потік записується номер позиції цього символу.
- Символ переміщується на початок алфавіту (операція move-to-front).

Таким чином, часто повторювані символи починають кодуватися малими числовими значеннями, що призводить до появи великої кількості малих кодів у вихідній послідовності.
При декодуванні використовується ідентичний початковий алфавіт. Для кожного зчитаного числового значення: Вибирається символ із відповідної позиції алфавіту, символ додається до вихідного потоку, алфавіт оновлюється шляхом переміщення цього символу на початок.

3. **Взаємодія BWT та MTF з алгоритмами стискання:**
Реалізовані перетворення можуть застосовуватися як окремо так і спільно. Найбільший ефект досягається при послідовному застосуванні BWT → MTF → алгоритм стискання, оскільки:

   - BWT групує однакові символи;
   - MTF перетворює ці групи у послідовності малих чисел;
   - Хаффман або LZW ефективно стискають таку структуру даних.

   Декодування виконуються у зворотному порядку, що забезпечує повне відновлення початкового файлу без втрати інформації.

3.2 Порівняльний аналіз

Для аналізу було використано такі типи файлів по 10 файлів кожного типу:

- **TXT**
- **CSV**
- **BMP**
- **PDF**
- **EXE**

Коефіцієнт стискання розраховується, як:

$$K = \frac{\text{SIZE}_{\text{архів}}}{\text{SIZE}_{\text{оригінал}}} \tag{1}$$

Результати:
сорі, що не нормальною таблицею

Хаффман:

```
=== Compression coefficients (K = compressed / original) ===

Folder: bmp tests
  HUFF           : n=10  avg=0.6824  min=0.2315  max=1.1043
  HUFF+BWT       : n=10  avg=0.6834  min=0.2325  max=1.1056
  HUFF+MTF       : n=10  avg=0.5628  min=0.2038  max=1.1027
  HUFF+BWT+MTF   : n=10  avg=0.4710  min=0.1454  max=1.0144

Folder: csv tests
  HUFF           : n=10  avg=0.5971  min=0.4349  max=0.7725
  HUFF+BWT       : n=10  avg=0.5981  min=0.4359  max=0.7740
  HUFF+MTF       : n=10  avg=0.5995  min=0.4505  max=0.8130
  HUFF+BWT+MTF   : n=10  avg=0.3629  min=0.1931  max=0.6157

Folder: exe tests
  HUFF           : n=10  avg=0.6842  min=0.3876  max=0.8389
  HUFF+BWT       : n=10  avg=0.6852  min=0.3886  max=0.8399
  HUFF+MTF       : n=10  avg=0.6482  min=0.3034  max=0.8269
  HUFF+BWT+MTF   : n=10  avg=0.4930  min=0.2431  max=0.6553

Folder: pdf tests
  HUFF           : n=10  avg=0.9955  min=0.9897  max=1.0006
  HUFF+BWT       : n=10  avg=0.9965  min=0.9907  max=1.0015
  HUFF+MTF       : n=10  avg=0.9905  min=0.9830  max=0.9963
  HUFF+BWT+MTF   : n=10  avg=0.9674  min=0.9392  max=0.9884

Folder: txt tests
  HUFF           : n=10  avg=0.5918  min=0.5692  max=0.6274
  HUFF+BWT       : n=10  avg=0.5928  min=0.5702  max=0.6284
  HUFF+MTF       : n=10  avg=0.6390  min=0.6150  max=0.6703
  HUFF+BWT+MTF   : n=10  avg=0.4602  min=0.4136  max=0.4777
```

LZW:

```
=== Compression coefficients (K = compressed / original) ===

Folder: bmp tests2
  LZW            : n=10  avg=0.5120  min=0.0231  max=1.2182
  LZW+BWT        : n=10  avg=0.5423  min=0.0334  max=1.3010
  LZW+MTF        : n=10  avg=0.6242  min=0.0332  max=1.3464
  LZW+BWT+MTF    : n=10  avg=0.5306  min=0.0368  max=1.2479

Folder: csv tests2
  LZW            : n=10  avg=0.4324  min=0.1311  max=0.7861
  LZW+BWT        : n=10  avg=0.4275  min=0.1514  max=0.8029
  LZW+MTF        : n=10  avg=0.6623  min=0.3927  max=1.1078
  LZW+BWT+MTF    : n=10  avg=0.4173  min=0.1441  max=0.7673

Folder: exe tests2
  LZW            : n=10  avg=0.6381  min=0.2554  max=0.9022
  LZW+BWT        : n=10  avg=0.6398  min=0.2462  max=0.9007
  LZW+MTF        : n=10  avg=0.7656  min=0.2849  max=1.0547
  LZW+BWT+MTF    : n=10  avg=0.6235  min=0.2291  max=0.8811

Folder: pdf tests2
  LZW            : n=10  avg=1.3300  min=1.2533  max=1.4080
  LZW+BWT        : n=10  avg=1.3598  min=1.2952  max=1.4175
  LZW+MTF        : n=10  avg=1.3745  min=1.3259  max=1.4159
  LZW+BWT+MTF    : n=10  avg=1.3647  min=1.3029  max=1.4181

Folder: txt tests2
  LZW            : n=10  avg=0.4570  min=0.4051  max=0.4969
  LZW+BWT        : n=10  avg=0.5514  min=0.4982  max=0.5786
  LZW+MTF        : n=10  avg=0.7871  min=0.7440  max=0.8468
  LZW+BWT+MTF    : n=10  avg=0.5488  min=0.4916  max=0.5745
```

4 Висновки

У даній лабораторній роботі було реалізовано та досліджено вплив перетворень Барроуза–Вілера (BWT) та Move-to-Front (MTF) на ефективність алгоритмів стискання даних Хаффмана та LZW. Експериментальне дослідження проводилося на п'яти різних типах файлів (BMP, CSV, EXE, PDF, TXT), по десять файлів кожного типу, з використанням чотирьох режимів стискання: без оптимізацій, з BWT, з MTF та з BWT+MTF.

Для текстових та табличних даних (TXT, CSV):

1. Найкращі результати забезпечує Хаффман з комбінацією BWT+MTF.
2. LZW також показує хороші результати, але поступається оптимізованому Хаффману.

Для зображень BMP:

1. Обидва алгоритми показують ефективне стискання.

Для виконуваних файлів EXE:

1. Обидва алгоритми стискають помірно.
2. Оптимізації BWT+MTF знову ж таки краще працюють з Хаффманом.

Для PDF-файлів:

1. Жоден з алгоритмів не демонструє суттєвого виграшу.
2. Повторне стискання вже стиснених форматів є малоефективним.

Отримані результати повністю відповідають теоретичним властивостям досліджуваних алгоритмів. Перетворення BWT та MTF є ефективними методами попередньої обробки даних для ентропійного стискання, зокрема алгоритму Хаффмана. Найкращі результати досягаються при послідовному застосуванні BWT -> MTF -> Хаффман. Для алгоритму LZW ці перетворення мають обмежену ефективність і в ряді випадків можуть навіть погіршувати результат стискання.

Таким чином, у межах лабораторної роботи було експериментально підтверджено, що вибір алгоритму стискання та попередніх перетворень повинен враховувати структуру та тип вхідних даних.