Задача № 3

## Преобразование Берроуза-Уилера (BWT)

Данное преобразование позволяет выстроить символы в порядке следования контекстов, т.е. символы, идущие в одном контексте, группируются вместе. Дадим описание метода в версии, когда контекст (как это принято в теории информации) образован предыдущими символами, а не последующими (как это делается во многих поверхностных описаниях метода в Интернете).

Пусть дано сообщение (текст) `message`. Составим массив из циклических сдвигов влево исходного текста. Затем отсортируем массив, причем сравнение строк при сортировке будем начинать с *предпоследнего* символа, двигаясь к началу строки.

| | Исходный массив | Отсортированный массив |
|---|---|---|
| 0 | message | emessag |
| 1 | essagem | essagem |
| 2 | ssageme | sagemes |
| 3 | sagemes | message |
| 4 | agemess | ssageme |
| 5 | gemessa | agemess |
| 6 | emessag | gemessa |

Результатом преобразования является последний столбец отсортированного массива с указанием позиции в нем первого символа исходного текста. В нашем случае результат преобразования будет `gmseesa 1`. Мы видим, что буквы выстроены по контекстам, упорядоченным в алфавитном порядке.

Обратное преобразование делается так. Сортируем буквы в алфавитном порядке, запоминая их исходные позиции, т.е. получаем массив пар <буква, исходная позиция>, упорядоченный по буквам (при этом одинаковые буквы должны быть упорядочены по номерам позиций). Затем переписываем буквы во второй массив по номерам исходных позиций, запоминая позицию буквы в первом (отсортированном) массиве. Начиная с позиции, указанной как второй параметр преобразования (в нашем примере – 1), переписываем букву из второго массива на выход, переходя на позицию буквы в отсортированном массиве, и продолжаем этот процесс, пока не восстановим все сообщение.

| | Вход | Первый массив | | Второй массив | | Выход |
|---|---|---|---|---|---|---|
| 0 | g | a | 6 | g | 3 | m |
| 1 | m | e | 3 | m | 4 | e |
| 2 | s | e | 4 | s | 5 | s |
| 3 | e | g | 0 | e | 1 | s |
| 4 | e | m | 1 | e | 2 | a |
| 5 | s | s | 2 | s | 6 | g |
| 6 | a | s | 5 | a | 0 | e |

**Задание.** Запрограммировать прямое и обратное преобразование BWT. Подвергнуть результат преобразования сжатию методом "Стопка книг". Сравнить степени сжатия при применении "Стопки книг" к исходному и преобразованному текстам. При реализации BWT желательно использовать хороший метод сортировки, например, QSort, псевдокод которого приведен ниже.

A – сортируемый массив, L и R – соответственно левая и правая границы массива.

```
QSort (L, R)
  while (L < R)
    x = A[L], i = L, j = R;
    while (i <= j)
      while (A[i] < x) i++;
      while (x < A[j]) j--;
      if (i <= j) A[i] ↔ A[j], i++, j--;
    if ((j - L) > (R - i)) QSort (i, R), R = j;
    else QSort (L, j), L = i;
```