**Задача № 1**

Во всех вариантах данной задачи необходимо реализовать указанный алгоритм сортировки массива целых чисел.

Количество чисел в массиве определяется окончанием стандартного потока ввода и заранее не известно.

**1\_1. Сортировка выбором.**

**1\_2. Сортировка вставками.**

**1\_3. Сортировка пузырьком.**

|  |  |
| --- | --- |
| in | out |
| 3  1  2 | 1  2  3 |

**Задача № 2**

Во всех вариантах данной задачи необходимо использовать **пирамидальную** сортировку.

**2\_4. Строки.**

Напишите программу, печатающую набор строк в лексикографическом порядке.

Строки разделяются символом перевода строки '\n'. Если последний символ в потоке ввода '\n', считать, что после него нет пустой строки. Максимальная длина строки 255 символов.

|  |  |
| --- | --- |
| in | out |
| 4  caba  abba  ab  aba | ab  aba  abba  caba |

**Задача № 3**

Во всех данного раздела необходимо использовать сортировку **слиянием**.

**3\_2. Современники.**

Группа людей называется современниками если был такой момент, когда они могли собраться вместе. Для этого в этот момент каждому из них должно было  уже исполниться 18 лет, но ещё не исполниться 80 лет.

Дан список Жизни Великих Людей. Необходимо получить максимальное количество современников. В день 18летия человек уже может принимать участие в собраниях, а в день 80летия и в день смерти уже не может.

|  |  |
| --- | --- |
| In | Out |
| 3  2 5 1980 13 11 2055  1 1 1982 1 1 2030  2 1 1920 2 1 2000 | 3 |

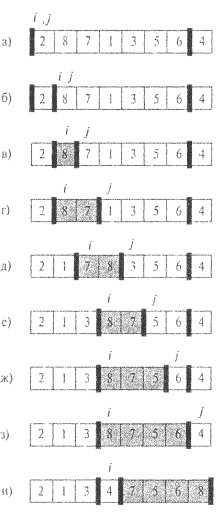
**Задача № 4**

Даны неотрицательные целые числа n,k и массив целых чисел из [0..10^9] размера n. Требуется найти k-ю порядковую статистику. т.е. напечатать число, которое бы стояло на позиции с индексом k (0..n-1) в отсортированном массиве. Напишите нерекурсивный алгоритм.

Требования к дополнительной памяти: O(n). Требуемое среднее время работы: O(n).

Функцию Partition следует реализовывать методом прохода двумя итераторами в одном направлении. Описание для случая прохода от начала массива к концу:

* Выбирается опорный элемент. Опорный элемент меняется с последним элементом массива.
* Во время работы Partition в начале массива содержатся элементы, не бОльшие опорного. Затем располагаются элементы, строго бОльшие опорного. В конце массива лежат нерассмотренные элементы. Последним элементом лежит опорный.
* Итератор i указывает на начало группы элементов, строго бОльших опорного.
* Итератор j больше i, итератор j указывает на первый нерассмотренный элемент.
* Шаг алгоритма. Рассматривается элемент, на который указывает j. Если он больше опорного, то сдвигаем j.  
  Если он не больше опорного, то меняем a[i] и a[j] местами, сдвигаем i и сдвигаем j.
* В конце работы алгоритма меняем опорный и элемент, на который указывает итератор i.



**4\_4.** Реализуйте стратегию выбора опорного элемента “случайный элемент”. Функцию Partition реализуйте методом прохода двумя итераторами от конца массива к началу.

|  |  |
| --- | --- |
| In | Out |
| 10 4  1 2 3 4 5 6 7 8 9 10 | 5 |
| 10 0  3 6 5 7 2 9 8 10 4 1 | 1 |
| 10 9  0 0 0 0 0 0 0 0 0 1 | 1 |

**Задача № 5**

**5\_1. Первые k элементов длинной последовательности.**

Дана очень длинная последовательность целых чисел длины n. Требуется вывести в отсортированном виде её первые k элементов. Последовательность может не помещаться в память. Время работы O(n \* log(k)). Доп. память O(k). Использовать слияние.

|  |  |
| --- | --- |
| In | Out |
| 9 4  3 7 4 5 6 1 15 4 2 | 1 2 3 4 |

**Задача № 6**

**6\_3. Binary MSD для long long.**

Дан массив неотрицательных целых 64-разрядных чисел. Количество чисел не больше 106. Отсортировать массив методом MSD по битам (бинарный QuickSort).

|  |  |
| --- | --- |
| In | Out |
| 3  4 1000000 7 | 4 7 1000000 |

**Задача № 7**

**7\_1. Быстрейшая сортировка.**

Дан массив целых чисел в диапазоне [0..10^9]. Размер массива кратен 10 и ограничен сверху значением 2.5 \* 10^7 элементов. Все значения массива являются элементами псевдо-рандомной последовательности. Необходимо отсортировать элементы массива за минимально время и вывести каждый **десятый** элемент отсортированной последовательности.

Минимальный набор оптимизаций, который необходимо реализовать:

1. Оптимизация ввода/вывода

2. Оптимизация выбора опорного элемента

3.

Оптимизация концевой рекурсии

|  |  |
| --- | --- |
| In | Out |
| 3 0 2 1 5 4 21 4 6 5 | 21 |