Задание 3. Сортировки

1 Заметим, что если у i-го подошедшего клиента время облуживания t_i, то суммарное время ожидания будет

$$T = \sum_{i=1}^{n}(n-i+1)t_i$$

Далее рассмотрим $t_i, t_j : i < j, t_i \geq t_j$. Если мы поменяем этих клиентов местами, T уменьшится на $(j-i)t_i$ и увеличится на $(j-i)t_j$, в результате T уменьшится на $(j-i)(t_i-t_j) \geq 0$, т. е. T не увеличится. Получаем, что расстановка с наименьшим суммарным временем ожидания — это расстановка при которой время ожидания каждого следующего клиента не убывает. Алгоритм тем самым должен просто отсортировать клиентов по убыванию времени ожидания. Для этого подойдет сортировка слиянием, работающая за $O(n\log n)$, что является наилучшим вариантом с точки зрения асимптотики.

2 Вначале выделим память под два массива: m длиной $k-1$, ord длиной n. Эти операции в общей сложности $\Theta(n)$. Затем, пройдем один раз по исходному массиву и на i-е место в массиве m запишем, сколько котов имеют массу не более i+2(нумерация массивов с 0). Это также займет $\Theta(n)$. Далее, пройдем исходный массив заново, совершая следующте действия:
Допустим, что кот с кличкой i имеет массу j. Также допустим, что до этого мы встретили p котов той же массы. Тогда мы сделаем запись $ord\big[m[j-3]+p\big] = i$(если $j = 2, ord[p] = i$). Получается, что мы разбили массив индексов на последовательные ячейки, каждая из которых содержит клички котов определенной массы, при этом эти ячейки стоят по возрастанию массы. Последняя операция также линейна по n. В итоге, алгоритм работает за $\Theta(n)$.

3 Установим три указателя на начало каждого из массивов и увеличим число уникальных элементов на нужное значение (если 3 уникальных то на три, если 2 на 2 и если все равны то на 1), в отдельных переменных будем хранить последнее число, записанное из каждого из массивов. Далее шаг следующий: после сдвига сначала оставляем только уникальные элементы на указателях, затем сравниваем каждое из чисел с соответствующим значением в переменной, и увеличиваем количество если не совпали. Если доходим до конца массива, соответствующий указатель останавливается. Итого, мы проходим каждый массив один раз. На каждом шаге делается число сравнений, не зависящих от длин массивов. Полученный алгоритм работает за $O(n)$.

4 На вход задачи поступает массив a из n чисел. Постройте алгоритм, находящий число инверсий в массиве, то есть таких пар индексов i, j, что $i < j$ и $a[i] > a[j]$.
Рекомендация: модифицируйте алгоритм сортировки слиянием.

5 Сперва заметим, что порядок, в котором стоят элементы массива только меняет местами слагаемые в сумме, то есть сама сумма от него не зависит. Поэтому далее считаем массив отсортированным по возрастанию. Поделим весь массив на пары $(a_k, a_{2n-k}), k = 0, \ldots, n-1$, т. е. так чтобы первый стоял с последним, второй с предпоследним и т. д. Далее рассмотрим конкретную пару (a_k, a_{2n-k}). Если число s находится между a_k и a_{2n-k} то сумма $|s-a_k|+|s-a_{2n-k}|$ равна $a_{2n-k}-a_k$. Если же s лежит за пределами этого отрезка, сумма будет строго больше $a_{2n-k}-a_k$. Отсюда получаем, что для всех пар число s должно быть внутри образованных этими парами отрезков. Остается заметить, что a_n стоит в паре с самим собой, т. е. образует отрезок из одной точки, лежащий в пересечении всех других отрезков. Тем самым, для минимизации суммы необходимо и достаточно, чтобы s совпадало с a_n. Так как мы предполагали, что массив отсортирован по возрастанию, для исходного массива s должно быть равно $a_{(n)}$ — n-ой порядковой статистике. Алгоритм находящий n-ую порядковую статистику работает за $O(n)$.

6 На вход подается массив $a_1, \ldots, a_n$, в котором один из элементов встречается не меньше $\lceil \frac{n}{2} \rceil$ раз. Постройте алгоритм, находящий этот элемент.

7 Дан массив из n чисел. Нужно разбить этот массив на максимальное количество непрерывных подмассивов так, чтобы после сортировки элементов внутри каждого подмассива весь массив стал отсортированным. Предложите $O(n\log n)$ алгоритм для решения этой задачи