Задание 4

Оценим сложность оптимального алгоритма снизу. Для этого посчитаем количество перестановок, таких что каждый элемент находится на расстоянии не более k от своего начального положения.

Рассмотрим первый элемент отсортированного массива. В неотсортированном массиве он может стоять на (k + 1) позиции. Рассмотрим второй элемент. Он может стоять на

(k + 2) позициях, но одна из них занята первым элементом => остаётся (k + 1) позиция. Рассмотрим i-ый элемент,

k + 1 <= i <= n – k (массив нумеруется с 1). Он может стоять на (2k + 1) позициях, но k из них уже заняты => остаётся

(k + 1) позиций. Таким образом для каждого элемента, индекс которого <= n - k есть (k + 1) позиция на которых они могут стоять. Таким образом для первых (n - k) элементов всего есть (k + 1)^(n - k) возможных перестановок.

Рассмотрим (n - k + 1)-ый элемент. Для него есть (2k) позиций, но k из них уже заняты, остаётся (k - 1) позиция. Рассмотрим (n - k + 2)-ой элемент. Для него есть (2k - 1) позиций, но k из них уже заняты, остаётся (k - 2) позиция. Рассмотрим последний элемент. Для него осталось ровно одна позиция. Таким образом для последних k элементов всего k! возможных перестановок.

Итого: k!*(k + 1)^(n – k) возможных перестановок. Каждая перестановка – это лист в дереве ветвлений. Значит высота этого дерева не меньше чем

$\log_2$ (k!*(k + 1)^(n – k)) = klogk + nlog(k + 1) - klog(k + 1) =

nlog(k + 1) - klog(1 + 1/k) = 1/2nlog(k+1) +

(1/2nlog(k+1)-klog(1+1/k)).

Докажем, что (1/2nlog(k+1)-klog(1+1/k)) > 0.

k^2>k+1, при k>2

$k+1>(k+1)^2/(k^2)$

$\log(k+1)>\log((1+1/k)^2)$

$\log(k+1)>2\log(1+1/k)$

$k\log(k+1)>2k\log(1+1/k)$

$n\log(k+1)>2k\log(1+1/k)$, так как $n>k$

$1/2n\log(k+1)>k\log(1+1/k)$

$1/2n\log(k+1)-k\log(1+1/k)) > 0$.

Таким образом оценка снизу сложности оптимального алгоритма равна $1/2n\log(k+1)$ или просто $n\log(k+1)$.

Теперь алгоритм. Создадим минимальную кучу на первых k элементах массива (То есть мы используем $O(k)$ дополнительной памяти). Теперь для каждого следующего элемента массива, начиная с $(k + 1)$-ого делаем следующее: добавляем этот элемент в кучу, извлекаем из кучи минимальный элемент и ставим его на $(i-k)$-ую позицию в массиве, где i - индекс текущего элемента. Когда больше не останется элементов в массиве, просто по очереди извлечем из кучи все k элементов и поставим на соответствующие позиции в массиве.

Сложность алгоритма: $O(n\log(k))$

Доказательство корректности алгоритма:

База индукции: первый элемент поставлен правильно. Доказательство: в неотсортированном массиве его индекс

$<= k + 1$, то есть его нам надо выбрать из $(k+1)$ элементов. Это мы и сделали: вначале в куче k элементов, на первом шаге мы добавили в неё ещё один, стало $(k+1)$ элементов, и из этих $(k+1)$ элементов мы выбрали минимум.

Шаг индукции: если все элементы до i стоят на своих местах (то есть как в отсортированном массиве), то i-ый элемент мы поставим правильно. Доказательство: в момент, когда из кучи извлекается минимум (то есть элемент, который мы поставим на i-тую позицию) в куче находятся элементы с

i - k до i + k (изначального неотсортированного массива), за вычетом элементов на позициях [i-k,i-1] (отсортированного массива), то есть тех элементов, которые уже стоят на правильных местах. Таким образом на позицию i мы выбираем элемент из числа всех тех, которые могут стоять на этой позиции, то есть мы выбираем правильно.

Ну и в конце, когда остается k элементов в куче и по индукции все первые (n-k) элементов стоят на правильных местах мы просто берем минимум из кучи и записываем в соответствующую ячейку массива.