# Алгоритмы и модели вычислений. Задание 9: сортировки

Сергей Володин, 272 гр.

задано 2014.04.10

## (каноническое) Задача 38

## (каноническое) Задача 39

## (каноническое) Задача 40

(Модифицируем алгоритм merge sort. Задачу давал Пименов.)

n−размер массива.

Количество инверсий в массиве при $n \leqslant 1$ равно 0.

Пусть посчитаны инверсии для двух половинок, а также половинки отсортированы.

$c \stackrel{\text{def}}{=} \frac{n}{2}, A = [1, c], B = [c+1, n]$ −множества индексов половинок. Тогда посчитаны инверсии для таких пар индексор $i \neq j$ :

- $i, j \in A$−элементы до середины
- $i, j \in B$−элементы после середины

Тогда осталось посчитать инверсии для пар, индексы которых принадлежат разным множествам. Очевидно, что сортировка половинок не изменила количество инверсий для таких пар, т.к. порядок элементов такой пары в массиве не изменился после сортировки.

Будем производить слияние двух упорядоченных половинок $l = (l_1, \ldots, l_c)$ и $r = (r_1, \ldots, r_{n-c})$, при каждом выборе элемента (на каждой итерации цикла) будем считать инверсии текущего элемента с последующими из другого множества (так, заметим, обойдем все необходимые пары). Полученные результаты будем суммировать (это возможно, т.к. каждая пара рассматривается один раз).

Пусть получены $k-1$ элементов итогового массива, выбраны i элементов из первой половинки, j из правой. Инверсии для $k-1$ также посчитаны.

k-й равен $l_{i+1}$, если $l_{i+1} < r_{j+1}$ или $r_{j+1}$ иначе.

Найдем число инверсий для каждого случая:

- Если добавляем $l_{i+1}$, то последующими элементами из другого множества (из другой половинки) могут быть $r_{j+1} < r_{j+2} < \ldots$. Но (условие случая) $l_{i+1} < r_{j+1} < \ldots \Rightarrow$инверсий с последующими нет (т.к. "более левый" в исходном массиве элемент меньше правых из другого множества).
- Если добавляем $r_{j+1}$, то последующими элементами из другого множества будут $l_{i+1} < l_{i+2} < \ldots$. Но (условие случая) $r_{j+1} < l_{i+1} < \ldots$. Получаем, что число инверсий равно количеству оставшихся элементов из левой половинки, т.е. $c - i$ (т.к. "более правый" в исходном массиве элемент меньше такого количества "более левых" из другого множества).

Очевидно, что время работы такого алгоритма равно времени работы merge sort, т.е. $T \in O(n \log n)$