1. k-ая порядковая статистика

k-ой порядковой статистикой набора элементов линейно упорядоченного множества X называется такой его элемент, который является k-ым элементом набора в порядке сортировки: $X = x_1, x_2, ..., x_n$; Y:$\{y_i \in X\}$, где $y_1 \leq y_2 ... \leq y_n$

Поиск k-ой порядковой статистики:

Будем использовать процедуру рассечения массива элементов из алгоритма сортировки QuickSort. Пусть нам надо найти k-ую порядковую статистику, а после рассечения опорный элемент встал на позицию m. Возможны три случая:

1) k = m порядковая статистика найдена

2) k < m. Рекурсивно ищем k-ую порядковую статистику в первой половине массива.

3) k > m. Рекурсивно ищем (k - m - 1)-ую статистику во второй половине массива.

2. Теоремы о сортировках

Сортировка – некая перестановка, которая соответствует свойствам упорядоченности.

1.ТЕОРЕМА о сортировках,в которых меняют местами только соседние элементы:

Среди таких сортировок не может быть тех, что работают быстрее, чем за $O(n^2)$

Доказательство:

Рассмотрим неупорядоченный набор данных: $a_1, a_2, ..., a_n$ Пусть "неправильные"пары (инверсия) - те, в которых упорядоченность не соблюдается. Количество таких инверсий: $O(\frac{n(n-1)}{2}) = O(\frac{n^2-n}{2}) = O(n^2)$

2.ТЕОРЕМА о сортировках выбором

Среди всех сортировок, использующих операцию swap, сортировка выбором (по количеству swap'ов) наиболее эффективна

Замечание: задачу поиска нельзя решить быстрее, чем за $O(nlogn)$

3.ТЕОРЕМА о нижней оценке для сортировки сравнениями

Не существует сортировки, основанной на сравнениях и работающей быстрее, чем за $O(nlogn)$

Док-во:

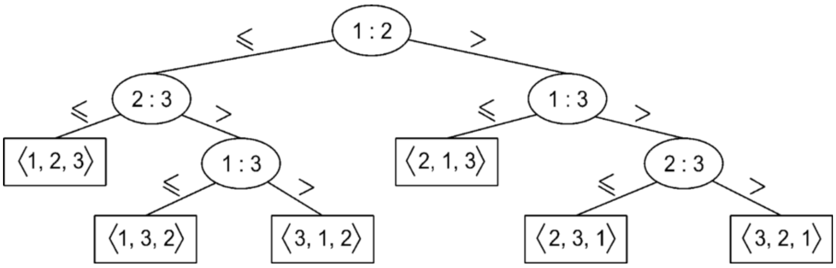

Рис. 1. Пример дерева для алгоритма сортировки трех элементов. Внутренний узел, помеченный как i : j , указывает сравнение между a_i и a_j. Лист, помеченный перестановкой $\{\pi(1), \pi(2), \ldots, \pi(n)\}$, указывает упорядочение $a_{\pi(1)} \leq a_{\pi(a)} \leq ... \leq a_{\pi(n)}$.

Любому алгоритму сортировки сравнениями можно сопоставить дерево. В нем узлам соответствуют операции сравнения элементов, ребрам – переходы между состояниями алгоритма, а листьям – конечные перестановки элементов (соответствующие завершению алгоритма сортировки). Необходимо доказать, что высота такого дерева для любого алгоритма сортировки сравнениями не меньше чем $O(nlogn)$,где n – количество перестановок.

Ограничимся рассмотрением сортировки перестановок n элементов. При сравнении некоторых двух из них, существует два возможных исхода ($a_i \leq a_j$) и $a_i > a_j$, значит, каждый узел дерева имеет не более двух сыновей. Всего существует n! различных перестановок n элементов, значит, число листьев нашего дерева не менее n! (в противном случае некоторые перестановки были бы не достижимы из корня, а, значит, алгоритм не правильно работал бы на некоторых исходных данных).

Докажем, что двоичное дерево с не менее чем n! листьями имеет глубину $O(nlogn)$. Легко показать, что двоичное дерево высоты h имеет не более чем 2^h листьев. Значит, имеем неравенство $n! \leq l \leq 2^h$, где l – число листьев.
Прологарифмировав его, получим: $h \geq log_2 n! = log_2 1 + log_2 2 + ... + log_2 n > \frac{n}{2} log_2 \frac{n}{2} = \frac{n}{2}(log_2 n - 1) = O(nlogn)$

3. Жадный алгоритм

- это алгоритм, который на каждом шагу делает локально наилучший выбор в надежде, что итоговое решение будет оптимальным => максимум и минимум не обязательно достижимы.

Принцип жадного выбора:
Говорят, что к оптимизационной задаче применим принцип жадного выбора, если последовательность локально оптимальных выборов даёт глобально оптимальное решение. В типичном случае доказательство оптимальности следует такой схеме:
1) Доказывается, что жадный выбор на первом шаге не закрывает пути к оптимальному решению: для всякого решения есть другое, согласованное с жадным выбором и не хуже первого.
2) Показывается, что подзадача, возникающая после жадного выбора на первом шаге, аналогична исходной.
3) Рассуждение завершается по индукции.

Рассмотрим задачу: Пусть имеется некоторое количество монет и с их помощью мы должны произвести размен некоторой суммы денег x. Пусть сумма будет равна 12, тогда существует жадный размен и оптимальный, которые не совпадают.

Таблица 1. Задача о размене

Монеты	Жадный размен	Оптимальный размен
10	10	6
6	1	6
1	1	–

То есть жадный алгоритм действует так: на i-ом шаге он берёт ту монету, которая является наибольшей из имеющихся и которая не больше суммы на i-ом шаге.
Например, задача о непрерывном рюкзаке тоже не при всяком наборе входных данных решается жадным алгоритмом. Эта задача аналогична задаче о размене.

Жадный алгоритм будет срабатывать эффективно при любых наборах данных, если в задаче имеется матроид.