Матосновы алгоритмов

Мастера конспектов
на основе лекций А. С. Охотина и А. В. Тискина

22 января 2020 г.

Основные моменты.

Содержание

1 Лекция 1.

1.1 Быстрая сортировка.

Алгоритм 1. *Быстрая сортировка. Выбираем опорный элемент, с которым сравниваем все остальные элементы (на это уходит линейное время). Затем рекурсивно работаем с тем, что справа от него и слева от него.*

Теорема 1. *Если все элементы массива различны и опопрный элемент выбирается случайно, то среднее время работы алгоритма* - $\Theta(n \log n)$.

Доказательство. Время работы пропорционально числу сравнениий между элементами. Расматриваем два элемента y_i и y_j, $i < j$, тогда они сравниваются только, если выбран один из них в качестве опорного. Если будет выбран какой-то y_k, $i < k < j$, то они никогда больше не будут сравнены, если что-то на отрезке не между ними - плевать, относительно отрезка между ними ничего не поменялось. Тогда среднее количество сравнений между этими элементами:

$$\frac{2}{j-i+1}.$$

Тогда всего среднее количество сравнений:

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} = \sum_{i=1}^{n-1} \sum_{k=1}^{n-1} \frac{2}{k+1} = O(n \log n).$$

Последняя оценка получается из

$$\sum_{k=1}^{n} \frac{1}{k} \approx \int_1^n \frac{1}{x} dx = \ln n.$$

□

1.2 Сортировка кучей.

Алгоритм 2. *Сортировка кучей. Для начала, мы строим дерево: записываем по порядочку все вершины так, что у вершины x_i потомки - x_{2i} и x_{2i+1}. Затем начинаем на все вершины, кроме висячих смотреть и делать вот что: если она меньше потомка, то поменяем с ним (если меньше обоих, то с меньшим). Так доведём её до куда сможем, и продолжим рпссмотрение для оставшихся невисячих вершин (в изначальном дереве). Так сверху окажется наименьшая вершина, вынесем её, затем - по индукции.*

Утверждение 1. Работает за $O(n \log n)$ (построение дерева - $O(\log n)$, вынесение вершин - $O(n)$), можно разогнать оценку до $2n$.

1.3 Скорость сортировки.

Теорема 2. *Всякий алгоритм сортировки, основанный на сравнении, требует $\Omega(n \log n)$ операций сравнения.*

Доказательство. Построим дерево того, как мы спускаемся к определению последовательности, его высота ограничивается $\log_2 n!$, оценим факториал $\left(\frac{n}{e}\right)^n < n! < n^n$, а после - оценим через это логарифм $n \log_2 n - O(n)$. □

Алгоритм 3. *Сортировка подсчётом. Если у нас есть массив из конечного обозримого количества типов элементов, можно сначала посчитать количество первого, затем количество второго, и так далее. Время работы - $O(n+k)$, где k - количество типов, n - количество переменных.*

Алгоритм 4. *Поразрядная сортировка. Сортируем числа сначала по первому разряду, затем по второму, и так далее... Время работы: $O(l(n+k))$, где сравниваются строки длины l, алфавит из k символов.*

1.4 Нахождение i-го по величине элемента массива.

Алгоритм 5. *Нахождение i-го элемента. Делим массив на пятёрки подряд идущих элементов (возможно, последняя пятёрка будет неполной). Теперь в каждой пятёрке выделяем медианы, и смотрим на медиану медиан. Сделаем её опорным элементом и как в быстрой сортировке, раскидаем всё по сторонам. Если этот элемент под номером i, то мы его нашли, иначе - действуем рекурсивно с одной из сторон. Время работы - линейное.*

1.5 Метод динамического программирования.

Задача 1. Имеется стержень длины n. Продав стержень длины i, можно выручить p_i денежных единиц. Как выгоднее всего распилить имеющийся стержень?

Алгоритм 6. *Начинаем с первого, и делаем полный перебор. Говнище*

Алгоритм 7. *Жадный алгоритм. Отпиливаем самый дорогой кусок, затем опять самый дорогой из возможных, и так далее. Не самый оптимальный.*

Алгоритм 8. *Метод динамического программмрования. Суть этого метода такова. Пусть на каждом шаге надо сделать выбор (принять решение). Известно, что какой-то выбор приводит к оптимальному результату. Этому выбору соответствует некий набор подзадач. Тогда сперва находятся ответы для всех подзадач данной задачи, возникающих при различном выборе, после чего, имея все эти ответы перед глазами, можно будет в каждом случае сделать наилучший выбор.*

Пример(ы) 1. Пусть стержень длины 0 не стоит нисколько. Для j от 1 до n пока не найдено никаких способов продать стержень, для всякой длины отрезаемого куска, сложим его цену с выручкой за остаток. Если так можно выручить больше известного, то цена стержня длины j улучшается, и так рекурсивно мы дойдём до получения цены за весь стержень.