Тема 6. Некоторые алгоритмы

Лекция 19. Исследование сортировок

Рассматриваем сортировки с точки зрения
— гарантированной трудоемкости
— средней трудоемкости
— дополнительной памяти

Любая рекурсивная процедура захватывает дополнительную память пропорционально глубине рекурсии.

Примитивные сортировки — $O(n^2)$, доп. память не требуется

с использованием сравнений:

heapsort — гарантировано $O(n \log n)$, доп. память не требуется

quicksort — гарантировано $O(n^2)$, средняя $O(n \log n)$, доп. память — глубина рекурсии $O(\log n)$

mergesort — гарантировано $O(n \log n)$, доп. память $O(n)$

+ варианты гарантировано $O(n \log^2 n)$, доп. память не требуется

+ варианты гарантировано $O(n \log n)$, доп. память не требуется, но жуткая константа в $O()$:)

без использования сравнений

countsort — гарантировано $O(n)$, доп. память $O(n + 2^p)$, p — число бит в представлении чисел

radixsort — гарантировано $O(np)$, p — число бит в представлении чисел

Теорема. Любая сортировка массива из n элементов, использующая попарные сравнения и перестановки элементов, не может иметь гарантированную трудоемкость T с оценкой лучше по порядку, чем $O(n \log n)$, т.е. существует $c = \text{const}$, что для любой такой сортировки $\geq c(n \log n)$.

Доказательство. Идея:

Дерево алгоритма (вершины — сравнения, ребра — переход с другой паре элементов).

В этом дереве $n!$ концевых вершин (листьев) так как каждая перестановка на входе должна в итоге прийти к отсортированному состоянию. Если — глубина дерева, то получаем $2^h \geq n!$.

Формула Стирлинга $n! \approx \sqrt{2\pi n}(n/e)^n$.

Средняя трудоемкость быстрой сортировки.

Теорема. Быстрая сортировка имеет среднюю трудоемкость $O(n \log n)$ при условии равномерного распределения позиции разделения массива на каждом шаге.

Доказательство.

Обозначим через T_n среднюю трудоемкость сортировки массива из n элементов. Эта трудоемкость складывается из операций по разделению массива на две части (что требует порядка n операций) и трудоемкости сортировок двух подмассивов (длиной k и $n - k$, если предположить, что разделение произошло в позиции k). Принимая во внимание наше определение средней трудоемкости, получаем оценку

$$T_n \leq cn + \frac{1}{n-1} \sum_{k=1}^{n-1} (T_k + T_{n-k}) = cn + \frac{2}{n-1} \sum_{k=1}^{n-1} T_k,$$

где c — некоторая константа.

Докажем, что можно выбрать константы a, b, так, что будет выполнено неравенство

$$T_n \le an\log_2 n + b \quad \forall n > 0.$$

Доказательство проведем по индукции. База индукции ($n = 1$) очевидна. Так как массив из одного элемента сортировать не надо, то мы можем считать, что требуемое неравенство выполнено с $b = 1$ (эта единственная операция нужна, чтобы при выполнении алгоритма проверить равенство $n = 1$).

Пусть указанное неравенство верно для $k \le n-1$, докажем его справедливость для $k = n$. Используя предположение индукции, имеем цепочку неравенств

$$\sum_{k=1}^{n-1} T_k \le \sum_{k=1}^{n-1}(ak\log_2 k + b) \le a\sum_{k=1}^{n/2} k\log_2 k + a\sum_{k=n/2+1}^{n-1} k\log_2 k + b(n-1) \le$$

$$\le a\log_2\frac{n}{2}\sum_{k=1}^{n/2} k + a\log_2 n\sum_{k=n/2+1}^{n-1} k + b(n-1) = a\log_2 n\sum_{k=1}^{n-1} k - a\sum_{k=1}^{n/2} k + b(n-1) =$$

$$= \frac{an(n-1)}{2}\log_2 n - \frac{an(n+2)}{8} + b(n-1).$$

Подставим получившуюся оценку в исходное неравенство

$$T_n \le cn + \frac{2}{n-1}\sum_{k=1}^{n-1} T_k \le cn + an\log_2 n - \frac{an(n+2)}{4(n-1)} + 2b \le an\log_2 n + b + [b + n(c - a/4)].$$

Требуемое неравенство будет выполнено, если при подходящем выборе a и b мы сможем сделать неположительным последнее слагаемое. Учитывая, что база индукции выполнена для $b = 1$, получаем для a неравенство

$$a \ge \frac{4b}{n} + 4c \ge 2 + 4c, \quad n > 1.$$

Таким образом, требуемая оценка выполнена с $b = 1$, $a = 4c + 2$, где константа c определяется выбранным алгоритмом разделения массива на две части. Теорема доказана.

Будет ли в реальности выполнено условие равновероятной позиции разделения ? Это зависит от подхода к выбору разделяющего элемента.

Топологическая сортировка

Есть частично-упорядоченное множество. Надо его выстроить в порядке, когда для всех сравнимых элементов выолнены условия “упорядоченности”. Приложение — планирование работ, некоторые операции должны быть выполнены раньше других, а для некоторых операций взаимный порядок произволен.

Топологическая сортировка — обход графа в глубину.

Граф: вершины — это элементы, ребра — отношения порядка (от меньшего к большему).

Берем произвольную вершину и запускаем поиск в глубину. В момент перекрашивания вершины в черный цвет, добавляем ее в начало списка упорядоченных вершин. Если вернулись в начальную вершину, но есть еще непройденные вершины, то выбираем произвольную непройденную и продолжаем процесс с нее.

Следствие. При сортировке линейно упорядоченного множества размера N топологическая сортировка имеет трудоемкость $O(N^2)$. (Есть большая аналогия с методом перестановки максимума)