Поиск k-ой порядковой статистики

Определение:

k**-ой порядковой статистикой** набора элементов линейно упорядоченного множества называется такой его элемент, который является k-ым элементом набора в порядке сортировки

Содержание

Модификация QuickSort

Описание алгоритма

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. Возможно три случая:

- **k = m**. Порядковая статистика найдена.
- **k < m**. Рекурсивно ищем k-ую статистику в первой половине массива.
- **k > m**. Рекурсивно ищем $(k - m - 1)$-ую статистику во второй половине массива.

Код алгоритма

Ниже представлен код представленного алгоритма. При реализации, однако, вместо рекурсивных вызовов изменяются границы поиска статистики во внешнем цикле. В коде считаем, что процедура **partition** принимает массив и границы отрезка, который будет рассечён (причём правая граница отрезка не включается) и возвращает индекс опорного элемента. Также, считается, что массив индексируется с нуля.

```
int findOrderStatistic(int[] array, int k) {
  int left = 0, right = array.length;
  while (true) {
    int mid = partition(array, left, right);

    if (mid == k) {
      return array[mid];
    }
    else if (k < mid) {
      right = mid;
    }
    else {
      k -= mid + 1;
      left = mid + 1;
```

```
    }
  }
}
```

Анализ времени работы

Аналогично QuickSort, может возникнуть такой же худший случай (процедура **partition** возвращает каждый раз левую или правую границу рассматриваемой части), при котором время работы составит $\Omega(n^2)$. Однако, если считать, что **partition** возвращает все элементы рассматриваемого отрезка с равной вероятностью, то можно оценить матожидание времени работы как $O(n)$.

Будем оценивать количество сравнений. При поиске статистики в массиве размера n функция **partition** (точнее, одна из распространённых вариаций) совершает не более $n-1$ сравнений. Далее, в зависимости от k выбирается левая или правая половины (или вообще алгоритм завершает работу). Оценку проводим сверху, то есть, будем считать, что каждый раз выбирается большая половина.

$$T(n) \le \frac{1}{n}\sum_{k=1}^{n}\left(T\left(\max\{k-1; n-k\}\right) + n - 1\right) =$$
$$= n - 1 + \frac{1}{n}\sum_{k=1}^{n} T(\max\{k-1; n-k\}) = n - 1 + \frac{2}{n}\sum_{k=\lfloor n/2 \rfloor}^{n-1} T(k)$$

Предположим, что $T(k) \le ck$ для некоторой константы c и всех $k < n$ (будем доказывать оценку по индукции). Тогда верно неравенство:

$$T(n) = n - 1 + \frac{2}{n}\sum_{k=\lfloor n/2 \rfloor}^{n-1} ck$$

Преобразуем сумму из правой части равенства по формуле суммы арифметической прогрессии и оценим преобразованное выражение:

$$\sum_{k=\lfloor n/2 \rfloor}^{n-1} ck = \frac{1}{2}\left(\left\lceil \frac{n}{2} \right\rceil - 1\right)\left(c\left\lfloor \frac{n}{2} \right\rfloor + c(n-1)\right) \le \frac{c}{2}\left(\frac{n+1}{2} - 1\right)\frac{3n-2}{2} = c\,\frac{n-1}{4}\,\frac{3n-2}{2}$$

Воспользуемся полученной оценкой для оценки исходного выражения. Также, предположим, что $c \ge 4$:

$$T(n) \le n - 1 + \frac{2c}{n}\,\frac{n-1}{4}\,\frac{3n-2}{2} = n - 1 + c\,\frac{n-1}{2n}\,\frac{3n-2}{2} \le \frac{c}{4}(n-1) + \frac{c}{4}\left(\frac{n-1}{n}(3n-2)\right) \le$$
$$\le \frac{c}{4}(n - 1 + 3n - 2) = \frac{c}{4}(4n-3) \le cn$$

Для довершения доказательства необходима проверка базы индукции, но она тривиальна: для выборки порядковой статистики из одного элемента сравнений не требуется: $T(1) = 0 < 4$. Итого, мы доказали, что $T(n) \le 4n$, следовательно, $T(n) = O(n)$

Ссылки

- Selection algorithm — Wikipedia (http://en.wikipedia.org/wiki/BFPRT)
- Donald Knuth. The Art of Computer Programming, Volume 3: Sorting and Searching, Third Edition. Addison-Wesley, 1997. ISBN 0-201-89685-0. Section 5.3.3: Minimum-Comparison Selection, pp.207–219.

Источник — «http://neerc.ifmo.ru/wiki/index.php?title=Поиск_k-ой_порядковой_статистики&oldid=36923»

- Эта страница последний раз была отредактирована 12 мая 2014 в 22:48.