Курс kiev-clrs – Лекция 6. Медианы и порядковые статистики

Олег Смирнов

16 мая 2009 г.

Содержание

1 Цель лекции

- Дать понятие i-й порядковой статистики и медианы массива
- Алгоритм выбора порядковой статистики за линейное ожидаемое время
- Алгоритм выбора за линейное время в наихудшем случае

2 Введение

Профессор работает консультантом в нефтяной компании, которая запланирует провести магистральный трубопровод от восточного до западного края нефтяного месторождения с n скважинами. От каждой скважины к магистральному трубопроводу кратчайшим путем проведены рукава. Каким образом профессор может выбрать оптимальное расположение трубопровода (т.е. такое, при котором общая длина всех рукавов была бы минимальной) по заданым координатам скважин (x, y)?

Легко видеть, что в случае чётного количества скважен n, трубопровод можно провести в любом месте при условии, что по обе его стороны (с севера и с юга) будет равное количество скважин. Множества скважин можно представить в виде массива их y координат, отсортированного по возрастанию элементов. Условие будет достигнуто, если трубопровод проходит между скважинами с номерами $n/2$ и $n/2 + 1$ в указанном массиве.

В случае нечётного количества, можно найти скважину, координата y которой лежит "в центре" массива, т.е. с порядковым номером $\frac{n+1}{2}$. Если провести трубопровод через указанную скважину, то растояние до неё будет равно нулю и задача сведется к предыдущему случаю.

3 Порядковые статистики

Будем называть i-й порядковой статистикой множества, состоящего из n элементов, i-й элемент в порядке возратания. Например, минимум такого множества – это первая порядковая статистика ($i = 1$), а его максимум – это n-я порядковая статистика. Медиана неформально обозначает середину множества. Если n нечётное, то медиана единственная,

и её индекс равен $i = \frac{n+1}{2}$; если же n чётное, то медианы две, и их индексы равны $i = n/2$ и $i = n/2 + 1$. Таким образом, независимо от чётности n, медианы располагаются при $i = \lfloor \frac{n+1}{2} \rfloor$ (нижняя медиана) и $i = \lceil \frac{n+1}{2} \rceil$ (верхняя медиана).

Наивный алгоритм выбирает минимум и максимум массива за $\Theta(n)$, а произвольную k-ую статистику – за $\Theta(n \lg n)$ итераций.

4 Алгоритм выбора за линейное ожидаемое время

Идея: использовать процедуру разбиения алгоритма Quicksort, на каждом шаге рекурсии уходя в ту половину массива (левую или правую), где содержится искомый элемент.

```
RANDOM_SELECT(A, p, q, i)
1  if p = q
2     then return A[p]
3          r ←Random_Partition(A, p, q) //индекс опорного элемента
4          k ← r − p + 1 //k = rank(A[r])
5          if i = k
6             then return A[r]
7          elseif i < k
8             then return Rand_Select(A, p, r − 1, i)
9             else  return Rand_Select(A, r + 1, q, i − k)
```

Переменная k содержит ранг опорного элемента r:

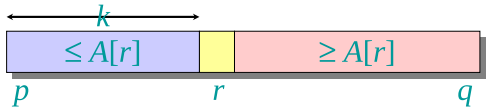

Рис. 1: Разбиение массива

Для выбора $i = 7$ порядковой статистики, алгоритм сгенерирует разбиение относительно опорного элемента 6:

На следующем шаге будет рекурсивно вызван алгоритм для правой (большей) части массива, т.к. ранг опорного элемента $k = 4$ меньше 7:

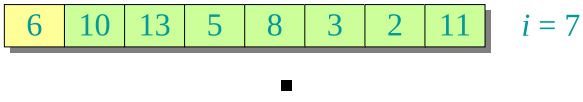

Рис. 2: Пример работы: опорный элемент

Рис. 3: Пример работы: шаг рекурсии

5 Анализ алгоритма

5.1 Лучший случай

В случае, если алгоритм на каждом шаге генерирует разбиение строго пополам или в любом другом константом соотношении, допустим 1:10

$$T(n) \leqslant T(\frac{9}{10}n) + \Theta(n)$$

Решая неравенство по основному методу, получим $T(n) \leqslant \Theta(n)$

5.2 Худший случай

Если алгоритм всегда разбивает массив в соотношении $0\,..\,n-1$, время работы квадратично:

$$T(n) = T(n-1) + \Theta(n) = \Theta(n^2)$$

Попасть в наихудший случай можно только тогда, когда на каждом шаге генератор случайных чисел генерирует “плохое” разбиение.

5.3 Средний случай

Пусть $T(n)$ – случайная величина, показывающая время выполнения алгоритма Random_Select для массива размера n различных элементов. Необходимо оценить математическое ожидание $E[T(n)]$.

Пусть X_k – индикаторная случайная величина для $k = 0\,..\,n-1$, равная

$$X_k = \begin{cases} 1, \text{если сгенерировано разбиение } k : n-k-1 \\ 0, \text{иначе} \end{cases}$$

Тогда время работы можно оценить аналогично алгоритму Quicksort, рассмотрев совокупность случаев:

$$T(n) \leqslant \begin{cases} T(max\{0, n-1\} + \Theta(n), \text{ для разбиения } 0 : n-1 \\ T(max\{1, n-2\} + \Theta(n), \text{ для разбиения } 1 : n-2 \\ .. \\ T(max\{n-1, 0\} + \Theta(n), \text{ для разбиения } n-1 : 0 \end{cases}$$

$$= \sum_{k=0}^{n-1} X_k(T(max\{k, n-k-1\}) + \Theta(n))$$

$$E[T(n)] = \sum_{k=0}^{n-1} E[X_k(T(max\{k, n-k-1\}) + \Theta(n))] =$$

$$= \frac{1}{n} \sum_{k=0}^{n-1} E[T(max\{k, n-k-1\}] + \Theta(n)) =$$

т.к. $T(max\{k, n-k-1\})$ и X_k – независимы

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

Неравенство решается методом подстановки.

$$E[T(n)] \leqslant \frac{2}{n} \sum_{k=\lfloor n/2 \rfloor}^{n-1} E[T(k)] + \Theta(n) \leqslant$$

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

$$= \frac{2c}{n} \sum_{k=\lfloor n/2 \rfloor}^{n-1} k + \Theta(n) \leqslant$$

$$\leqslant \frac{3}{4}cn + \Theta(n) =$$

$$= cn - (\frac{1}{4}cn - \Theta(n))$$

Итак, алгоритм Random_Select работает за $\Theta(n)$ в среднем случае и за $\Theta(n^2)$ в худшем случае. Вероятность получить худший случай уменьшается с ростом размера массива n.

6 Алгоритм выбора за линейное время в наихудшем случае

Идея алгоритма: генерировать "хороший" опорный элемент, используя детерминированную процедуру Partition

1. Разбиение входного массива на пять групп по $\lfloor n/5 \rfloor$ элементов в каждой и одну группу размером $n \mod 5$ (возможно пустую)
2. Каждая из пяти групп размера $\lfloor n/5 \rfloor$ сортируется методом вставки, а затем в каждой выбирается медиана
3. Путем рекурсивного вызова алгоритма Select определяется медиана x множества из $\lceil n/5 \rceil$ медиан из пункта 2
4. С помощью алгоритма Partition генерируется разбиение исходного массива относительно медианы x
5. Пусть $k = rank(x)$. Алгоритм ветвления аналогичен предыдущему случаю:

```
1  if i = k
2     then return x
3  elseif i < k
4     then return Select(i-й элемент в меньшей половине разбиения)
5  elseif i > k
6     then return Select(i − k-й элемент в большей половине)
```

Очевидно, что время работы алгоритма будет слагаться из частей:

1. $\Theta(n)$
2. $\Theta(n)$
3. $T(n/5)$
4. $\Theta(n)$
5. $T(a/5)$, где a – некая константа. В случае, если константа будет равна $4/5$, время работы алгоритма будет лог-линейное. Если константа строго меньше $4/5$ – время работы будет линейно от размера входного массива.

7 Анализ алгоритма

Представим графически массив элементов в виде пяти групп и одной меньшей:

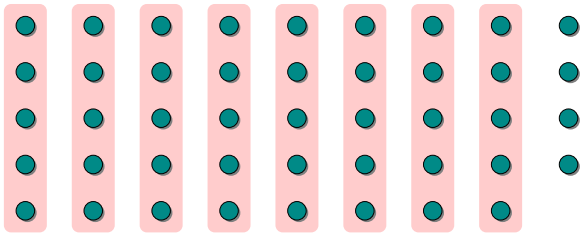

Рис. 4: Анализ: шаг 1

В каждой группе можно выделить медианы (обозначена желтым). Отношения между элементами групп и медианами обозначены стрелками. Стрелка направлена от меньшего элемента к большему:

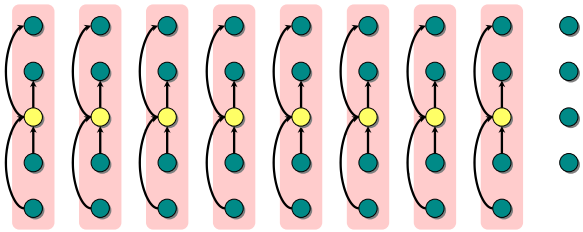

Рис. 5: Анализ: шаг 2

Медиана медиан (элемент x) обозначена красным:

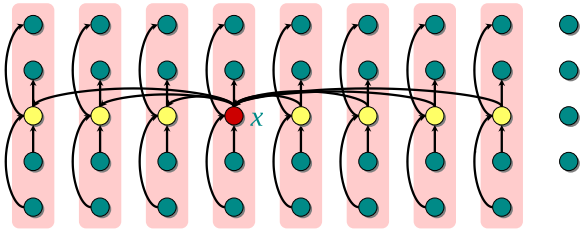

Рис. 6: Анализ: шаг 3

Колличество групп элементов, находящихся слева от элемента x будет равно $\lfloor \lfloor n/5 \rfloor /2 \rfloor$. По отношению транзитивности легко заметить, что

количество элементов, $\leqslant x$, будет равно $3\lfloor\lfloor n/5\rfloor/2\rfloor$. На рисунке выделены синим цветом:

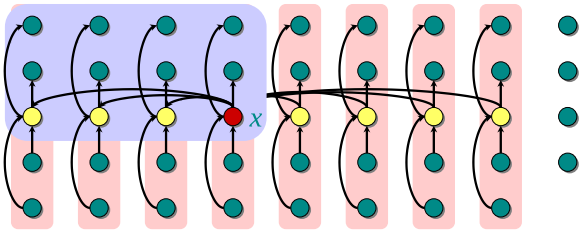

Рис. 7: Анализ: шаг 4

Аналогично для элементов $\geqslant x$:

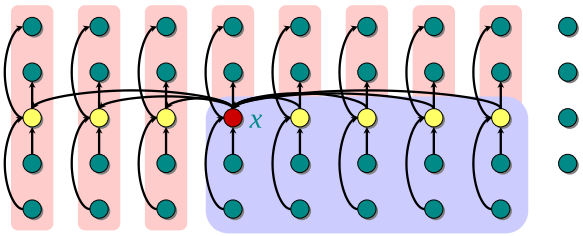

Рис. 8: Анализ: шаг 5

Таким образом, какое бы не было сгенерировано разбиение, в каждой из частей массива по обе стороны от медианы медиан x будет находится не менее чем $3\lfloor n/10\rfloor$ элементов. Следовательно по другую сторону – не более $7\lfloor n/10\rfloor$ элементов. Можно избавиться от "пола" в формуле, положив, что для $n \geqslant 50$: $3\lfloor n/10\rfloor \geqslant n/4$. Таким образом, если в каждой из частей находится не менее $n/4$ элементов, значит в большей – не более $\frac{3}{4}n$ элементов.

Рекуррентность решается по методу подстановки:

$$T(n) \leqslant T(n/5) + T(\frac{3}{4}n) + \Theta(n)$$

$$T(n) \leqslant \frac{c}{5}n + \frac{3}{4}cn + \Theta(n) =$$

$$= \frac{19}{20}cn + \Theta(n) =$$

$$= cn(-\frac{1}{20}cn + \Theta(n))$$