Курс kiev-clrs – Лекция 4. Сортировка Quicksort, рандомизированные алгоритмы

Олег Смирнов

4 апреля 2009 г.

Содержание

1 Цель лекции

- Рассмотреть сортировку Хоара
- Рассмотреть рандомизированные алгоритмы
- Проанализировать алгоритм Хоара в классическом и в рандомизированном варианте

2 Алгоритм Хоара: Quicksort

Алгоритм входит в Top10 вместе с методом Монте-Карло, симплекс-методом и быстрым преобразованием Фурье.

Алгоритм вида "Разделяй и властвуй сортировка делается "на месте так же как сортировка вставкой в отличии от сортировки слиянием. Шаги алгоритма:

- Divide: деление массива на два подмассива относительно опорного элемента (pivot) x: элементы левого подмассива $\leqslant x$, правого $\geqslant x$
- Conquer: рекурсивная сортировка двух подмассивов
- Combine: тривиально

Проще говоря, Quicksort – это рекурсивное деление, тогда как Mergesort – рекурсивное слияние.

Ключевой частью алгоритма является процедуры Partition, которая выполняется за линейное время ($O(n)$):

```
PARTITION(A, p, q) //A[p..q]
1  x ← A[p] //pivot
2  i ← p
3  for j ← p + 1 to q
4        do if A[j] ⩽ x
5              then i ← i + 1
6                   обменять A[i] ⟺ A[j]
7  обменять A[p] ⟺ A[j]
8  return i
```

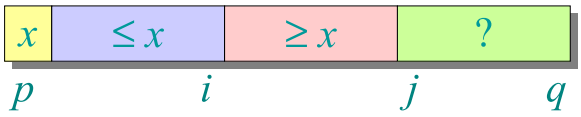

Рис. 1: Инвариант цикла

Доказательство инварианта (рис. 1):

- Инициализация: в начале цикла инвариант выполняется, так как $i = p$ и $j = p + 1$
- Сохранение: если на очередном шаге $A[j] \geqslant x$, инвариант выполняется и счетчик j увеличивается на единицу
- Завершение: если $A[j] \leqslant x$, инвариант выполняется через обмен элемента $A[j]$ с $A[i]$ и сдвига границы левой части массива $i = i + 1$

Алгоритм весьма прост для понимания. Время выполнения процедуры равно $O(n)$ для массива из n элементов.

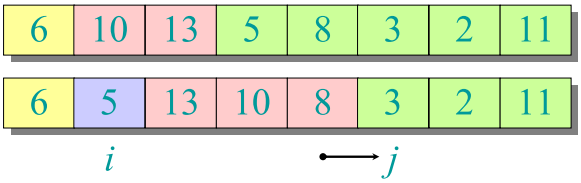

Рис. 2: Пример работы

Пример работы процедуры (рис. 2):

- шаге 0: i на 6, j на 10
- шаге 1: обменеваются местами 10 и 5, j на 8
- шаге 2: обменеваются 13 и 3, i на 3, j на 2
- шаге 2: обменеваются 10 и 2, i на 2, j на 11
- на последнем шаге опорный элемент перемещается на середину (вместо 2)

В конце работы элементы меньше или равные опорному находятся слева, остальные – справа.

Алгоритм QuickSort:

```
QUICKSORT(A, p, q)
1  if p < q
2     then r ← Partition(A, p, q)
3          Quicksort(A, p, r-1)
4          Quicksort(A, r+1, q)
```

Начальные параметры вызова: Quicksort(A, 1, n)

Один из способов оптимизации Quicksort – использования отдельной процедуры для маленьких массивов, например для $n \leqslant 5$ (например прямое сравнение). Также применяют хвостовую оптимизацию рекурсии.

3 Анализ Quicksort

Предположим, что все элементы различны, т.к. в случае дублирующихся элементов алгоритм работает более эффективно.

3.1 Наихудший случай

Легко видеть, что наихудшим образом алгоритм будет себя вести, если подпрограмма, выполняющая разбиение, порождает одну подзадачу с $n-1$ элементов, а вторую – с 0 элементов. Это происходит в том случае, если все элементы массива больше либо меньше выбранного опорного элемента.

$$\begin{aligned} T(n) = \\ = T(0) + T(n-1) + \Theta(n) \\ = \Theta(1) + T(n-1) + \Theta(n) \\ = T(n-1) + \Theta(n) \\ = \Theta(n^2) \text{ (арифметическая прогрессия)} \end{aligned}$$

Дерево рекурсии для случая $T(n) = T(0) + T(n-1) + cn$ (рис. 3).

Высота дерева $h = n$. Сумма правой ветви – $\Theta(\sum_{k=1} ck) = \Theta(n^2)$, левых ветвей – $\Theta(1)n$. Суммарное время работы в таком случае равно $T(n) = \Theta(n) + \Theta(n^2) = \Theta(n^2)$

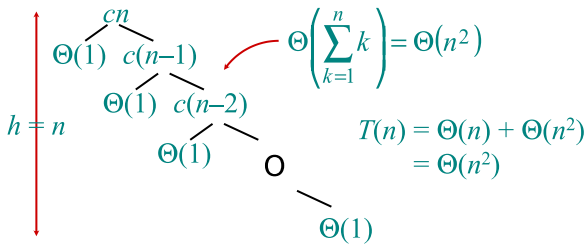

Рис. 3: Дерево наихудшего случая

3.2 Наилучший случай

Возникает, когда процедура делит массив на две равные части: $n/2$ и $n/2$. Время работы оценивается рекуррентностью:

$$T(n) \leqslant 2T(\frac{n}{2}) + \Theta(n)$$

Согласно второму случае основного метода, ответ $T(n) = O(n \lg n)$

3.3 Средний случай

Рассмотрим "почти средний"случай. Предположим, что процедура всегда разбивает массив на части $\frac{1}{10}$ к $\frac{9}{10}$

Получим рекуррентость: $T(n) = T(\frac{1}{10}n) + T(\frac{9}{10}n) + \Theta(n)$. Построим дерево, явно вводя константу cn вместо $\Theta(n)$.

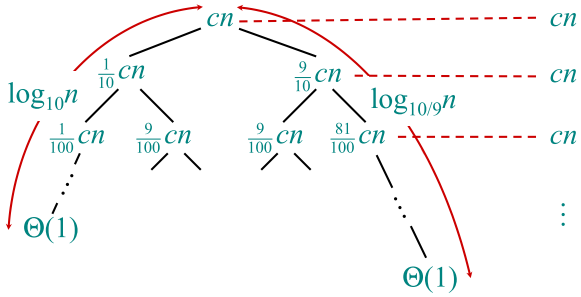

Рис. 4: Дерево "почти среднего"случая

Время работы каждого уровня дерева будет $\leqslant cn$. Глубина дерева будет разной для левой и правой ветви (рис. 4). В левой части $\log_{10} n$, в

правой $\log_{\frac{10}{9}} n$

$$\begin{aligned} T(n) \leqslant \\ \leqslant cn \log_{\frac{10}{9}} n + \Theta(n) = \\ = O(n \lg n) \end{aligned}$$

$$\begin{aligned} T(n) \geqslant \\ \geqslant cn \log_{10} n + \Theta(n) \end{aligned}$$

Таким образом, $T(n) = \Theta(n \lg n)$

Т.е., несмотря на то, что разбиение в соотношении один к девяти выглядит довольно несбалансированным, в асимптоптическом пределе алгоритм ведет себя так же, как и при делении задачи на две равные части. Фактически, даже разбиение девяносто девять к одному даст время работы алгоритма $O(n \lg n)$. Причина в том, что любое разбиение, характеризующееся конечной *константной* пропорциональности, приводит к образованию рекурсивного дерева высотой $\Theta(\lg n)$ и временем работы каждого уровня $O(n)$. Время работы алгоритма будет составлять $O(n \lg n)$.

Предположим, что процедура поочередно генерирует "хорошие" и "плохие" разбиения.

В "хорошем случае": $L(n) = 2U(\frac{n}{2}) + \Theta(n)$

В "плохом случае": $U(n) = L(n-2) + \Theta(n)$

Получаем систему уравнений, в которой граничные случае с константными данными опущены.

$$\begin{aligned} L(n) = \\ = 2(L(\frac{n}{2} - 1) + \Theta(\frac{n}{2})) + \Theta(n) = \\ = 2L(\frac{n}{2} - 1) + \Theta(n) = \\ = \Theta(n \lg n) \text{ (по основному методу)} \end{aligned}$$

4 Рандомизированные алгоритмы

Предположим, компания хочет взять на работу нового программиста. Согласно договоренности, агентство должно присылать работодателю по одному кандидату в день. HR проводит с каждым собеседование, после чего принимает решение, брать его на работу или нет. За каждого направленного кандидата компания платит агентству некоторую сумму. Однако наем выбранного кандидата на работу фактически стоит дороже, поскольку при этом необходимо уволить предыдущего сотрудника, а также уплатить агентству более значительную сумму за выбранного кандидата.

Компания пытается сделать так, чтобы должность всегда занимал наиболее достойный из всех кандидатов. Как только квалификация очередного претендента окажется выше квалификации существующего, он будет уволен, а его место займет более подходящий кандидат. Работодатель готов понести все необходимые расходы, но хочет оценить, во что обойдется ему подбор нового сотрудника.

Процедура Hire описывает алгоритм найма в виде псевдокода:

```
HIRE(n)
1  Best ← 0
2  for i ← 1 to n
3        do Собеседование с кандидатом i
4           if кандидат i лучше Best
5              then Best ← i
6                   Нанимаем кандидата i
```

Пусть стоимость собеседования равна c_i, а стоимость найма – c_h. Пусть m – количество нанятых сотрудников. Тогда полная стоимость, затраченная при работе этого алгоритма, равна $O(nc_i + mc_h)$.

Очевидно, что в интересах агенства максимизировать m, а в интересах компании – минимизировать.

Одним из способов избежать наихудшего случая, когда $m = n$, является рандомизация – контроль за распределением входных данных.

5 Вероятностный анализ. Часть 1

При анализе рандомизированных алгоритмов будет использоваться индикаторная случайная величина. С помощью неё можно сделать удобный переход от вероятности к математическому ожиданию. Пусть дано пространство выборки S и событие A. Тогда индикаторная случайная величина $I\{A\}$, связанная с событием A, определяется так:

$$I\{A\} = \begin{cases} 1, \text{если событие } A \text{ произошло} \\ 0, \text{событие } A \text{ не произошло} \end{cases}$$

Пример использования: определим мат. ожидание того, что при подбрасывании монеты выпадет орел. Пространство событий имеет вид $S = \{H, T\}$, где вероятность выпадения $Pr\{H\} = Pr\{T\} = \frac{1}{2}$. Можно определить индикаторную величину X_H, связанную с событием H. Математическое ожидание того, что выпадет орел, равняется мат. ожиданию индикаторной величины X_H:

$$\begin{aligned} E[X_H] = E[I\{H\}] = \\ = 1 \cdot Pr\{H\} + 0 \cdot Pr\{T\} = \\ = 1 \cdot (1/2) + 0 \cdot (1/2) = \\ = 1/2. \end{aligned}$$

Полезная лемма: пусть событие A принадлежит пространству событий S, и пусть $X_A = I\{A\}$. Тогда мат. ожидание

$$E[X_A] = Pr\{A\} \tag{1}$$

Указание к доказательству: использовать определению мат. ожидания для случайной величины $E[X] = \sum_x xPr\{X = x\}$

6 Вероятностный анализ. Часть 2

Необходимо оценить мат. ожидание события, соответствующего найму нового программиста. Пусть X – случайная величина, значение которое равно количеству наймов нового сотрудника. По определению мат. ожидания:

$$E[X] = \sum_x xPr\{X = x\}$$

Используем индикаторную величину:

$$X_i = I\{i\text{-й кандидат нанят}\} = \begin{cases} 1, \text{если кандидат нанят} \\ 0, \text{если кандидат отклонен} \end{cases}$$

$$X = X_1 + X_2 + .. + X_n$$

Из леммы:

$$E[X_i] = Pr\{i\text{-й кандидат нанят}\}$$

Вероятность того, что претендент с номером i выше квалификации с номерами от 1 до $i-1$ равна $1/i$, т.о.

$$E[X_i] = 1/i$$

Тогда:

$$\begin{aligned} E[X] = E[\sum_i X_i] = \\ = \sum_i E[X_i] = \\ = \sum_i \frac{1}{i} = \text{ (гармоническое число)} \\ = \ln n + O(1) \end{aligned}$$

Таким образом, если проведено интервью со всеми n кандидатами, в среднем будет нанято $\ln n$ из них. Тогда полная стоимость найма будет составлять $O(c_h \ln n)$.

7 Рандомизированный Quicksort

Исправить ситуацию с наихудшим случаем можно двумя способами: случайно перемешать элементы массива или случайно выбирать опорный элемент. Тогда:

- время выполнения не зависит от порядка элементов
- не зависит от входного распределения
- наихудший случай зависит только от характеристик генератора случайных чисел

Для реализации достаточно модифицировать процедуру разбиения:

RANDOMIZED_PARTITION(A, p, r)
1 $i \leftarrow$ Random(p, r)
2 обменять $A[i] \iff A[j]$
3 **return** Partition(A, p, r)

8 Анализ рандомизированного Quicksort

Легко показать, что в наихудшем случае время работы Randomized_Quicksort равно $\Theta(n^2)$

$$T(n) = \max_{0 \leqslant q \leqslant n-1} (T(q) + T(n-q-1)) + \Theta(n)$$

Параметр q изменяется в интервале от 0 до $n-1$, поскольку на выходе Partition мы получаем две подзадачи суммарным размером $n-1$. Предположим, что $T(n)$ ограничено сверху cn^2:

$$\begin{aligned} T(n) &\leqslant \max_{0 \leqslant q \leqslant n-1} (cq^2 + c(n-q-1)^2) + \Theta(n) = \\ &= c \cdot \max_{0 \leqslant q \leqslant n-1} (q^2 + (n-1-q)^2) + \Theta(n) \leqslant \\ &\leqslant cn^2 - c(2n-1) + \Theta(n) \leqslant \\ &\leqslant cn^2 \end{aligned}$$

Т.е. $T(n) = O(n^2)$. Аналогично доказывается $T(n) = \Omega(n^2)$

Для среднего случая рандомизированной версии Quicksort, когда все элементы во входном массиве различны, время работы $T(n) = \Theta(n \lg n)$.

8.1 Схема доказательства

Для $k = 0, 1, \ldots, n-1$ индикаторные случайные переменные

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

равны 0 все кроме одной. Тогда мат. ожидание

$$E[X_k] = Pr\{X_k = 1\} = \frac{1}{n}, \text{ т.к. все разбиения равновероятны}$$

Время работы алгоритма

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

Используя индикаторную переменную, можно перейти от системы к сумме:

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

Мат. ожидание времени работы рандомизированного Quicksort будет равно:

$$E[T(n)] = \sum E[X_k] E[T(k) + T(n-k-1) + \Theta(n)] =$$

используя свойство линейности мат. ожидания и независимость переменных X_k:

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

можно показать, что два первых слогаемых равны:

$$= \frac{2}{n} \sum_{k=0}^{n-1} E[T(k)] + \frac{1}{n} \Theta(n^2) =$$

случаи $k = 0$ и $k = 1$ константны:

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

Используя метод подстановок, можно показать, что

$$E[T(n)] \leqslant an \lg n, \text{ для } a > 0$$

Уазание: использовать неравенство:

$$\sum_{k=2}^{n-1} k \lg k \leqslant \frac{1}{2} n^2 \lg n - \frac{1}{8} n^2$$