

# Разбор примерной контрольной работы по алгоритмам

Орлов Никита

28 февраля 2017 г.

## Задача 1

Приведите примеры функций  $f(n)$  и  $g(n)$ , таких что  $f(n) = \Theta(g(n))$  и одновременно  $2^{f(n)} = \bar{O}(2^{g(n)})$ , или докажите, что таких функций не существует.

### Решение

Допустим, что  $f(n) = \Theta(g(n))$ . Распишем  $\Theta$  по определению:

$$f(n) = \Theta(g(n)) \Leftrightarrow \exists c_1, c_2, n_0 > 0 \forall n \geq n_0 : 0 \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$$

Возведем 2 в степень всех частей неравенства, при этом неравенство останется верным, поскольку  $a < b \Rightarrow 2^a < 2^b$ :

$$2^0 \leq 2^{c_1 g(n)} \leq 2^{f(n)} \leq 2^{c_2 g(n)}$$

Получили, что  $2^{f(n)} = \Theta(2^{g(n)})$ , противоречие условию, а значит таких функций не существует.

## Задача 2

Решите рекуррентное соотношение в терминах  $\Theta$ :

$$T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{n}{4}\right) + 5n$$

### Решение

Оценим сумму сверху и снизу. Так как

$$2T\left(\frac{n}{3}\right) + 5n \leq T\left(\frac{n}{3}\right) + T\left(\frac{n}{4}\right) + 5n \leq 2T\left(\frac{n}{4}\right) + 5n$$

По основной теореме о рекуррентных соотношениях,

$$O(n) \leq T\left(\frac{n}{3}\right) + T\left(\frac{n}{4}\right) + 5n \leq O(n) \Rightarrow T(n) = \Theta(n)$$

## Задача 3

Медиана упорядоченного по возрастанию множества из  $k$  элементов – это  $\frac{k}{2}$ -ый по порядку элемент этого множества в предположении, что  $k$  – четно. Пусть  $X[1..n]$  и  $Y[1..n]$  – два отсортированных массива, каждый из которых содержит по  $n$  элементов. Опишите алгоритм, в котором поиск медианы всех  $2n$  элементов, содержащихся в массивах  $X$ ,  $Y$ , выполнялся бы за время  $O(\log n)$ .

### Решение

Будем поступать следующим образом. Возьмем медианы в этих массивах, пусть их индексы в массивах равны соответственно  $m_1$  и  $m_2$ . Если  $X[m_1] < Y[m_2]$ , то будем искать медиану в массивах  $X[: m_1]$  и  $Y[m_2 :]$ , если  $X[m_1] > Y[m_2]$ , то будем искать медиану в массивах  $X[m_1 :]$  и  $Y[: m_2]$  рекурсивно. Если же медианы равны, то выдаем одну из них. Этот алгоритм на каждом шаге уменьшает в два раза размер задачи, а значит сложность – сумма убывающей геометрической прогрессии – равна  $O(\log n)$

## Задача 4

Опишите, как реализовать структуру данных, поддерживающую два стека, используя один массив  $A[1..n]$  из  $n$  элементов. Опишите реализацию процедур *IsEmpty1()*, *Push1(v)*, *Pop1()* и аналогичные для второго стека. Переполнение должно происходить при попытке вставки нового элемента в один из стеков в ситуации, когда суммарное число элементов в обоих стеках равно  $n$ .

### Решение

Так как нам известен размер массива, будем заполнять один стек с начала массива, а другой с конца. Будем хранить индексы последних элементов каждого стека. Примерная реализация структуры:

---

**Алгоритм 1** Структура double-ended stack

---

**struct** double-ended stack:

array  $A[1..n]$

integer  $end1 = 1, end2 = n$

**function**  $Push1(v)$

**if**  $end1 + 1 == end2$  **then**

**return error**

$end1 := end1 + 1$

$A[end1] = v$

**function**  $Pop1()$

**if**  $end1 - 1 == -1$  **then**

**return error**

$res := A[end1]$

$end1 := end1 - 1$

**return**  $res$

**function**  $IsEmpty1()$

**return**  $end1 == 0$

**function**  $Push2(v)$

**if**  $end2 - 1 == end2$  **then**

**return error**

$end2 := end2 - 1$

$A[end2] = v$

**function**  $Pop2()$

**if**  $end2 + 1 == n + 1$  **then**

**return error**

$res := A[end2]$

$end2 := end2 + 1$

**return**  $res$

**function**  $IsEmpty2()$

**return**  $end2 == n + 1$

---

## Задача 5

Структура данных поддерживает операцию, такую что последовательность из  $n$  вызовов той операции занимает время  $\Theta(n \log n)$  в худшем случае. Какова амортизированная сложность этой операции? Какой может быть сложность этой операции в худшем случае? Приведите пример структуры данных и операции, для которых достигается этот худший случай?

## Решение

Получить амортизированную стоимость просто: поделим время последовательности вызовов на число вызовов. Получим:

$$\frac{n \log n}{n} = \log n$$

*TO BE CONTINUED. ETA: 22:00 28.02.2017*