

# Практика по алгоритмам

Всеволод Опарин, Алексей Давыдов

Осень, 2014

## 1 Практика 1. Асимптотика

### 1.1 Вспомогательные факты

1. Сумма арифметической прогрессии ( $a_{i+1} = a_i + d$ ):  $a_1 + a_2 + \dots + a_n = \frac{a_1 + a_n}{2} \cdot n$ .
2. Сумма геометрической прогрессии:  $1 + p + \dots + p^{n-1} = \frac{p^n - 1}{p - 1}$ ,  $\sum_{i=0}^{\infty} p^i = \frac{1}{1-p}$ . при  $0 < p < 1$ .
3. Гармонический ряд:  $\sum_{i=1}^n \frac{1}{i} = \log n + O(1)$ ,  $\sum_{i=1}^n \frac{1}{i^{1+\epsilon}} = O(1)$ .
4. Оценки на суммы:

(a) Мажорирование:

$$\sum_{i=0}^n i \leq \sum_{i=0}^n n = O(n^2).$$

(b) Разделение на суммы

$$\sum_{i=0}^n i \geq \sum_{i=\lfloor n/2 \rfloor}^n \lfloor n/2 \rfloor = \Omega(n^2).$$

(c) Интегрирование

$$\Omega(n^2) = \int_0^n x dx \leq \sum_{i=1}^n i \leq \int_1^{n+1} x dx = O(n^2).$$

### 1.2 Практика

Асимптотика:

1.  $n^k$  и  $c^n$  сравнить по  $O$ -нотациями.
2.  $p(n) = a_0 + a_1x + \dots + a_dx^d$ . Для каких  $k$  какие отношения между  $p(n)$  и  $n^k$ .
3.  $\max(f(n), g(n)) = \Theta(f(n) + g(n))$ ?
4.  $2^{n+1} = O(2^n)$ ?  $2^{2n} = O(2^n)$ ?
5.  $f(n) = O(f(n)^2)$ ?
6.  $f(n) + g(n) = \Theta(\min(f(n), g(n)))$ ?
7.  $f(n) = O(g(n)) \Rightarrow \log f(n) = O(\log g(n))$ ?
8.  $f(n) = O(g(n)) \Rightarrow 2^{f(n)} = O(2^{g(n)})$ ?

9.  $f(n) = f(n/2)$ ?
10.  $f(n) + o(f(n)) = \Theta(f(n))$ ?
11.  $\log n! = \Theta(n \cdot \log n)$ ?

Суммы:

1.  $\sum_{i=1}^n \frac{1}{i} = \Omega(\log n)$  через интегралы и разбиения на части.
2.  $\sum_{k=1}^n \frac{1}{k^2} = O(1)$ .
3.  $\sum_{k=0}^{\log n} \lceil n/2^k \rceil$ .
4.  $\sum_{k=1}^n \frac{1}{2k-1} = \ln(\sqrt{n}) + O(1)$ .
5.  $\sum_{k=0}^{\infty} (k-1)/2^k = ?$ .
6.  $\prod_{k=1}^n (2 \cdot 4^k)$ .
7.  $\prod_{k=2}^n (1 - 1/k^2)$ .

### 1.3 Домашнее задание

Дедлайн: 23.59 11.09.2014

Группа Давыдова решает все суммы в качестве домашнего задания.

1. Посчитать произведения

- (a)  $\prod_{k=1}^n (2 \cdot 4^k)$ .
- (b)  $\prod_{k=2}^n (1 - 1/k^2)$ .

2. Заполнить табличку.

$A$	$B$	$O$	$o$	$\Theta$	$\omega$	$\Omega$
$\lg^k n$	$n^\epsilon$					
$n^k$	$c^n$					
$\sqrt{n}$	$n^{\sin n}$					
$2^n$	$2^{n/2}$					
$n^{\lg m}$	$m^{\lg n}$					
$\lg(n!)$	$\lg(n^n)$					

3. Упорядочить функции в порядке возрастания.

$\lg(\lg^* n)$	$2^{\lg^* n}$	$(\sqrt{n})^{\lg n}$	$n^2$	$n!$	$(\lg n)!$
$(3/2)^n$	$n^3$	$\lg^2 n$	$\lg n!$	$2^{2^n}$	$n^{1/\lg n}$
$\ln \ln n$	$\lg^* n$	$n \cdot 2^n$	$n^{\lg \lg n}$	$\ln n$	$1$
$2^{\ln n}$	$(\lg n)^{\lg n}$	$e^n$	$4^{\lg n}$	$(n+1)!$	$\sqrt{\lg n}$
$\lg^* \lg n$	$2^{\sqrt{2 \lg n}}$	$n$	$2^n$	$n \lg n$	$2^{2^{n+1}}$

Примечание:  $\lg^*(n) = \begin{cases} 0 & \text{если } n \leq 1; \\ 1 + \lg^*(\lg n) & \text{иначе.} \end{cases}$

## 2 Практика 2. Линейные алгоритмы

### 2.1 Практика

1. Реализовать стек с операциями PUSH, POP, MAX при условии, что каждая операция работает за константное время.
2. Реализовать очередь с операциями PUSH, POP, MAX при условии, что каждая операция работает за константное время.
3. Дана последовательность  $a_1, a_2, \dots, a_n \in \mathbb{N}$  и  $S \in \mathbb{N}$ . Найти  $l, r$  ( $1 \leq l \leq r \leq n$ ) такие, что сумма  $\sum_{i=l}^r a_i = S$ . Задачу требуется решить за линейное от  $n$  время.

### 2.2 Домашнее задание

Дедлайн: 18 сентября, 23.59

1. Дана последовательность  $a_1, a_2, \dots, a_n \in \mathbb{Z}$ . Найти  $l, r$  ( $1 \leq l \leq r \leq n$ ) такие, что сумма  $\sum_{i=l}^r a_i$  была бы максимальной. Задачу требуется решить за линейное от  $n$  время.
2. Дана скобочная последовательность, составленная из скобок '(', ')', '[', ']', '{', '}'. Последовательность называется корректной, если каждой открывающей скобке соответствует закрывающая скобка того же типа, и соблюдается вложенность. Примеры,  $(\{\})$  и  $()()$  – корректные, а  $()$  и  $[(])$  – нет.

Предоставьте алгоритм, который проверяет корректность последовательности за линейное время.

3. Дан массив целых чисел  $a_i$ . Придумайте структуру данных, которая бы умела отвечать на запросы вида: "По данным  $l$  и  $r$  вернуть  $\sum_{i=l}^r a_i$ ." за  $O(1)$ .

Разрешается сделать предподсчет за  $O(n)$ . Значения в массиве не меняются.

4. Дано число, представленное  $n$  цифрами в десятичной записи без ведущих нулей. Из числа требуется вычеркнуть ровно  $k$  цифр так, чтобы результат был бы максимальным. Задачу требуется решить за линейное от  $n$  время.

**Внимание!**  $k$  подается на вход и может быть порядка  $n$ . Решение за  $O(kn)$  приравнивается к квадратичному от  $n$ .

5. (\*) Дана последовательность объектов  $a_1, a_2, \dots, a_n$ . Над объектами определена операция сравнения. Известно, что в последовательности есть элемент присутствующий не менее чем  $\lfloor \frac{n}{2} \rfloor + 1$ . Требуется найти элемент  $a$  за линейное время и константу дополнительной памяти при условии, что последовательность задана форвард-итератором.
6. (\*) Дана последовательность  $a_1, a_2, \dots, a_n \in \mathbb{N}$ . Найти  $l, r$  ( $1 \leq l \leq r \leq n$ ) такие, что значение  $(r - l + 1) \min_{i \in [l, r]} a_i$  было бы максимально. Задачу требуется решить за линейное от  $n$  время.

## 3 Практика 3. Разделяй-и-властвуй

### 3.1 Вспомогательные факты

1. Пусть  $a \geq 1$  и  $b > 1$  — константы,  $f(n)$  — функция,  $T(n)$  определено при неотрицательных  $n$  формулой

$$T(n) = aT(n/b) + f(n),$$

где под  $n/b$  понимается либо  $\lceil n/b \rceil$ , либо  $\lfloor n/b \rfloor$ . Тогда

- (a) если  $f(n) = O(n^{\log_b a - \epsilon})$  для некоторого  $\epsilon > 0$ , то  $T(n) = \Theta(n^{\log_b a})$ ;
  - (b) если  $f(n) = \Theta(n^{\log_b a})$ , то  $T(n) = \Theta(n^{\log_b a} \lg n)$ ;
  - (c) если  $f(n) = \Omega(n^{\log_b a + \epsilon})$  для некоторого  $\epsilon > 0$  и если  $af(n/b) \leq cf(n)$  для некоторой константы  $c < 1$  и достаточно больших  $n$ , то  $T(n) = \Theta(f(n))$ .
2. Комментарий к задачам про оценку асимптотики. Можно считать, что для некоторой константы  $n_0$  и всех  $n \leq n_0$  верно, что  $T(n) = 1$ .
  3. **Утверждение.** Пусть  $y(n)$  — монотонная неограниченная функция, и  $f(n) = O(g(n))$ . Тогда  $f(y(n)) = O(g(y(n)))$ .

**Доказательство.** Нам известно, что для некоторой константы  $c$  и  $n_0$ ,  $f(n) \leq c \cdot g(n)$  при  $n \geq n_0$ . Заметим, что, начиная с некоторой величины  $n'$ ,  $y(n) > n_0$ . Значит,  $f(y(n)) \leq c \cdot g(y(n))$ , начиная с  $n'$ . Значит,  $f(y(n)) = O(g(y(n)))$ .

4. Разбор задачи  $T(n) = 2 \cdot T(\sqrt{n}) + \log n$ .

Проведем замену переменных. Пусть  $y = \log n$ . Тогда  $T(2^y) = 2 \cdot T(2^{\frac{y}{2}}) + y$ . Пусть  $T'(y) = T(2^y)$ . Тогда, для  $T'$  справедлива следующая рекуррентная формула

$$T'(y) = 2 \cdot T'(y/2) + y.$$

По теореме о рекуррентных соотношениях получаем, что  $T(2^y) = T'(y) = O(y \cdot \log y)$ . Подставим  $y = \log n$ . По предыдущему пункту получим, что  $T(2^{\log n}) = T(n) = O(\log n \cdot \log \log n)$ .

### 3.2 Практика

1. Определить асимптотику.

- (a)  $T(n) = 2 \cdot T(\lfloor \frac{n}{2} \rfloor + 17) + n$ .

### 3.3 Домашнее задание

Дедлайн: 25 сентября, 23.59

1. Определить асимптотику.

- (a)  $T(n) = T(\lceil \frac{n}{2} \rceil) + 1$ .

- (b)  $T(n) = T(a) + T(n - a) + n$  для произвольной константы  $a$ .

- (c)  $T(n) = T(\alpha \cdot n) + T((1 - \alpha) \cdot n) + n$  для произвольной константы  $\alpha \in (0, 1)$ .

- (d)  $T(n) = 4 \cdot T(\lfloor \frac{n}{2} \rfloor) + n^k$  для  $k \in \{1, 2, 3\}$ .

2. Определить асимптотику  $T(n) = 2 \cdot T(\lfloor \log n \rfloor) + 2^{\log^* n}$ .

3. Есть  $k$  отсортированных массивов. В сумме массивы содержат  $n$  элементов. Слить массивы за  $O(n \log k)$ .

4. Назовем массив  $A[1..n]$  унимодальным, если он сначала возрастает, а потом убывает. Строго говоря, существует такое  $m \in [1, n]$ , что

- $A[i] < A[i + 1]$  для  $1 \leq i < m$ ;
- $A[i] > A[i + 1]$  для  $m \leq i < n$ .

Элемент с номером  $m$  назовем *пиком*.

(a) Постройте алгоритм для поиска пика за  $O(\log n)$ .

(b) Дан выпуклый многоугольник на плоскости из  $n$  вершин. Вершины заданы в порядке обхода по часовой стрелке. Никакие три подряд идущие вершины не лежат на одной прямой. Требуется найти минимальный прямоугольник со сторонами, параллельными осям координат, содержащий данный многоугольник (bounding box) за  $O(\log n)$ .

5. \* Дан массив из  $n$  различных элементов. Требуется найти число инверсий за  $O(n \log n)$  (число инверсий в массиве  $a$ , это число таких пар  $(i, j)$ , что  $i < j$ , но  $a[i] > a[j]$ ).

6. \* Структура данных файл последовательного доступа поддерживает следующие операции:

- *Read()*: чтение числа из файла на текущей позиции и перевод позиции вперед на 1 элемент.
- *Write(x)*: запись числа в файл в текущую позицию и перевод позиции вперед на 1 элемент.
- *Rewind()*: перевод позиции на начало файла.

Требуется отсортировать файл за  $O(n \log n)$  используя  $O(1)$  памяти и два дополнительных файла.

## 4 Практика 4. Сортировки

### 4.1 Практика

1. Даны два массива  $a$  и  $b$  длины  $n$ , сгенерировать все попарные суммы  $a_i + b_j$  в отсортированном порядке.
  - (a) За  $O(n^2 \log n)$ .
  - (b) За  $O(n^3)$  с использованием  $O(n)$  дополнительной памяти.
  - (c) За  $O(n^2 \log n)$  с использованием  $O(n)$  дополнительной памяти.
2. В свободное время Анка-пулеметчица любит сортировать патроны по серийным номерам. Вот и сейчас она только разложила патроны на столе в строго отсортированном порядке. Но тут Иван Васильевич распахнул дверь с такой силой, что все патроны на столе подпрыгнули и немного перемешались. Оставив ценные указания, Иван Васильевич отправился восвояси. Как оказалось, патроны перемешались не сильно. Каждый патрон отклонился от своей позиции не более чем на  $k$ . Всего патронов  $n$ . Помогите Анке отсортировать патроны.
  - (a) Отсортируйте патроны за  $O(nk)$ .
  - (b) Отсортируйте патроны за  $O(n + I)$ , где  $I$  — число инверсий.
  - (c) Докажите нижнюю оценку на время сортировки  $\Omega(n \log k)$ .
  - (d) Отсортируйте патроны за  $O(n \log k)$ .
3. Дано  $n$  точек на плоскости. Никакие три не лежат на одной прямой. Соединить точки
  - (a)  $(n - 1)$ -звенной ломаной без самопересечений (не замкнутой);
  - (b)  $n$ -звенной ломаной без самопересечений (замкнутой)за  $O(n \log n)$ .
4. Дан набор из  $n$  отрезков  $[a_i, b_i]$ . Числа  $a_i, b_i$  — вещественные.
  - (a) Найти такое вещественное  $x$ , что  $|\{i: x \in [a_i, b_i]\}|$  максимально.
  - (b) Длину объединения отрезков.
  - (c) Для каждого  $k$  определим множество точек  $S_k$ , покрытых ровно  $k$  отрезками. Множество  $S_k$  можно представить как набор непересекающихся отрезков. Найти суммы длин этих отрезков для каждого  $k$ .за  $O(n \log n)$
5. Даны два массива  $a$  и  $b$  одинаковой длины. Нужно найти такую перестановку  $p$ , что  $\sum_{i=1}^n a_{p_i} b_i \rightarrow \max$ . Решение обосновать.

## 4.2 Домашнее задание

Дедлайн: 2 октября, 23.59

1. Дано бинарное дерево:  $Tree ::= Node(Tree, Tree) | Empty$  (эта запись означает, что дерево — это либо вершина с парой потомков-деревьев, либо особое значение  $Empty$ ). Определим функцию  $\mathbf{rank}(x)$  следующим образом:

- $\mathbf{rank}(Empty) = 0$
- $\mathbf{rank}(Node(left, right)) = \min(\mathbf{rank}(left), \mathbf{rank}(right)) + 1$ .

Назовем бинарное дерево *скошенным влево (левацким)*, если для его вершин выполнено следующее свойство:

$$\forall_{x=Node(left, right)} \mathbf{rank}(left) \geq \mathbf{rank}(right).$$

*Скошенная влево (левацкая) куча* — это скошенное влево дерево, в вершинах которого хранятся данные, для которых выполнено свойство кучи.

- Докажите, что для любого скошенного влево дерева  $\mathbf{rank}(T) \leq \log |T|$  ( $|T|$  обозначает количество вершин в дереве  $T$ ).
  - Придумайте, как слить две скошенные влево кучи  $H_1$  и  $H_2$  за время  $O(\log |H_1| + \log |H_2|)$ .
  - Придумайте, как используя операцию слияния, построенную на предыдущем шаге, реализовать операции:
    - $Insert(x)$  — добавление элемента  $x$  в кучу,
    - $Pull()$  — удаление минимального элемента из кучи.
2. Дано  $2n - 1$  коробок с черными и белыми шарами. В  $i$ -ой коробке находится  $w_i$  белых и  $b_i$  черных шаров. Всего в коробках находится  $W$  белых и  $B$  черных шаров. Требуется выбрать  $n$  коробок таким образом, чтобы суммарное число белых шаров в них было не менее  $\frac{W}{2}$ , а черных не менее  $\frac{B}{2}$ . Решить за  $O(n \log n)$ .
  3. На прямой расположено  $n$  точек  $p_1, p_2, \dots, p_n$  в порядке возрастания. Каждая точка имеет вес  $w_i \geq 0$ . Требуется найти такую точку  $q$ , что  $\sum_i w_i \cdot |p_i - q|$  имела бы минимальное значение. Время работы:  $O(n)$ .

**Примечание.** Точки на вход подаются в порядке возрастания.

4. (\*) Дан массив из  $n + 1$  целого числа от 1 до  $n$ . Массив доступен только на чтение, есть  $O(1)$  дополнительной памяти. Найти за  $O(n)$  любое число, которое встречается хотя бы два раза.

## 5 Практика 5. Быстрая сортировка

### 5.1 Практика

1. Робот Иван Семеныч пробует пирожки. Содержимое пирожков делится на три типа. Всего пирожков  $n$ . Каждый пирожок можно попробовать не более одного раза. Пирожки можно менять местами. Память у робота маленькая,  $O(\log n)$ .

Помогите Ивану Семенычу отсортировать пирожки по типу: сначала первый, потом второй, потом третий. Сортировка должна работать за линейное время.

2. Разработайте алгоритм, который позволит находить  $k$ -порядковую статистику через медиану.
3. Рассмотрим следующий алгоритм быстрой сортировки массива  $A[1..n]$ .

- Взять элемент с номером  $\text{pivot} = \lceil \frac{n}{2} \rceil$ .
- Перенести в отдельный массив все элементы меньше  $A[\text{pivot}]$ , затем равные, затем большие, сохраняя порядок.
- Запустить алгоритм рекурсивно на элементах меньших  $A[\text{pivot}]$ , затем на больших.

Будем оценивать сложность алгоритма как суммарную длину всех массивов, на которых будет рекурсивно запущен алгоритм.

Постройте алгоритм, который получает на вход  $n$  и строит наиболее сложный пример для сортировки за линейное от  $n$  время.

4. (\*) Куча хранится в массиве длины  $n$ . Родитель  $p$  хранит детей в ячейках  $2 \cdot p + 1$  и  $2 \cdot p + 2$ . Алгоритм приступает к сортировке. Сортировка устроена следующим образом.

- Поменять первый и последний элемент кучи местами.
- Уменьшить размер кучи на единицу.
- Запустить `heapifyDown` на первом элементе.

`heapifyDown` меняет родителя с наибольшим ребенком (при условии, что ребенок больше родителя) и запускается рекурсивно. Сложность кучи определим через число вызовов функции `heapifyDown`. Требуется придумать алгоритм, который по  $n$  строит самую сложную кучу. Ответ должен быть в виде перестановки из  $n$  элементов. Решить за  $O(n \log n)$ .

5. \* Придумайте структуру данных, которая поддерживает следующие операции (в оценках времени работы  $n$  — текущее количество элементов) :

- *Insert*( $x$ ) — добавление элемента  $x$  за  $O(\log n)$ ,
- *Pull*() — удаление минимального элемента за  $O(\log n)$ ,
- *Copy*() — копирование структуры за  $O(1)$  (после копирования с каждой из копий можно независимо проделывать любую из данных трех операций).

Подсказка: за основу можно взять левацкую кучу.