Практика по алгоритмам

Александр Мишунин, Алексей Давыдов*

Осень, 2015

*Составители сборника не являются авторами самих задач. Авторы не указаны в учебных целях.

1 Практика 1. Ассимптотика

1.1 Практика

Все функции: $\mathbb{N} \to \mathbb{N}$

- $f = O(g) \Leftrightarrow \exists_{N, C>0} \forall_{n \geq N} f(n) \leq C \cdot g(n)$
- $f(n) = \Omega(g(n)) \Leftrightarrow g(n) = O(f(n))$
- $f(n) = \Theta(g(n)) \Leftrightarrow f(n) = O(g(n)) \wedge f(n) = \Omega(g(n))$
- $f(n) = o(g(n)) \Leftrightarrow \forall_{C>0} \exists_N \forall_{n \geq N} f(n) < C \cdot g(n)$
- $f(n) = \omega(g(n)) \Leftrightarrow g(n) = o(f(n))$

1. (a) $\max(f(n), g(n)) = \Theta(f(n) + g(n))$?
   (b) $\min(f(n), g(n)) = \Theta(f(n) + g(n))$?
   (c) $f(n) = O(f(n)^2)$?
   (d) $f(n) = \Omega(f(n/2))$?
   (e) $f(n) = O(g(n)) \Rightarrow \log f(n) = O(\log g(n))$?
   (f) $f(n) = O(g(n)) \Rightarrow 2^{f(n)} = O(2^{g(n)})$?
   (g) $f(n) = o(g(n)) \Rightarrow \log f(n) = o(\log g(n))$?
   (h) $f(n) = o(g(n)) \Rightarrow 2^{f(n)} = o(2^{g(n)})$?
   (i) $\sum_{k=1}^{n} \frac{1}{k} = \Omega(\log n)$?
   (j) $\sum_{k=0}^{\log n} \lceil n/2^k \rceil = O(n^2)$?
   (k) $\prod_{k=1}^{n} (2 \cdot 4^k) = \Theta(1)$?

2. Во всех пунктах нужно ответить на вопрос “за сколько работает”:

   (a)

   ```
   for (a = 1; a < n; a++)
       for (b = 0; b < n; b += 1)
           ;
   ```

   (b)

   ```
   for (a = 1; a < n; a++)
       for (b = 0; b < n; b += a)
           ;
   ```

   (c) Найти такие a, b, c: $abc = n$, $a + b + c = \min$. Решение:

   ```
   for (a = 1; a <= n; ++a)
       for (b = 1; a * b <= n; ++b)
           c = N / a / b, ... ;
   ```

   (d) Еще одно решение задачи (c):

   ```
   for (a = 1; a * a * a <= n; ++a)
       for (b = 1; b * b <= n; ++b)
           c = N / a / b, ... ;
   ```

(e) И еще одно решение задачи (c):

- `for (a = 1; a * a * a <= n; ++a)`
- `    for (b = a; a * b * b <= n; ++b)`
- `        c = N / a / b, ... ;`

(f) Дополнительный вопрос: что делает этот код?

- `a = 1, b = n;`
- `while (a < b) {`
- `    while (x[a] < M && a <= b) a++;`
- `    while (x[b] > M && a <= b) b--;`
- `    if (a <= b) swap(x[a++], x[b--]);`
- `}`

(g) Дополнительный вопрос: а если бы вместо 2 было бы 1?

- `while (a >= 2)`
- `    a = sqrt(a);`

(h) Решето Эратосфена (пользуемся, что: $p_n \approx n \ln n$)

- `for (p = 2; p < n; p++)`
- `    if (min_divisor[p] == 0) // is prime`
- `        for (x = p + p; x < n; x += p)`
- `            if (min_divisor[x] == 0)`
- `                min_divisor[x] = p;`

1.2 Домашнее задание

1. (a) Если в определении O опустить условие про N (т.е. оставить просто $\forall n$), будет ли полученное определение эквивалентно исходному?

   (b) Тот же вопрос про o.

2. Во всех пунктах нужно ответить на вопрос “за сколько работает”:

   (a)

   ```
   for (i = 0; i < n; i++)
       if (used[i] == 0)
           for (j = i; used[j] == 0; j = (j*17+2) % n)
               used[j] = 1;
   ```

   (b) Школьная арифметика (длины чисел до n, система счисления десятичная):

      i. Сложение в столбик,

      ii. Умножение в столбик,

      iii. Деление в столбик.

   (c)

   ```
   int n = (int) s.length();
   vector<int> pi (n);
   for (int i=1; i<n; ++i) {
       int j = pi[i-1];
       while (j > 0 && s[i] != s[j])
           j = pi[j-1];
       if (s[i] == s[j]) ++j;
       pi[i] = j;
   }
   ```

3. Заполнить табличку.

A	B	O	o	Θ	ω	Ω
$\lg^k n$	n^ϵ					
n^k	c^n					
$\sqrt{n}$	$n^{\sin n}$					
2^n	$2^{n/2}$					
$n^{\lg m}$	$m^{\lg n}$					
$\lg(n!)$	$\lg(n^n)$					

4. (*) Вот обычный (но медленный) алгоритм Евклида:

   - `int gcd (int x, int y) {`
   - `    if (x < y)`
   - `        return gcd(x, y - x);`
   - `    if (x > y)`
   - `        return gcd(x - y, y);`
   - `    if (x == y)`
   - `        return (x + y) / 2;`
   - `}`

   Расширим его так:

   - `pair<int,int> gcd+ (int x, int y, int u, int v) {`
   - `    if (x < y)`
   - `        return gcd(x, y - x, u, u + v);`
   - `    if (x > y)`
   - `        return gcd(x - y, y, u + v, v);`
   - `    if (x == y)`
   - `        return pair<int,int> ((x + y) / 2, (u + v) / 2);`
   - `}`

   Что за пара будет в ответе, при запуске gcd$+(x, y, x, y)$? (Ответ нужно обосновать)

5. (*) Порой нам нужно найти не только d = НОД(x,y), но и такую пару целых чисел a и b, что $ax + by = d$. Придумайте, как изменить алгоритм Евклида, чтобы он находил такую пару.

6. Докажите, что следующий алгоритм находит gcd двух чисел:

   - `int gcd(int x, int y) {`
   - `    if (x == y)`
   - `        return (x + y) / 2;`
   - `    if (x % 2 == 0 && y % 2 == 0)`
   - `        return 2 * gcd(x / 2, y / 2);`
   - `    if (x % 2 == 0 && y % 2 != 0)`
   - `        return gcd(x / 2, y);`
   - `    if (x % 2 != 0 && y % 2 == 0)`
   - `        return gcd(x, y / 2);`
   - `    if (x > y)`
   - `        return gcd (x - y, y);`
   - `    if (x < y)`
   - `        return gcd (x, y - x);`
   - `}`

   Оцените его асимптотику.

2 Практика 2. Жадность. Линейные алгоритмы

1. 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.

2. Реализовать стек с операциями `PUSH`, `POP`, `MAX` при условии, что каждая операция работает за константное время.

3. Реализовать очередь с помощью двух стеков.

4. Реализовать очередь с операциями `PUSH`, `POP`, `MAX` при условии, что каждая операция работает за константное время.

5. Дано число, представленное n цифрами в десятичной записи без ведущих нулей. Из числа требуется вычеркнуть ровно k цифр так, чтобы результат был 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.

6. Преподаватели сделали n заявок на занятие. Каждое занятие начинается в момент b_i и кончается в момент e_i (занимает интервал $[b_i, e_i)$). Два занятия в одной аудитории быть не могут. Распределите заявки по аудиториям так, чтобы общее число аудиторий было минимально. Решить за $O(n \log n)$.

7. Придумайте алгоритм, который по заданному дереву T на n вершинах строит совершенное паросочетание, или говорит что такого нет. Время $O(n)$.

2.1 Домашнее задание

1. (группа Давыдова) Дано число, представленное n цифрами в десятичной записи без ведущих нулей. Из числа требуется вычеркнуть ровно k цифр так, чтобы результат был 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.

2. Придумайте алгоритм, который по заданному дереву T на n вершинах строит совершенное паросочетание (такое подмножество ребер, что каждая вершина дерева инцидентна ровно одному из них), или говорит, что такого нет. Время $O(n)$.

3. (группа Мишунина) Дан набор кубиков, на каждой грани каждого из них написано целое число. Петя и Вася играют в игру: каждый выбирает себе по кубику, затем каждый бросает свой кубик. Выигрывает тот, у кого выпало число больше (выпадение любой грани у кубика равновероятно). Петя выбирает кубик первым, и ему нужен алгоритм выбора лучшего кубика (лучший кубик — тот, у которого вероятность выиграть не меньше вероятности проиграть против любого другого кубика). Кубики заданы в виде массива шестерок чисел. Напишите псевдокод, возвращающий номер лучшего кубика (любого, если таких несколько) или -1, если такого кубика не существует (никто не обещает, что набор не пустой...).

4. (группа Давыдова) 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 $O(n)$. (Это означает, что нам дают массив, затем разрешают $O(n)$ времени считать какую-нибудь вспомогательную информацию и только после этого начинают задавать вопросы).

5. Дана последовательность $a_1, a_2, \cdots, a_n \in \mathbb{N}$ и $S \in \mathbb{N}$. Найти l, r $(1 \le l \le r \le n)$ такие, что сумма $\sum_{i=l}^{r} a_i = S$. Задачу требуется решить за линейное от n время.

6. Дана последовательность $a_1, a_2, \cdots, a_n \in \mathbb{Z}$. Найти l, r $(1 \le l \le r \le n)$ такие, что сумма $\sum_{i=l}^{r} a_i$ была бы максимальной. Задачу требуется решить за линейное от n время.

3 Практика 3. Сортировки и поиск

3.1 Практика

1. Дана монотонная функция $[1 \dots n] \to \{0, 1\}$. Напишите псевдокод, находящий последний 0 и первую 1 за $\mathcal{O}(\log n)$ вызовов функции.

2. Сделать предподсчет за $\mathcal{O}(n \log n)$, чтобы за $\mathcal{O}(\log n)$ отвечать на запрос "сколько раз число x встречается на отрезке $[l..r]$"?

3. Робот Иван Семеныч пробует пирожки. Содержимое пирожков делится на три типа. Всего пирожков n. Каждый пирожок можно попробовать не более одного раза. Любые два пирожка можно поменять местами. Память у робота маленькая, $\mathcal{O}(\log n)$ бит. Помогите Ивану Семенычу отсортировать пирожки по типу: сначала первый, потом второй, потом третий. Сортировка должна работать за линейное время.

4. Придумайте детерминированную структуру данных на основе бинарной кучи, которая умеет делать `Insert(x)`, `DeleteMedian()`, все операции за $\mathcal{O}(\log n)$.

5. Модифицируйте операцию `SiftUp` для бинарной кучи так, чтобы она по-прежнему работала за $\mathcal{O}(\log n)$, но при этом делала лишь $\mathcal{O}(\log \log n)$ сравнений.

6. Даны два сортированных массива длины n. Без дополнительного предподсчета найти k-ю порядковую статистику в объединении массивов.

   (a) За $\mathcal{O}(\log^2 n)$.

   (b) За $\mathcal{O}(\log n)$.

7. Дана обычная бинарная куча (с минимумом в голове), за сколько можно узнать k-й минимум?

   (a) $\mathcal{O}(k \log n)$

   (b) $\mathcal{O}(k^2)$

   (c) $\mathcal{O}(k \log k)$

8. Дан массив из n чисел и m чисел $p_1, p_2, \dots p_m$, нужно за $\mathcal{O}(n \log m + m)$ для каждого i найти p_i-ую порядковую статистику.

3.2 Домашнее задание

1. 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 $\mathcal{O}(nk)$.

   (b) Отсортируйте патроны за $\mathcal{O}(n + I)$, где I — число инверсий.

   (c) Докажите нижнюю оценку на время сортировки $\Omega(n \log k)$.

   (d) Отсортируйте патроны за $\mathcal{O}(n \log k)$.

2. Дан массив из n чисел и m чисел $p_1, p_2, \dots p_m$, нужно за $\mathcal{O}(n \log m + m)$ для каждого i найти p_i-ую порядковую статистику.

3. Даны два массива a и b одинаковой длины. Нужно найти такую перестановку p, что $\sum_{i=1}^{n} a_{p_i} b_i \to \max$.

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 `SiftDown` на первом элементе.

   `SiftDown` меняет родителя с наибольшим ребенком (при условии, что ребенок больше родителя) и запускается рекурсивно. Требуется придумать алгоритм, который по n выдвет перестановку чисел от 1 до n, которая является корректной кучей и приводит к максимальному количеству вызовов `SiftDown` при сортировке. Время работы — $\mathcal{O}(n \log n)$.

5. Найти количество AVL деревьев высоты h из n вершин по простому модулю p за $\mathcal{O}(hn^{\log_2 3})$ (AVL дерево — корневое двоичное дерево, в котором у каждой вершины высоты двух дочерних поддеревьев отличаются не более, чем на 1).

4 Практика 4. Структуры данных

4.1 Домашнее задание

1. 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 $\mathcal{O}(\log n)$.

   (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, параллелльными осям координат, содержащий данный многоугольник (bounding box) за $\mathcal{O}(\log n)$.

   (c) Придумайте, как проверить, лежит ли точка в многоугольнике за $\mathcal{O}(\log n)$.

2. Рассмотрим структуру данных счетчик. Она поддерживает следующие операции:

   - Увеличение счетчика на 1.
   - Уменьшение счетчика на 1.
   - Сравнение счетчика с 0.

   Пусть счетчик реализован как одно число в двоичной системе счисления. Докажите, что амортизационное время работы всех операций $\mathcal{O}(1)$.

3. Рассмотрим бинарную скошенную систему исчисления. На каждой позиции в скошенной записи числа может стоять цифра 0, 1 или 2. Число $a_k a_{k-1} \cdots a_2 a_1$ в скошенной системе переводится в десятичную по формуле $\sum_{i=1}^{k} a_i \cdot (2^i - 1)$.

   В скошенной системе счисления есть два ограничения: цифра 2 может встречаться в записи не более одного раза; все цифры следующих меньших разрядов равны нулю. Пример первых чисел: 1, 2, 10, 11, 12, 20, 100, 101 ...

   (a) Докажите, что каждое неотрицательное целое число имеет ровно одну возможную запись в скошенной системе счисления.

   (b) Придумайте, как увеличить число в скошенной системе на единицу за $\mathcal{O}(1)$.

4. * Рассмотрим структуру данных “скошенный список”. Для того, чтобы получить скошенный список длины n, сперва запишем число n в скошенной системе счисления. Далее для каждого i смотрим в соответствующую позицию скошенной записи n и, если соответствующее число не ноль, рисуем столько полных двоичных деревьев высоты i. Пример скошенных списков длины 1, 2, 3, 4, 5 - см. картинку.

Рис. 1: Лист [a] (число: 1)

Рис. 2: Лист [a b] (число: 2)

Рис. 3: Лист [a b c] (число: 10)

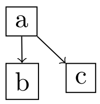


Рис. 4: Лист [a b c d] (число: 11)

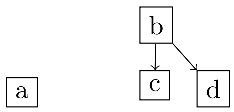
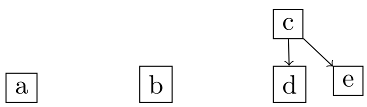


Рис. 5: Лист [a b c d e] (число: 12)



Придумайте, как реализовать следующие операции со списком длины n:

(a) Добавление элемента в начало списка за $\mathcal{O}(1)$.

(b) Доступ к i-му элементу за $\mathcal{O}(\log n)$.

(c) Возврат на произвольное предыдущее состояние за $\mathcal{O}(\log n)$.

(d) Получить список из k последних элементов данного списка за $\mathcal{O}(\log n)$.

5. (группа Мишунина) 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 $\mathbf{rank(}x\mathbf{)}$ следующим образом:

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

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 $|T| \geq 2^{\mathbf{rank}(T)-1}$ ($|T|$ 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.

6. Придумайте, как реализовать структуру данных, поддерживающую следующие операции:

- Добавление символа в конец.

- Откат назад на произвольный момент.
- Получение i-го символа.

Сложность каждой из операций не должна превышать $O(\log n)$.