Практика по алгоритмам, магистратура ВШЭ СПб

Осень 2021

Содержание

1 Асимптотика и линейные алгоритмы **2**
1.1 Практика . . . 2
1.2 Домашнее задание . . . 4

2 Новая русская музыкальная школа **5**
2.1 Практика . . . 5
2.2 Домашнее задание . . . 6

3 Шустрая сортировка и порядковые статистики **8**
3.1 Практика . . . 8
3.2 Домашнее задание . . . 10

4 Сортировки и жадности **11**
4.1 Практика . . . 11
4.2 Домашнее задание . . . 12

5 Демоническое программирование **13**
5.1 Практика . . . 13
5.2 Домашнее задание . . . 14

6 Динамика-2 и Хаффман **15**
6.1 Практика . . . 15

7 Поиск в глубину **16**
7.1 Практика . . . 16
7.2 Домашнее задание . . . 17

8 Кратчайшие пути **18**
8.1 Практика . . . 18
8.2 Домашнее задание . . . 20

9 Минимальные остовные деревья **21**
9.1 Практика . . . 21
9.2 Домашнее задание . . . 23

10 Поиск подстроки в строке **24**
10.1 Практика . . . 24
10.2 Домашнее задание . . . 25

11 Хеширование и бор **26**
11.1 Практика . . . 26
11.2 Домашнее задание . . . 27

12 Ахо — Корасик и теория чисел **29**
12.1 Практика 29
12.2 Домашнее задание 30

13 Высшая арифметика **31**
13.1 Практика 31
13.2 Домашнее задание 33

14 Оценки для хешей, BST и AVL **34**
14.1 Практика 34
14.2 Домашнее задание 36

1 Асимптотика и линейные алгоритмы

1.1 Практика

Напомним определения:

- $f(n) \in \mathcal{O}(g(n)) \equiv \exists N, C > 0 : \forall n \geq N : f(n) \leq C \cdot g(n)$
- $f(n) \in \Omega(g(n)) \equiv \exists N, C > 0 : \forall n \geq N : C \cdot g(n) \leq f(n)$
- $f(n) \in \Theta(g(n)) \equiv \exists N, C_1 > 0, C_2 > 0 : \forall n \geq N : C_1 \cdot g(n) \leq f(n) \leq C_2 \cdot g(n)$
- $f(n) \in o(g(n)) \equiv \forall C > 0 : \exists N : \forall n \geq N : f(n) < C \cdot g(n)$
- $f(n) \in \omega(g(n)) \equiv \forall C > 0 : \exists N : \forall n \geq N : C \cdot g(n) < f(n)$

Все функции здесь $\mathbb{N} \to \mathbb{N}$ или $\mathbb{N} \to \mathbb{R}_{>0}$ (далее будет ясно из контекста, какой класс функций используется). В дальнейшем, когда речь идет о принадлежности функций вышеопределенным множествам, мы будем использовать знак “$=$” вместо “$\in$”, т.к. в литературе обычно используются именно такие обозначения.

Асимптотики

1. Докажите, что:
   (a) $f(n) = \Omega(g(n)) \Leftrightarrow g(n) = \mathcal{O}(f(n))$
   (b) $f(n) = \omega(g(n)) \Leftrightarrow g(n) = o(f(n))$
   (c) $f(n) = \Theta(g(n)) \Leftrightarrow f(n) = \mathcal{O}(g(n)) \wedge f(n) = \Omega(g(n))$
2. **Контекст имеет значение**
   Правда ли, что $f(n) = \mathcal{O}(f(n)^2)$?
3. **Классы**
   Определим отношение “$\sim$”. Будем говорить, что $f \sim g$, если $f = \Theta(g)$. Покажите, что $\sim$ — отношение эквивалентности, т.е. оно
   - Рефлексивное: $\forall f : f \sim f$,
   - Симметричное: $\forall f, g : f \sim g \Leftrightarrow g \sim f$,
   - Транзитивное: $\forall f, g, h : (f \sim g) \wedge (g \sim h) \Rightarrow f \sim h$.
4. **Порядки**
   Определим отношение “$\preceq$”. Будем говорить, что $f \preceq g$, если $f = \mathcal{O}(g)$.
   Определим отношение $f \preceq g \equiv f = \mathcal{O}(g)$.
   (a) Докажите, что $\preceq$ — отношение предпорядка (рефлексивное и транзитивное)
   (b) Докажите, что $\preceq$ — не отношение частичного порядка, так как не удовлетворяет антисимметричности
   (c) Докажите, что $\preceq$ — отношение частичного порядка на классах эквивалентности по $\sim$?
5. Считайте, что функции здесь $\mathbb{N} \to \mathbb{N}$ и $\forall n : f(n) > 1 \wedge g(n) > 1$.
   (a) $f(n) = \Omega(f(n/2))$?
   (b) $f(n) = \mathcal{O}(g(n)) \Rightarrow \log f(n) = \mathcal{O}(\log g(n))$?
   (c) $f(n) = \mathcal{O}(g(n)) \Rightarrow 2^{f(n)} = \mathcal{O}(2^{g(n)})$?
   (d) $f(n) = o(g(n)) \Rightarrow \log f(n) = o(\log g(n))$?
   (e) $f(n) = o(g(n)) \Rightarrow 2^{f(n)} = o(2^{g(n)})$?
   (f) $\sum_{k=1}^{n} \frac{1}{k} = \Omega(\log n)$?

6. Определить асимптотику (считайте, что при $x \leq 100$ будет выполняться $T(x) = 100$).
   (a) $T(x) = T(a) + T(x - a) + n$ для натурального числа a.
   (b) $T(x) = T(\frac{x}{2}) + 1$.
   (c) $T(x) = 2 \cdot T(\sqrt{x}) + \log x$

Линейные алгоритмы

7. Вектор (расширяющийся массив) кроме операций `get(i)` и `set(i, x)` как у обычного массива умеет выполнять `push_back(x)` (`append(x)`) и `pop_back()` (`pop()`).

   Часто вектор реализуют следующим образом: храним обычный массив с некоторым запасом по памяти. Когда память заканчивается (из-за операций `push_back`), выделяем новый отрезок памяти большего размера, копируем туда текущий массив, старую память удаляем.

   Сравним две схемы перевыделения памяти:
   (a) При заполнении отрезка памяти размера n выделяем новый отрезок размером $n + c$ для какой-то константы c.
   (b) При заполнении отрезка памяти размера n выделяем новый отрезок размером cn для какой-то константы c.

   Оцените среднее (т. н. *амортизированное*) время работы операций для каждой из схем. На что влияет выбор константы c?

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

   Разрешается сделать предподсчёт за $\mathcal{O}(n)$. Значения в массиве не меняются.

9. Дана последовательность $a_1, a_2, \cdots, a_n \in \mathbb{N}$ 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.

10. 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.

    Придумайте алгоритм, который проверяет корректность последовательности за линейное время.

11. Пусть элементы здесь линейно упорядочены и мы умеем сравнивать их за $\mathcal{O}(1)$.
    (a) Придумайте стек, в котором можно узнавать минимум за $\mathcal{O}(1)$. Все остальные операции стека также должны работать за $\mathcal{O}(1)$.
    (b) Придумайте очередь, в которой можно узнавать минимум за $\mathcal{O}(1)$. Все остальные операции очереди должны работать за амортизированное $\mathcal{O}(1)$.
    (c) Придумайте вариант очереди с минимумом на основе пары из обычной очереди и дека потенциальных минимумов.

12. Дана последовательность $a_1, a_2, \cdots, a_n \in \mathbb{N}$. Для каждого a_i найти самый правый из элементов, которые левее и не больше его. Задачу требуется решить за линейное от n время.

13. Дана последовательность $a_1, a_2, \cdots, a_n \in \mathbb{N}$. Найти l, r $(1 \leq l \leq r \leq n)$ такие, что
    (a) значение $(r - l + 1) \min_{i \in [l,r]} a_i$ было бы максимально.
    (b) значение $\left(\sum_{i \in [l,r]} a_i\right) \min_{i \in [l,r]} a_i$ было бы максимально.

    Задачу требуется решить за линейное от n время.

14. Вам дан массив натуральных чисел и число k. Требуется найти подотрезок массива такой, что НОК чисел на нем равен k или заявить, что такого нет. Время работы: $\mathcal{O}(nT_{LCM}(k))$, где $T_{LCM}(k)$ — время подсчета НОК для чисел размера k.

1.2 Домашнее задание

В начале условия задачи могут быть указаны числа в квадратных скобках. Это номера групп, в которых эту задачу надо решать. Например, `[1, 3]` означает, что эту задачу нужно сдавать в первой и третьей группах.
Номера групп есть в распределении по группам.

1. `[1, 2, 3]` Дайте ответ для двух случаев $\mathbb{N} \to \mathbb{N}$ и $\mathbb{N} \to \mathbb{R}_{>0}$:
   (a) Если в определении $\mathcal{O}$ опустить условие про N (т.е. оставить просто $\forall n$), будет ли полученное определение эквивалентно исходному?
   (b) Тот же вопрос про o.
2. Считайте, что функции здесь $\mathbb{N} \to \mathbb{N}$ и $\forall n : f(n) > 1 \land g(n) > 1$.
   (a) `[2]` $f(n) = \mathcal{O}(g(n)) \Rightarrow \log f(n) = \mathcal{O}(\log g(n))$?
   (b) `[2]` $f(n) = \mathcal{O}(g(n)) \Rightarrow 2^{f(n)} = \mathcal{O}(2^{g(n)})$?
   (c) `[2, 3]` $f(n) = o(g(n)) \Rightarrow \log f(n) = o(\log g(n))$?
   (d) `[2, 3]` $f(n) = o(g(n)) \Rightarrow 2^{f(n)} = o(2^{g(n)})$?
   (e) `[2, 3]` $\sum_{k=1}^{n} \frac{1}{k} = \Omega(\log n)$?
3. `[1, 2, 3]` Заполните табличку и поясните (особенно строчки 4 и 7):

A	B	$\mathcal{O}$	o	Θ	ω	Ω
n	n^2	$+$	$+$	$-$	$-$	$-$
$\log^k n$	n^ϵ					
n^k	c^n					
$\sqrt{n}$	$n^{\sin n}$					
2^n	$2^{n/2}$					
$n^{\log_2 m}$	$m^{\log_2 n}$					
$\log(n!)$	$\log(n^n)$					

   Здесь все буквы, кроме n, — положительные константы.
4. `[3]` 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.
   Можно воспользоваться следующим планом решения:
   - Для каждого i найдите максимальное такое r_i, что $\sum_{j=i}^{r_i} a_j \le S$ за суммарное время $\mathcal{O}(n^2)$.
   - Найдите за $\mathcal{O}(n)$ ответ задачи, если известны $r_1, \ldots, r_n$.
   - Докажите, что $r_i \le r_{i+1}$.
   - Пользуясь предыдущим пунктом найдите все r_i за $\mathcal{O}(n)$.
5. `[2]` Дана последовательность $a_1, a_2, \cdots, a_n \in \mathbb{N}$.
   (a) Найти l, r $(1 \le l \le r \le n)$ такие, что значение $(r - l + 1) \min_{i \in [l,r]} a_i$ было бы максимально.
   (b) Найти l, r $(1 \le l \le r \le n)$ такие, что значение $\left(\sum_{i \in [l,r]} a_i\right) \min_{i \in [l,r]} a_i$ было бы максимально.

   Задачи требуется решить за линейное от n время.
6. `[1, 2, 3]` Вам дан массив из n элементов и список из m запросов $add(x, l, r)$: прибавить x к каждому элементу на отрезке $[l, r]$. За $\mathcal{O}(n+m)$ выведите массив, получающийся из исходного после выполнения заданных запросов.
7. `[1, 2, 3]` Определить асимптотику через Θ для $T(n) = 2 \cdot T(\lfloor \log n \rfloor) + 2^{\log^* n}$, где $\log^* n$ — итерированный логарифм.
   Итерированный логарифм определяется так: $\log^*(n) = \begin{cases} 0 & \text{если } n \le 1; \\ 1 + \log^*(\log n) & \text{иначе.} \end{cases}$

2 Новая русская музыкальная школа

2.1 Практика

Двоичная куча:

```
def siftUp(cur): # Просеивание вверх
  while cur > 1 and H[cur] < H[cur // 2]:
    swap(H[cur], H[cur // 2])
    cur //= 2

def siftDown(cur): # Просеивание вниз
  while cur * 2 <= sz:
    mv = cur * 2
    if cur * 2 + 1 <= sz and H[cur * 2 + 1] < H[mv]:
      mv = cur * 2 + 1
    if H[cur] <= H[mv]:
      break
    swap(H[cur], H[mv])
    cur = mv

def getMin():
  return H[1]

def add(x):
  sz += 1
  H[sz] = x
  siftUp(sz)

def extractMin():
  swap(H[1], H[sz])
  sz -= 1
  siftDown(1)
  return H[sz + 1]
```

1. `Count(l, r, x)`
   Сделайте предподсчет за $\mathcal{O}(n \log n)$, чтобы за $\mathcal{O}(\log n)$ отвечать на запросы вида: “сколько раз число x встречается на отрезке $[l..r]$”?

2. **Рваные кеды**
   Есть n верёвок, которые можно резать, каждая имеет целую длину l_i. Нужно получить k одинаковых кусков максимальной целой положительной длины (можно оставлять неиспользованные обрезки). $\mathcal{O}(n \log l_{\max})$.

3. **k-Merge**
   Есть k отсортированных массивов. В сумме массивы содержат n элементов. Слить массивы за $\mathcal{O}(n \log k)$.

4. **Интермеццо**

   (a) Вычислите $\sum\limits_{i=0}^{\infty} \frac{1}{2^i}$.

   (b) Вычислите $\sum\limits_{i=0}^{\infty} \frac{i}{2^i}$.

5. **Апгрейды** `Build`, `HeapSort`
   Пусть в массиве H записаны какие-то объекты (ключи) в произвольном порядке (т. е. свойство кучи может не выполняться).

   (a) Что делает следующий код? Оцените время работы.

```
sz = n
for i in range[1, n]:
  siftUp(i)
```

(b) Что делает следующий код? Оцените время работы.

```
sz = n
for i in range[n, 1]:
  siftDown(i)
```

(c) Опишите Inplace HeapSort, то есть сортировку кучей, использующую $\mathcal{O}(1)$ дополнительной памяти.

6. **Частичная сортировка**
Дан массив длины n, выдать в порядке возрастания наименьшие k элементов за $\mathcal{O}(n+k\log n)$. (Другими словами, выдать префикс длины k отсортированной версии массива.)

7. Придумайте структуру данных, которая умеет делать `add(x)`, `getMedian()`, `extractMedian()` — все за $\mathcal{O}(\log n)$.

   Медиана множества — это такой элемент, что половина оставшихся элементов меньше его, а другая половина — больше. Другими словами, это элемент, стоящий посередине в отсортированном порядке (причём такое определение подходит и для мультимножества).

   Если n нечётно, медиана определена однозначно. Если n чётно, то существуют разные подходы (например, полусумма двух «кандидатов»). В этой задаче для определённости мы выбираем меньшего из «кандидатов». Например, медиана $\{1, 1, 6, 3, 4, 7\}$ — это число 3, а медиана $\{1, 6, 3, 5, 4\}$ — это число 4.

8. Дана обычная бинарная куча (с минимумом в голове), требуется узнать k-й минимум.

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

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

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

9. 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 ключей.

10. Модифицируйте операцию `siftDown` для бинарной кучи так, чтобы она по-прежнему работала за $\mathcal{O}(\log n)$, но при этом делала лишь $\log_2 n + \mathcal{O}(\log\log n)$ сравнений ключей.

2.2 Домашнее задание

Запись **(2.5)** означает, что за полное решение этой задачи или пункта даётся 2.5 балла. Запись (*) означает, что задача дополнительная: за её решение даются баллы, но она не учитывается в знаменателе оценки.
Не забывайте смотреть на задачи, которые обсуждались в классе! В них много полезных идей.
В некоторых задачах ограничен размер дополнительной памяти (то есть используемой памяти без учёта памяти для входа). В частностни, использовать $\mathcal{O}(1)$ дополнительной памяти неформально означает, что разрешено использовать лишь константное количество новых переменных, а создавать новые массивы (или структуры) неконстантного размера запрещено.

1. **(1)** Есть m стойл с координатами $x_1, \ldots, x_m$ и n коров. Расставить коров по стойлам (не более одной в стойло) так, чтобы минимальное расстояние между коровами было максимально. $\mathcal{O}(m(\log m + \log x_{\max}))$.

2. **(1.5 + 0.5)** Даны два массива a и b длины n, сгенерировать все попарные суммы $a_i + b_j$ в сортированном порядке. Складывать все суммы в один массив необязательно, можно печатать ответ постепенно по ходу выполнения.

   (a) **(0.5)** За $\mathcal{O}(n^2 \log n)$.

   (b) **(0.5)** За $\mathcal{O}(n^3)$ с использованием $\mathcal{O}(n)$ дополнительной памяти.

   (c) **(0.5)** За $\mathcal{O}(n^2 \log n)$ с использованием $\mathcal{O}(n)$ дополнительной памяти.

(d) **(+0.5)** `(*)` За $\mathcal{O}(n^3)$ с использованием $\mathcal{O}(1)$ дополнительной памяти.

Разрешается изменять исходные массивы. (В предыдущих пунктах тоже разрешается, но это ни на что не влияет: там есть хотя бы линейный запас по памяти, поэтому их можно просто скопировать.)

3. **(1)** Даны два отсортированных массива длины n, которые нельзя модифицировать. Найдите k-ю порядковую статистику в объединении массивов (то есть элемент, который окажется на k-й позиции, если массивы слить), используя $\mathcal{O}(1)$ дополнительной памяти.

(a) **(0.5)** За $\mathcal{O}(\log^2 n)$.

(b) **(0.5)** За $\mathcal{O}(\log n)$.

3 Шустрая сортировка и порядковые статистики

3.1 Практика

```
def qsort(a):
  n = len(a)
  if n <= 1:
    return a
  lx = []; mx = []; rx = []
  x = a[randint from [0, n)]
  for v in a:
    if v < x:
      lx.append(v)
    elif x < v:
      rx.append(v)
    else:
      mx.append(v)
  return qsort(lx) + mx + qsort(rx)
```

1. **Какую сортировку применить?**
   Инверсией в массиве чисел $a[\ldots]$ называется такая пара индексов i, j, что $i < j$, но $a_i > a_j$. Дан массив из n различных элементов. Требуется найти число инверсий за $\mathcal{O}(n \log n)$.

2. **Задача о флаге Нидерландов**
   Робот Иван Семёныч 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. У робота есть лишь $\mathcal{O}(1)$ памяти. Помогите Ивану Семёнычу отсортировать пирожки по типу: сначала первый тип, потом второй, потом третий. Время работы — $\mathcal{O}(n)$.

3. **Inplace QuickSort3**
   Пользуясь предыдущей задачей, постройте inplace-версию `QuickSort`, то есть такую, которая не использует дополнительных массивов.

```
def qsort3_inplace(a, l, r): # сортируем полуинтервал a[l, r)
  if r - l <= 1:
    return
  x = a[randint from [l, r)]

  # Partition
  # ...
  # a[l:i) < x
  # a[i:j) == x
  # a[j:r) > x

  qsort3_inplace(a, l, i)
  qsort3_inplace(a, j, r)
```

4. **k-я порядковая статистика**
   Задача поиска k-й порядковой статистики формулируется своим простейшим решением:

```
def order_statistic(a, k):
  a.sort()
  return a[k]
```

   В частности, медиана — это $n/2$-я порядковая статистика.

   Алгоритм выше работает за $\mathcal{O}(n \log n)$, наша цель — построить алгоритм за $\mathcal{O}(n)$. Считаем, что в массиве все числа различны.

   (a) Постройте вероятностный алгоритм поиска k-й порядковой статистики со средним (по случайным битам) временем работы $\mathcal{O}(n)$. Подсказка: модифицируйте алгоритм `QuickSort`.

   (b) Докажите оценку на среднее время работы этого алгоритма (это на порядок проще, чем оценка для `QuickSort`). Подсказка: в каком случае мы достаточно удачно выбрали опорный элемент? С какой вероятностью это происходит? Каково матожидание числа шагов, после которого такое разбиение произойдёт один раз?

5. **(*) k-я порядковая статистика детерминированно**

   (a) Постройте детерминированный (то есть без использования случайных битов) алгоритм поиска k-й порядковой статистики с гарантированным временем работы $\mathcal{O}(n)$. Докажите оценку на время работы.

   (b) Постройте детерминированную версию алгоритма `QuickSort` с гарантированным временем работы (то есть временем работы в худшем случае) $\mathcal{O}(n \log n)$. (Такой алгоритм получается с большой константой, поэтому непрактичен.)

6. **Опять частичная сортировка**
   Дан массив длины n, выдать в порядке возрастания наименьшие k элементов за $\mathcal{O}(n+k \log k)$. (Другими словами, выдать префикс длины k отсортированной версии массива.)

   Заметим, что в прошлый раз мы научились решать эту задачу за $\mathcal{O}(n + k \log n)$ с помощью кучи.

7. Пусть задан массив A из $n = a \cdot k$ различных чисел. Требуется разбить массив на k частей по a элементов в каждой так, чтобы любой элемент части i был бы меньше любого элемента части $i+1$ ($\forall i \in [1, k-1]$). $\mathcal{O}(n \log k)$ в среднем.

8. Дан массив из $2 \cdot n - 1$ числа, который нельзя модифицировать. Есть дополнительная память на $n+1$ элемент массива и ещё $\mathcal{O}(1)$ сверху. Требуется найти медиану за $\mathcal{O}(n \log n)$.

9. Даны массив из n чисел и m чисел $p_1, p_2, \ldots p_m$, нужно за $\mathcal{O}(n \log m + m)$ для каждого i найти p_i-ю порядковую статистику.

3.2 Домашнее задание

1. `[1, 2, 3]` **(1)** Дан набор из n пар гаек и болтов, в разных парах размеры гаек и болтов различны. Гайки и болты перемешаны. Требуется для каждой гайки найти соответствующий болт. Сравнивать можно только болты с гайками (сравнить две гайки между собой, или два болта между собой невозможно). $\mathcal{O}(n \log n)$ в среднем.

2. `[1]` **(1)** Оцените время работы (в терминах Θ) детерминированного алгоритма поиска порядковой статистики (задача 5 из классной работы), если вместо пятерок разбивать элементы на

   (a) **(0.5)** семёрки.

   (b) **(0.5)** тройки.

3. `[2, 3]` **(1)** Даны 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.

4. `[1, 2, 3]` **(1.25)** Дан массив `A[1..n]` из n различных чисел. Массив не обязательно отсортирован. Требуется найти k ближайших к медиане элементов за линейное время. Решить для двух метрик.

   (a) **(0.75)** По позиции в отсортированном массиве.

   $$d(x, \texttt{median}) = |\texttt{pos}(x) - \texttt{pos}(\texttt{median})|,$$

   где $\texttt{pos}(x)$ — позиция элемента x в отсортированном массиве.

   (b) **(0.5)** По значению.

   $$d(x, \texttt{median}) = |x - \texttt{median}|.$$

5. `(*)` **(+1)** Даны два массива из положительных целых чисел a и b, размер обоих равен n. Выбрать массив p из k различных чисел от 1 до n так, чтобы $\frac{\sum_{i=1}^{k} a_{p_i}}{\sum_{i=1}^{k} b_{p_i}} \to \max$. Время $\mathcal{O}(n \log M)$, где $M = \max(\max(a_i), \max(b_i), n)$. Подсказка: используйте бинпоиск по ответу.

6. (a) `[2, 3]` **(0.75)** Дан массив из $2 \cdot n - 1$ числа, который нельзя модифицировать. Есть дополнительная память на $n + 1$ элемент массива и ещё $\mathcal{O}(1)$ сверху. Требуется найти медиану за $\mathcal{O}(n \log n)$.

   (b) `[1, 2, 3]` **(0.75)** Дана последовательность из n чисел, нужно за один проход и $\mathcal{O}(n)$ времени в среднем найти в ней k минимумов, используя $\mathcal{O}(k)$ дополнительной памяти.

7. `[1, 2, 3]`

   (a) **(0.5)** Докажите, что для поиска максимума в массиве различных чисел потребуется как минимум $n/2$ сравнений.

   (b) **(0.5)** Докажите, что для поиска максимума в массиве различных чисел потребуется как минимум $n-1$ сравнение. (Заметим, что $n-1$ сравнения достаточно, поэтому эта оценка точная.)

8. `(*)` **(+1.5)** Дан массив из $2n$ различных чисел.

   (a) **(+0.5)** Найдите минимальное и максимальное за $3n - 2$ сравнения.

   (b) **(+1)** Докажите, что это точная нижняя оценка, то есть меньшего количества сравнений может не хватить.

4 Сортировки и жадности

4.1 Практика

1. Покажите, что любая сортировка сравнениями, которая верно работает хотя бы на доле $\frac{1}{100^n}$ от всех перестановок, не может работать за $o(n \log n)$ на всех тестах.

2. **Покрытие точек единичными отрезками**
   Дано n точек на прямой. Построить наименьшее множество отрезков, имеющих единичную длину, которые покрывают все данные точки.

3. **Выбор заявок**
   Дано n заявок — полуинтервалов $[l_i, r_i)$ Выбрать наибольшее число заявок, чтобы их полуинтервалы не пересекались.

4. **Мягкие дедлайны**

   В фирму поступило n заказов, у каждого есть время исполнения t_i и жёсткий дедлайн d_i. В каком порядке выполнять заказы, чтобы всё успеть?

5. **(*) Жёсткие дедлайны**

   В фирму поступило n заказов, у каждого есть время исполнения t_i и жёсткий дедлайн d_i. В каком порядке и **какие** выполнять заказы, чтобы успеть **побольше**?

6. **Заказы и время конца**

   В фирму поступило n заказов, которые можно выполнять в произвольном порядке. На выполнение заказа i необходимо время t_i. В каждый момент времени можно работать ровно над одним заказом. Пусть e_i – момент окончания выполнения заказа номер i. Распределите работу над заказами так, чтобы минимизировать $\sum_i e_i$. Время $\mathcal{O}(n \log n)$.

7. **Авторитеты**

   Есть n человек. Человек i готов примкнуть к нашей банде, если наш авторитет хотя бы a_i, при этом он к нашему авторитету прибавит b_i. Наш изначальный авторитет равен A. $a_i, b_i, A \in \mathbb{Z}$. Можем ли завербовать всех людей? $\mathcal{O}(n \log n)$.

8. **Выбор заявок в маршрутке**

   Маршрутка совершает рейс от первой до n-й остановки. В маршрутке m мест для пассажиров. Есть k человек, про каждого заранее известно, что он хочет доехать от остановки s_i до f_i. Проезд для пассажира стоит 1 вне зависимости от расстояния между остановками. Максимизируйте прибыль, при условии, что можно выбирать, кого сажать в маршрутку на каждой остановке. $n, m, k \leq 10^6$.

9. **(*) Идейная**

   Машина тратит единицу топлива на километр, имеет бак объёма k и находится в начале прямой дороги в точке 0. Для всех $i \in \mathbb{N} \cup \{0\}$ на i-м километре дороги есть заправочная станция со своей положительной ценой c_i. Определите за время $\mathcal{O}(n)$, как проехать n километров за минимальную стоимость.

10. **(*) Подпоследовательности и сортировки**

    Даны скобочные последовательности из круглых скобок. в каком порядке их склеить, чтобы получилась правильная?

11. **(*) Станки и сортировки**

    Имеется n деталей и два станка. Каждая деталь должна сначала пройти обработку на первом станке, затем — на втором. При этом i-ая деталь обрабатывается на первом станке за a_i времени, а на втором — за b_i времени. Каждый станок в каждый момент времени может работать только с одной деталью. Требуется составить такой порядок подачи деталей на станки, чтобы итоговое время обработки всех деталей было бы минимальным. $\mathcal{O}(n \log n)$.

4.2 Домашнее задание

1. **(0.5)** Докажите, что `extractMin` в куче не может работать за время $o(\log n)$.

   Пояснение: нужно доказать, что невозможно написать функцию `extractMin` таким образом, чтобы время её работы было $o(\log n)$, где n — размер кучи. Требования на функцию `extractMin` следующие. Пусть у нас есть произвольный массив H с выполненным свойством кучи, имеющий минимум m. `extractMin` может менять H, но в конце должен оставить массив с выполненным свойством кучи, в которой на один элемент меньше — а именно, на элемент m. В конце работы `extractMin` возвращает m.

2. В свободное время Анка-пулемётчица любит сортировать патроны по серийным номерам. Вот и сейчас она только разложила патроны на столе в строго отсортированном порядке, как 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 несильно. Каждый патрон отклонился от своей позиции не более чем на k. Всего патронов n. Помогите Анке отсортировать патроны.

   (a) **(0.5)** Отсортируйте патроны за $\mathcal{O}(nk)$.

   (b) **(0.5)** Отсортируйте патроны за $\mathcal{O}(n \log k)$

   (c) **(1)** Докажите нижнюю оценку на время сортировки $\Omega(n \log k)$.

3. **(1) Белоснежка и гномы, которые не хотят спать**
   Даны n гномов. Если i-го гнома укладывать спать a_i минут, он потом спит b_i минут. Можно ли сделать так, чтобы в какой-то момент все гномы спали? $\mathcal{O}(n \log n)$.

4. **(0.75 + 0.75) Степени и антиклики**
   Будем называть *независимым множеством* или *антикликой* подмножество попарно несмежных вершин графа. Пусть в графе G есть n вершин, а максимальная степень равна d. Найдите в нём независимое множество размера хотя бы

   a) **(0.75)** $\frac{n}{d+1}$ за время $\mathcal{O}(n)$

   b) **(+0.75)** `(*)` $\sum\limits_{v \in V(G)} \frac{1}{\deg(v)+1}$ за время $\mathcal{O}(n \log n)$

   Считайте, что граф уже дан в памяти в виде массива, где для каждой вершины хранится список её соседей.

5. **(+0.75)** `(*) [1, 3]` **Идейная**
   Машина тратит единицу топлива на километр, имеет бак объёма k и находится в начале прямой дороги в точке 0. Для всех $i \in \mathbb{N} \cup \{0\}$ на i-м километре дороги есть заправочная станция со своей положительной ценой c_i. Определите за время $\mathcal{O}(n)$, как проехать n километров за минимальную стоимость.

6. **(+0.75)** `(*)` **Подпоследовательности и сортировки**
   Даны скобочные последовательности из круглых скобок. в каком порядке их склеить, чтобы получилась правильная?

7. **(+0.75)** `(*)` **Станки и сортировки**
   Имеется n деталей и два станка. Каждая деталь должна сначала пройти обработку на первом станке, затем — на втором. При этом i-ая деталь обрабатывается на первом станке за a_i времени, а на втором — за b_i времени. Каждый станок в каждый момент времени может работать только с одной деталью. Требуется составить такой порядок подачи деталей на станки, чтобы итоговое время обработки всех деталей было бы минимальным. $\mathcal{O}(n \log n)$.

5 Демоническое программирование

5.1 Практика

1. Дан массив из n целых чисел и число d. Найти подпоследовательность максимальной длины с условием, что соседние элементы в ней должны отличаться не более чем на d.

2. Дан массив из n целых чисел, число d и число k. Найти подпоследовательность длины k с максимальной суммой элементов при условии, что соседние элементы в ней должны отличаться не более чем на d.

3. **НОП**
Даны две последовательности длины n. Найдите наибольшую общую подпоследовательность этих последовательностей за $\mathcal{O}(n^2)$.

4. **Рюкзаки**
Даны n предметов. У каждого есть цена v_i и вес w_i. Найти max цену, которую можно набрать предметами суммарного веса $\leq S$. Время $\mathcal{O}(nS)$.

   (a) Каждый предмет можно брать сколько угодно раз. Память $\mathcal{O}(S)$.

   (b) Каждый предмет можно брать один раз. Память $\mathcal{O}(nS)$.

   (c) Каждый предмет можно брать один раз. Память $\mathcal{O}(S)$.

   (d) Те же пункты, но хотим набрать вес ровно S.

   (e) Те же пункты, но цен нет и хотим набрать вес ровно S.

5. **Лабиринт**
Дана матрица, в каждой клетке лежат монетки разной ценности. Некоторые клетки непроходимые. За один ход можно сместиться вверх или вправо. Рассмотрим все пути из левой нижней клетки в верхнюю правую.

   (a) Найти число таких путей (по модулю $10^9 + 7$).

   (b) Найти путь, сумма ценностей монет на котором максимальна/минимальна.

6. У профессора есть k яиц и n-этажное здание. Он хочет узнать такое максимальное x, что если яйцо бросить с x-го этажа, оно не разобьётся. Неразбившиеся яйца можно переиспользовать. Минимизировать число бросков в худшем случае.

   (a) $\text{poly}(n, k)$.

   (b) $\mathcal{O}(kn^2)$.

   (c) (*) $\mathcal{O}(kn \log n)$.

7. **НВП за $\mathcal{O}(n \log n)$**
Найдите максимальную возрастающую подпоследовательность за $\mathcal{O}(n \log n)$. Найти длину и восстановить ответ.

8. **НОП revisited**
Даны две последовательности длины n. Найдите наибольшую общую подпоследовательность этих последовательностей за $\mathcal{O}(n \log n)$, если известно, что в одной из последовательностей все элементы различны.

9. Найти максимальное по весу паросочетание за $\mathcal{O}(n)$ на

   (a) дереве из n вершин,

   (b) (*) простом цикле из n вершин,

   (c) (*) связном неориентированном графе из n вершин и n рёбер.

   Веса на рёбрах.

5.2 Домашнее задание

1. **(1)** `[3]` **Рюкзаки**
   Даны n предметов. У каждого есть цена v_i и вес w_i. Найти max цену, которую можно набрать предметами суммарного веса $\le S$. Время $\mathcal{O}(nS)$.

   (a) **(0.75)** Каждый предмет можно брать один раз. Память $\mathcal{O}(nS)$.

   (b) **(0.25)** Каждый предмет можно брать один раз. Память $\mathcal{O}(S)$.

2. **(1)** `[3]` **Лабиринт**
   Дана матрица, в каждой клетке лежат монетки разной ценности. Некоторые клетки непроходимые. За один ход можно сместиться вверх или вправо. Рассмотрим все пути из левой нижней клетки в верхнюю правую.

   (a) **(0.5)** Найти число таких путей (можно считать ответ по модулю 10^9+7, чтобы в процессе вычислений не возникали длинные числа).

   (b) **(0.5)** Найти путь, сумма ценностей монет на котором максимальна/минимальна.

3. **(1 + 0.5)** **И снова подпоследовательности**
   Дан массив из n натуральных чисел: $a_1, \ldots, a_n$. Выберите подпоследовательность $i_1 \le \ldots \le i_k \in \{1, \ldots, n\}$ такую, что $l \le |i_j - i_{j-1}| \le r$ и $\sum_{j=1}^{k} a_{i_j} \to \max$.

   (a) **(1)** За $\mathcal{O}(n^2)$.

   (b) **(+0.5)** `(*)` За $\mathcal{O}(n)$.

4. **(1)** **Longest Common Prefix (LCP)**
   Дана строка `s[0:n]` длины n.
   Префикс строки s — это строка `s[0:i]` для какого-нибудь i (такой префикс называется i-м префиксом).
   Суффикс строки s — это строка `s[i:n]` для какого-нибудь i (такой суффикс называется i-м суффиксом).

   Для каждой пары (i, j) найти длину наибольшего общего префикса i-го и j-го суффиксов строки s. $\mathcal{O}(n^2)$.

   Формально, $\mathrm{LCP}_{i,j} = \max\{k \mid 0 \le k \le n,\ i + k \le n,\ j + k \le n$ и `s[i:i+k] == s[j:j+k]`$\}$.

5. **(1)** Дан набор нечестных монеток с вероятностью выпадения орла $p_1, p_2, \ldots, p_n$. Требуется посчитать вероятность выпадения ровно k орлов за $\mathcal{O}(n \cdot k)$. Операции над числами считать выполнимыми за $\mathcal{O}(1)$.

6. **(1)** `[1, 2]` Пусть есть n подарков разной натуральной стоимости и три поросёнка. Нужно раздать подарки как можно честнее (так, чтобы минимизировать разность суммарной стоимости подарков самого везучего поросёнка и самого невезучего). Придумайте алгоритм решения данной задачи за $\mathcal{O}(nW^2)$, где W — суммарная стоимость подарков.

7. **(1)** `[1, 2]` На билете есть $2n$-значный номер. Билет считается счастливым, если сумма первых n цифр совпадает с суммой последних n цифр. По заданному числу n требуется найти количество счастливых $2n$-значных билетов за $\mathcal{O}(n^2)$. Считайте, что стандартные арифметические операции над числами выполняются за $\mathcal{O}(1)$.

   (Более формально, можно считать, что нужно вывести не ответ целиком, а ответ по модулю $10^9 + 7$, чтобы в процессе вычислений не возникали длинные числа.)

8. **(+1.5)** `(*)` У профессора есть k яиц и n-этажное здание. Он хочет узнать такое максимальное x, что если яйцо бросить с x-го этажа, оно не разобьётся. Неразбившиеся яйца можно переиспользовать. Минимизировать число бросков в худшем случае.

   (a) **(+1)** $\mathcal{O}(kn^2)$.

   (b) **(+0.5)** $\mathcal{O}(kn \log n)$.

6 Динамика-2 и Хаффман

6.1 Практика

1. **(!) Разбиение на палиндромы**
   Разбить строку на минимальное число палиндромов за $\mathcal{O}(n^2)$ времени.
   (a) $\mathcal{O}(n^2)$ памяти.
   (b) **(*)** $\mathcal{O}(n)$ памяти.
2. **(!) Подпоследовательность-палиндром**
   Дана строка длины n. Найти максимальную по длине подпоследовательность, которая является палиндромом. $\mathcal{O}(n^2)$.
3. **Свертка.** Дана строка из латинских букв длины n, нужно ее запаковать в максимально короткую, используя правило $n(S) = \underbrace{SS\ldots S}_{n}$.
   Пример: `NEERCYESYESYESNEERCYESYESYES` $\to$ `2(NEERC3(YES))`.
4. **(!) Паросочетания**
   Дан взвешенный неориентированный граф из n вершин. Найдите максимальное по весу паросочетание.
   (a) $\mathcal{O}(2^n n^2)$.
   (b) $\mathcal{O}(2^n n)$.
5. **(!) Set Cover**
   Даны $A, B_1, \ldots, B_m \subseteq \{0, \ldots, n-1\}$. Выбрать минимальный набор $\{B_{i_j}\}$: $\bigcup B_{i_j} = A$.
   Чтобы везде можно было бесплатно применять битовую магию, предположим $n, m \le 64$.
   (a) $\mathcal{O}(2^m \operatorname{poly}(n))$.
   (b) $\mathcal{O}(2^m)$.
   (c) $\mathcal{O}(2^n m)$.
6. **(!) Хаффман за $\mathcal{O}(n)$.**
   Постройте код Хаффмана за $\mathcal{O}(n)$, если частоты букв уже даны в отсортированном порядке.
7. За какое время работает этот код?

```
for S ⊆ {1,...,n}:
  for T ⊆ S:
    cnt += 1
```

8. **Перевозка товаров**
   Есть грузовик с заданной вместимостью, задача – перевезти n вещей с заданными весами со склада в магазин минимальным числом заездов.
   (a) $\mathcal{O}(3^n \operatorname{poly}(n))$.
   (b) $\mathcal{O}(3^n)$.
   (c) **(*)** $\mathcal{O}(2^n n)$.
9. **Битоническая задача о коммивояжере**
   Найдите во взвешенном графе гамильтонов цикл минимального веса, который удовлетворяет дополнительно следующему свойству: сначала номера посещенных вершин возрастают, а затем убывают. Время $\mathcal{O}(n^2)$ (что? да!).
10. **(*) Пираты**
   Судно атакуют пираты. Для каждого пирата известны его азимут a_i и время t_i, через которое пират приплывёт и совершит непотребство. Однако, у судна есть лазерная пушка, которой оно защищается. У пушки есть начальный азимут a и угловая скорость вращения ω. Пушка уничтожает все объекты, на которые она сейчас направлена. Помогите судну определить порядок уничтожения пиратов за $\mathcal{O}(n^2)$, чтобы не допустить непотребства.

7 Поиск в глубину

7.1 Практика

1. **Просто ориентируй**
   Ориентировать неорграф так, чтобы он стал ацикличным за $\mathcal{O}(V+E)$.

2. **Берегите деревья**
   Ориентировать неорграф так, чтобы он стал сильно связным за $\mathcal{O}(V+E)$.

3. **Разве это не NP-трудно?**
   Дан DAG, найти в нем гамильтонов путь за $\mathcal{O}(V+E)$.

4. **Единственный топсорт**
   Дан DAG, проверить единственность топологической сортировки за $\mathcal{O}(V+E)$.

5. **dfs всемогущий**
   Даны два множества вершин: A и B, за $\mathcal{O}(V+E)$ проверить, есть ли путь из какой-нибудь вершины $a \in A$ в какую-нибудь вершину $b \in B$.

6. **Чётность циклов**
   За $\mathcal{O}(V+E)$ найти в неорграфе какой-нибудь цикл нечётной длины.

7. **В поисках простого цикла**
   a) Найти цикл в орграфе через данное ребро за $\mathcal{O}(E)$.
   b) Найти цикл в орграфе через данную вершину за $\mathcal{O}(E)$
   c) Найти в неорграфе какой-нибудь цикл за $\mathcal{O}(V)$.

8. **Ценность времени**
   Дано подвешенное дерево, нужно с линейным предподсчётом научиться отвечать на запрос «правда ли вершина u лежит в поддереве вершины v» online за $\mathcal{O}(1)$.

9. **Телепортация в дереве**
   Дано корневое дерево и m телепортов. Для каждой вершины v дерева насчитайте самую высокую вершину, куда можно телепортироваться из поддерева v, если разрешается неограниченное число раз спускаться по рёбрам и не более раза телепортироваться.

10. **Мосты**
    Дан связный неорграф, нужно найти в нем за $\mathcal{O}(V+E)$ все мосты.

11. **dfs и веса**
    В стране n аэропортов. Самолёт может сделать перелёт из аэропорта i в аэропорт j, израсходовав $w_{ij} > 0$ горючего. При этом w_{ij} может отличаться от w_{ji}, и $w_{ii} = 0$. Найдите минимальный размер бака, позволяющий добраться самолёту из любого города в любой, возможно с дозаправками.

12. **3-связность**
    Проверить граф на рёберную 3-связность за $\mathcal{O}(VE)$.

13. **(*) Нельзя не пройти**
    Найти в орграфе все вершины, через которые проходит любой путь $a \rightsquigarrow b$. $\mathcal{O}(V+E)$.

7.2 Домашнее задание

1. (1) [2] **dfs всемогущий**
   Даны два множества вершин: A и B, за $\mathcal{O}(V+E)$ проверить, есть ли путь из какой-нибудь вершины $a \in A$ в какую-нибудь вершину $b \in B$.

2. (1) [3] **Ценность времени**
   Дано подвешенное дерево, нужно с линейным предподсчётом научиться отвечать на запрос «правда ли вершина u лежит в поддереве вершины v» online за $\mathcal{O}(1)$. Подсказка: используйте времена входа и выхода.

3. (1) Корневое дерево T на V вершинах представлено массивом из V элементов. Все вершины пронумерованы, и для каждой вершины в массиве указан его родитель. Для корня r значение в массиве равно -1. Требуется определить, как будет выглядеть новое представление дерева, если корень r сменить на корень q. Разрешается использовать $\mathcal{O}(1)$ дополнительной памяти. Менять массив можно. Время $\mathcal{O}(V)$.

4. (1.25) **Лексмин топсорт**
   Найдите лексикографически минимальный из всех топологических порядков. $V, E \leq 10^6$.

5. (1.5) **Пути в дереве**
   Дано дерево $T = \langle V, E \rangle$.
   За $\mathcal{O}(V+E)$ вычислить для каждого ребра, сколько простых путей проходит через него.

6. (+1.5) (*) **Враги и двухпартийная система**
   У каждой вершины не более 3 врагов. Вражда – симметричное отношение. Разбить вершины на 2 доли так, чтобы с вершиной в долю попало не более 1 врага. $\mathcal{O}(V+E)$.

8 Кратчайшие пути

8.1 Практика

1. **Восстановление пути**
   Как вывести сам путь (а не только длину) в алгоритмах BFS, Дейкстры, Форда-Беллмана, Флойда-Уоршелла (2 способа)?

2. Как в алгоритма Форда-Беллмана понять, что в графе есть отрицательный цикл? Подсказка: сделайте ещё одну итерацию внешнего цикла.

3. Как в алгоритма Флойда-Уоршелла понять, что в графе есть отрицательный цикл? Подсказка: посмотрите на $d[i][i]$.

4. Дан произвольный взвешенный орграф, найдите расстояния от s до каждой из вершин (каждое расстояние — число или $\pm\infty$). В графе могут быть отрицательные циклы. $\mathcal{O}(VE)$.

5. Дан произвольный взвешенный орграф, найдите попарные расстояния между вершинами (каждое расстояние — число или $\pm\infty$). В графе могут быть отрицательные циклы. $\mathcal{O}(V^3)$.

6. Пусть в графе есть ребра веса 0 и 1. Придумайте, как найти кратчайшее расстояние от вершины s до остальных за $\mathcal{O}(V + E)$.

7. **Обмен местами**
   Есть ориентированный граф. Для каждой пары вершин a, b определена функция $f(a, b)$.
   Вася и Петя стоят в вершинах v и p, соответственно, и хотят поменяться местам, не оказываясь ни в какой момент времени в паре вершин с $f(a, b) < d$. За какое минимальное число ходов они могут это сделать? Ход — один из них переходит в смежную вершину. $\mathcal{O}(VE)$.

8. **Число кратчайших путей**
   Дан орграф, все $w_e > 0$. Дана стартовая вершина s, нужно для каждой вершины v найти число кратчайших путей из s в v. $\mathcal{O}(E \log V)$.

9. **Запросы к Роберту**
   Предподсчет за $\mathcal{O}(V^3)$ и запрос $\langle a, b, e\rangle$ за $\mathcal{O}(1)$ — существует ли кратчайший путь из a в b, проходящий через ребро e?

10. **Кратчайший путь через достопримечательности**
    Дан орграф. Найти кратчайший путь, проходящий по всем k выделенным вершинам. $k \le 10$.

11. **k-bfs**
    Пусть в графе все ребра имеют целый вес из $[1, k]$. Придумайте, как найти кратчайшее расстояние от вершины s до остальных за $\mathcal{O}(k(V+E))$. А за $\mathcal{O}(kV + E)$?

12. **Потенциалы**
    Пусть дан взвешенный граф G (возможно, с циклами отрицательного веса). На вершинах этого графа определим функцию $\phi(v)$ — потенциал. Заменим вес каждого ребра $w(u, v)$ на $w'(u, v) = w(u, v) + \phi(u) - \phi(v)$. Докажите, что кратчайшим путям между двумя вершинами в графе с весами w' будут однозначно соответствовать кратчайшие пути в графе с весами w.

13. **Кратчайшие пути между всеми парами вершин**
    Пусть во взвешенном графе G нет циклов отрицательной стоимости. Докажите, что если в качестве потенциала взять кратчайшее расстояние от некоторой вершины s, то все веса w' получатся неотрицательными (если соответствующие расстояния конечны).

    Покажите, как найти матрицу расстояний в графе с отрицательными весами за $\mathcal{O}(VE \log V)$.

14. **Откуда берутся графы?**
    Дана система из m неравенств на n переменных x_i.
    Каждое неравенство имеет вид $x_i - x_j \le \delta_{ij}$.

    (a) Найти решение системы или сказать, что его не существует, за $\mathcal{O}(n \cdot m)$.

(b) Пусть все $\delta_{ij} \geq 0$, решить задачу за $o(n \cdot m)$. Слишком просто? Тогда $\sum_i x_i \to \max$.

15. **Расстояние – это не только сумма**
Для каждой пары вершин в графе найти $w[a, b]$ – такой минимальный вес, что из a в b есть путь по рёбрам, вес которых не больше $w[a, b]$. $\mathcal{O}(V^3)$.

16. **Речной граф**
Даны две параллельных прямых (река). В реке есть n островов (точек). Мы хотим провести по реке корабль, представляющий собой открытый круг радиуса R, так, чтобы он не задел ни одного острова. Найти максимальный R, при котором это еще возможно, за $\mathcal{O}(n^2 \log n)$.

17. **Обмен валют**
Есть n валют и m обменников. i-й обменник предлагает менять валюту a_i на валюту b_i по курсу c_i/d_i. Можно ли, используя сколь угодно большие начальные сбережения и данные m обменников, сломать финансовую систему и бесконечно обогащаться? Считается, что у обменников есть бесконечное количество денег целевой валюты.

    (*) А теперь тоже самое с комиссией: x_i единиц валюты a_i перейдут в $(x_i - s_i)\frac{c_i}{d_i}$ единиц валюты b_i.

18. **Количество путей**
Найти количество путей (необязательно простых) в графе за $\mathcal{O}(V^3 \log k)$
    a) между всеми парами вершин длины ровно k.
    b) между парой вершин длины $\leq k$.

8.2 Домашнее задание

1. **(1.5) Дерево Штейнера**
   Дан взвешенный неориентированный граф $G = \langle V, E \rangle$. Веса рёбер неотрицательны. В графе есть подмножество вершин T, которые мы назовем терминалами. Минимальное дерево Штейнера – это связный подграф графа G минимального веса, содержащий все терминалы. Требуется найти такой подграф и доказать, что он является деревом.

   (a) **(0.5)** Доказать, что искомый подграф – дерево. (Чуть точнее: существует дерево, которое является связным подграфом минимального веса, содержащим все терминалы.)

   (b) **(0.5)** Пусть $|T| = 3$, решить за $\mathcal{O}(E \log V)$.

   (c) **(0.5)** Пусть $|T| = 4$, решить за $\mathcal{O}(V^3)$.

2. **(0.75) Расстояния в меняющемся графе**
   Нужно научиться на запрос «уменьшился вес ребра» за $\mathcal{O}(V^2)$ пересчитывать матрицу расстояний. Считайте, что в графе не было и не появилось отрицательных циклов.

3. **(1.75) Потенциалы Джонсона**
   Пусть во взвешенном графе G нет циклов отрицательной стоимости.

   (a) **(0.75)** На вершинах этого графа определим функцию $\phi(v)$ — потенциал. Заменим вес каждого ребра $w(v,u)$ на $w'(v,u) = w(v,u) + \phi(v) - \phi(u)$. Докажите, что кратчайшим путям между двумя вершинами в графе с весами w' будут однозначно соответствовать кратчайшие пути в графе с весами w.

   (b) **(0.5)** Возьмём в качестве потенциала кратчайшее расстояние от некоторой вершины s, то есть для каждой вершины v определим $\phi(v) \coloneqq \rho(s,v)$. Докажите, что все веса w' получатся неотрицательными (если соответствующие расстояния конечны).

   (c) **(0.5)** Найдите матрицу попарных расстояний в графе с отрицательными весами за $\mathcal{O}(VE \log V)$.

4. **(+1) (*) bfs и длинная очередь**
   Постройте матрицу $n \times n$, состоящую из клеток-стенок и пустых клеток, на которой при запуске `BFS` из какой-то клетки максимальный размер очереди будет $\omega(n)$. Ходить можно между клетками, смежными по стороне.

5. **(+1) (*) Форд-Беллман и число итераций**
   Пусть на вершинах графа задан порядок: $v_1, v_2, \cdots, v_n$. Пусть алгоритм Беллмана-Форда на каждой стадии рассматривает ребра в таком порядке: сначала ребра, ведущие из меньшей вершины в большую (в порядке возрастания номера исходящей вершины), а потом ребра, ведущие из большей вершины в меньшую (в порядке убывания номера исходящей вершины). Докажите, что если в графе нет циклов отрицательного веса, то алгоритм найдет все кратчайшие пути из v_1 за $\frac{n}{2}$ итераций.

6. **(+1) (*) Модульный граф**
   Пусть длина пути определяется как сумма весов всех ребер по модулю n. Найти кратчайший путь за $\mathcal{O}((V + E) \cdot n)$.

9 Минимальные остовные деревья

9.1 Практика

1. **Минимальный цикл**
   Найти во взвешенном неорграфе такой цикл, что максимальный вес ребра этого цикла минимален. $\mathcal{O}((V+E)\log E)$.

2. **Аэропутешествия**
   В стране n аэропортов. Самолет может сделать перелет из аэропорта i в аэропорт j, израсходовав w_{ij} горючего. При этом $w_{ij} = w_{ji}$, и $w_{ii} = 0$. Требуется найти минимальный размер бака, позволяющий добраться самолету из любого города в любой, возможно с дозаправками. У нас уже было решение с бинпоиском по ответу и dfsom за $\mathcal{O}(n^2 \log n)$. Теперь решите за $\mathcal{O}(n^2)$.

3. **Приближение коммивояжера**
   Дан полный неориентированный взвешенный граф с неотрицательными весами. Мы бы хотели решить задачу коммивояжёра на нём, но это слишком долго (мы умеем за $\mathcal{O}(2^n n^2)$). Поэтому мы хотим построить гамильтонов путь, вес которого будет не больше, чем в два раза хуже пути коммивояжёра (говорят, что мы ищем *2-приближение* для задачи коммивояжёра).

   Дано дополнительное условие на веса рёбер: для них выполняется неравенство треугольника, то есть $w(a,b) + w(b,c) \geq w(a,c)$ для любой тройки вершин a, b, c.

   Подсказки:

   a) сравните вес MST и вес оптимального пути коммивояжёра;
   b) постройте MST и обойдите его естественным образом.

4. **Online двудольность**
   Дан неорграф. В него в online добавляются рёбра.
   После каждого добавления говорить, двудолен ли граф.

   a) $\mathcal{O}(m \log n)$ через перекрашивание компонент.
   b) Быстрее через DSU.

5. **Чётность**

   В каждой клетке прямой записано число 0 или 1. Поступает информация: чётность количества единиц на отрезке $[L_i, R_i]$, найти первый запрос, после которого данные противоречивы.

6. **Единственность MST**

   a) Пусть все ребра графа имеют различный вес. Докажите, что минимальное покрывающее дерево единственно.
   b) Проверить, что минимальное по весу остовное дерево единственно. $\mathcal{O}(E \log V)$.

7. **MST в меняющемся графе**
   Дан взвешенный связный неориентированный граф $G = \langle V, E \rangle$ и некоторое минимальное остовное дерево на нём. Пусть некоторого ребра $e \in E$ изменился вес. По графу, остовному дереву, ребру e и его новому весу найдите новое минимальное остовное дерево за $\mathcal{O}(V+E)$.

8. **Второе по минимальности MST**

   Найдите за полиномиальное время второе по весу остовное дерево в неографе.

9. **Казалось бы, причём здесь СНМ?**

   У нас есть массив длины n, мы хотим выполнить m запросов вида «покрасить отрезок $[l_i, r_i]$ массива в цвет c_i» и вывести, что получилось в конце. Решите за $\mathcal{O}(n \log m)$.

   Подсказка: попытайтесь выполнять запросы, начиная с последнего.

10. **Алгоритм Борувки**

Рассмотрим следующий алгоритм поиска минимального покрывающего дерева:

```
while (в графе больше одной вершины):
```

a) Для каждой вершины найдем самое легкое инцидентное ей ребро и добавим его в множество S (одно и то же ребро может быть выбрано дважды).
b) Добавим все ребра из множества S в ответ.
c) Стянем граф по ребрам из S.

Докажите, что такой алгоритм найдет минимальное покрывающее дерево в случае, если веса всех ребер в графе различны, при этом время работы будет $\mathcal{O}(E \log V)$.

Придумайте, как модифицировать алгоритм, если возможны равные веса.

Ценность алгоритма по сравнению с Краскалом и Примом в том, что часто получается $\mathcal{O}(E)$. Постройте пример для алгоритма Борувки, на котором он делает $\Theta(1)$ фаз.

Занимательный факт. Борувка в худшем случае работает за $\mathcal{O}(E \log_{E/V} V)$.
Обычные Прим и Краскал дольше. Прим с d-ичной кучей внутри столько же.

11. (*) **Все деревья связаны**

Пусть дан взвешенный связный неорграф $G = \langle V, E\rangle$ с выделенной вершиной s. Все веса положительны и различны. Могут ли какое-либо минимальное покрывающее дерево в G и какое-либо дерево кратчайших путей из s не иметь ни одного общего ребра? Если да, приведите пример. Если нет, докажите, что такого не может быть.

9.2 Домашнее задание

1. **(1 + 0.75) Единственность MST**

   (a) **(0.5)** Докажите, что алгоритм Краскала может найти любое из минимальных остовных деревьев. Более формально, пусть дан граф G и MST T в нём. Тогда рёбра можно упорядочить по неубыванию веса так, чтобы алгоритм Краскала выдал дерево T в качестве ответа.

   (b) **(0.5)** Пусть все ребра графа имеют различный вес. Докажите, что минимальное покрывающее дерево единственно.

   (c) **(+0.75)** `(*)` Дан граф. Проверить, что минимальное по весу остовное дерево единственно. $\mathcal{O}(E \log V)$.

2. **(1) Краскал наоборот**
   Пусть дан связный взвешенный неорграф, будем рассматривать его ребра в порядке невозрастания веса и удалять текущее ребро, если связность графа при этом не нарушается. Докажите, что этот алгоритм находит минимальный остов, или придумайте контрпример.

3. **(1) Насколько сжатие путей быстрое?**
   Рассмотрим реализацию СНМ с операциями следующего вида:

   - `join(root, v)` – подвешивает корень `root` какого-то дерева к произвольной (не обязательно корневой) вершине другого дерева, то есть выполняет операцию `p[root] = v`.
   - `get(v)` – стандартный `get` с эвристикой сжатия путей;

   Теперь поступает m запросов. Сначала поступает первый блок запросов – в нём запросы только первого типа. Затем поступает второй блок запросов – в нём запросы только второго типа.

   Докажите, что обработка этих запросов такой реализацией СНМ работает за $\mathcal{O}(n + m)$, где n – число элементов.

4. **(1 + 0.5) Подсчёт опыта**
   Есть n игроков и m запросов. Изначально у всех игроков 0 опыта и каждый из них состоит в клане, состоящим из него одного. Запросы бывают трёх типов:

   - `join X Y` – объединить кланы, в которые входят игроки X и Y (если они уже в одном клане, то ничего не меняется).
   - `add X V` – добавить V единиц опыта всем участникам клана, в который входит игрок X.
   - `get X` – вывести текущий опыт игрока X.

   Решить за время:

   (a) **(1)** $\mathcal{O}((n + m) \log(n))$.

   (b) **(+0.5)** `(*)` $\mathcal{O}((n + m)\alpha(n))$.

10 Поиск подстроки в строке

10.1 Практика

Хешами пока пользоваться нельзя.
Строки обычно сравнивают *лексикографически* (например, так слова упорядочены в словаре). Строка s лексикографически меньше строки t, если:

- либо s является собственным (то есть не совпадающим со всей строкой) префиксом строки t;
- либо для некоторого числа $k \geq 0$ выполнено `s[0..k) == t[0..k)` и `s[k] < t[k]`.

Пусть дана строка `s[0..n)`.
i-м суффиксом s называют `s[i..n)`, i-м префиксом s называют `s[0..i]`.
Выпишем все n суффиксов строки s и отсортируем их в лексикографическом порядке. *Суффиксный массив* — это номера суффиксов в этом отсортированном массиве. Например, для `s = "acaba"` отсортированный массив суффиксов будет `["a "aba "acaba "ba "caba"]`, а суффиксный массив будет `[4, 2, 0, 3, 1]`.

1. **Периоды строки**
   Период строки s – число $T > 0$ такое, что для любого индекса $0 \leq i < (|s| - T)$ выполнено $s[i] = s[i + T]$.

   Найти все периоды строки за $\mathcal{O}(n)$. Решить с помощью Z-функции и с помощью префикс-функции.

2. **Позиция в суффиксном массиве**
   Найдите позицию строки (то есть нулевого суффикса) в её суффиксном массиве. $\mathcal{O}(n)$.

3. **Автомат КМП**
   На лекции мы научились искать образец s в тексте t за время $\mathcal{O}(|s| + |t|)$ и память $\mathcal{O}(|s|)$. Наш алгоритм даже готов принимать символы текста t по одному и онлайн отвечать, есть ли сейчас вхождение.

   Недостаток текущего алгоритма в том, что отдельный символ текста мы можем обрабатывать довольно долго, хоть и амортизированное время $\mathcal{O}(1)$. Улучшите алгоритм, чтобы обрабатывать любой символ текста за честное $\mathcal{O}(1)$. В чём недостаток получившегося решения?

4. **Строки в дереве**
   Дано подвешенное дерево, на ребрах которого написаны непустые строки суммарной длины n, и дан образец p (все над алфавитом Σ). Найдите все соответствующие вхождениям p отрезки вертикальных путей за $\mathcal{O}(n + |p| \cdot |\Sigma|)$.

5. **Одна ошибка и ты ошибся**
   Научиться искать образец в строке, если допустимо различие в один символ между образцом и найденной подстрокой. $\mathcal{O}(n + m)$.

6. **Различные подстроки**
   Найдите число различных подстрок строки за $\mathcal{O}(n^2)$.

7. **(*) Восстановление строки**
   За $\mathcal{O}(n)$ восстановить строку, если дана её

   (a) Z-функция;

   (b) префикс-функция.

   Если таких строк несколько, восстановить любую.

10.2 Домашнее задание

Хешами пока пользоваться нельзя.

1. **(1)** `[3]` **Одна ошибка и ты ошибся**
   Научиться искать образец в строке, если допустимо различие в один символ между образцом и найденной подстрокой. $\mathcal{O}(n+m)$.

2. **(0.75) Закрепление**
   Дана строка s. Её циклические сдвиги – это строки вида `s[i..n) + s[0..i)` для $0 \le i < n$ (сама строка s тоже является циклическим сдвигом s).

   По строке s определите, верно ли, что она (лексикографически) не больше любого своего циклического сдвига. Время $\mathcal{O}(|s|)$.

3. **(0.75) Префиксы префиксов**
   Дана строка s длины n. Для каждого i от 1 до n найти количество непустых префиксов строки `s[0..i)`, равных суффиксу той же строки `s[0..i)`. Время $\mathcal{O}(n)$.

4. **(1)** $Z \to \pi$
   Дан массив значений Z-функции неизвестной строки. Постройте массив значений префикс-функции этой строки. Саму строку восстанавливать не нужно. Время $\mathcal{O}(n)$.

   Подсказка к одному из решений: модифицируйте алгоритм подсчёта префикс-функции.

   (В обратную сторону легче восстановить строку и посчитать для неё Z-функцию.)

5. **(+1)** `(*)` **Породите две строки**
   Даны строки s и t. Найдите кратчайшую строку p такую, что строка p^∞ (строка p, повторённая бесконечно много раз) содержит s и t как подстроки. Если таких p несколько, найдите любую. Время $\mathcal{O}(|s|+|t|)$.

6. **(+1)** `(*)` **Восстановление строки**
   За $\mathcal{O}(n)$ восстановить строку, если дана её

   (a) **(+0.5)** Z-функция;

   (b) **(+0.5)** префикс-функция.

   Если таких строк несколько, восстановить любую.

11 Хеширование и бор

11.1 Практика

1. **Периоды строки**
   Период строки s – число $T > 0$ такое, что для любого индекса $0 \leq i < (|s| - T)$ выполнено $s[i] = s[i+T]$.

   Найти все периоды строки за $\mathcal{O}(n)$. В прошлый раз научились Z-функцией и префикс-функцией. Теперь хешами.

2. **Различные подстроки**
   Найдите число различных подстрок строки за $\mathcal{O}(n^2)$. В прошлый раз научились Z-функцией. Теперь хешами.

3. **Амнезия**
   Дана строка s длины n. Вы хотите посчитать массив значений Z-функции для всех позиций. Беда в том, что вы забыли алгоритм для подсчёта Z-функции. Используя хеши, посчитайте значения Z-функции за время $\mathcal{O}(n \log n)$.

4. **Сравнение строк на больше/меньше**
   Даны строки s и t длины n. Поступают запросы: (l_1, r_1, l_2, r_2). Нужно сравнить строки $s[l_1 \ldots r_1)$ и $t[l_2 \ldots r_2)$. Однако выдать нужно не просто «равны» или «не равны», а «больше», «меньше» или «равны». $\mathcal{O}(\log n)$ на запрос, $\mathcal{O}(n)$ на предподсчёт.

5. **Суффиксный массив за $\mathcal{O}(n \log^2 n)$**
   Дана строка длины n. Отсортируйте все её суффиксы за $\mathcal{O}(n \log^2 n)$.

6. **Общая подстрока**
   Найти наибольшую общую подстроку двух строк. $\mathcal{O}(n \log n)$.

7. **Поиск словарных слов**
   Даны словарь из слов $s_1, \ldots, s_k$ и текст t. Суммарная длина слов из словаря – L, максимальная – l_{max}. Найти все вхождения всех словарных слов в t за время:

   (a) $\mathcal{O}(L + k|t|)$.

   (b) $\mathcal{O}(|\Sigma|L + l_{max}|t|)$.

   (c) (*) $\mathcal{O}(L + l_{max}|t|)$.

8. **Две ошибки**
   Научиться искать образец в строке, если допустимо различие в два символа между образцом и найденной подстрокой. $\mathcal{O}(n + m)$.

9. **Подматрица в матрице**
   Даны два двухмерных массива A и B – большая картинка и маленькая картинка.
   Найдите точное совпадение B с подпрямоугольником A. За $\mathcal{O}(|A|)$.

10. **Разбиение текста**
    Дан словарь слов суммарной длины L и текст T. Длины слов в словаре не более l. Представьте текст в виде конкатенации минимального количества словарных слов.
    Слова можно использовать более одного раза.

11.2 Домашнее задание

1. **(0.75) Баг в Z-функции**
   Рассмотрим код Z-функции, написанный с ошибкой:

```
z = [0] * n
z[0] = 0
l = r = 0
for i in range(n):
  k = max(min(z[i - l], r - i), 0)
  while i + k < n and s[i + k] == s[k]:
    k += 1
  z[i] = k
  if i + z[i] > r:
    l = i
    r = i + z[i]
```

   Объясните, в чём ошибка, и докажите, что такой код работает за время $\Omega(n^2)$.

2. **(0.5)** `[3]` **Сравнение строк на больше/меньше**
   Даны строки s и t длины n. Поступают запросы: (l_1, r_1, l_2, r_2). Нужно сравнить строки $s[l_1 \dots r_1)$ и $t[l_2 \dots r_2)$. Однако выдать нужно не просто «равны» или «не равны», а «больше», «меньше» или «равны». $\mathcal{O}(\log n)$ на запрос, $\mathcal{O}(n)$ на предподсчёт.

   Подсказка: воспользуйтесь идеей решения задачи «Амнезия».

3. **(0.5)** `[3]` **Суффиксный массив за $\mathcal{O}(n \log^2 n)$**
   Дана строка длины n. Отсортируйте все её суффиксы за $\mathcal{O}(n \log^2 n)$.

4. **(0.5) k-й суффикс**
   Найти k-й в лексикографическом порядке суффикс в строке. $\mathcal{O}(n \log n)$.

5. **(1) Поиск с k ошибками**
   Научиться искать образец s в строке t, если допустимо различие в k символов между образцом и найденной подстрокой. $\mathcal{O}(|t| k \log |s|)$.

6. **(1)** `[1, 2]` **Наибольшая дважды подстрока**
   Найти наибольшую по длине строку, которая дважды без перекрытий встречается в заданной строке. $\mathcal{O}(n \log n)$.

7. **(1.5) Нажал кабан на баклажан**

   (a) **(0.5)** Дана строка s длины n. Поступают запросы: (l, r). Нужно ответить, является ли $s[l \dots r)$ палиндромом. $\mathcal{O}(1)$ на запрос, $\mathcal{O}(n)$ на предподсчёт.

   (b) **(1)** Найдите количество подпалиндромов (то есть подстрок, являющихся палиндромами) строки. $\mathcal{O}(n \log n)$. Подсказка: для каждой позиции i в строке посчитайте количество палиндромов с центром в i-м символе.

8. `(*)` **(+2) Антихеш-тест для $M = 2^{64}$**
   Долгое время в спортивном программировании все использовали в качестве модуля полиномиального хеша $M = 2^{64}$. Это очень удобно: можно проводить все вычисления в беззнаковом 64-битном типе (`uint64_t`), и все вычисления сами будут производиться по модулю M — и код проще и короче, и работает быстрее.

   Но летом 2012 года спортивные программисты заметили, что строка Туэ — Морса (известная аж с 1851 года) «ломает» такое хеширование.

   Определим строку Туэ — Морса. Сначала введём строки S_i:

   - $S_0 = a$
   - $S_1 = ab$
   - $S_2 = abba$
   - $S_n = S_{n-1}(\neg S_{n-1})$, где $\neg$ обозначает замену всех букв a на b и наоборот.

Заметим, что каждая строка является префиксом всех последующих. Обозначим как S_∞ бесконечную строку из букв a и b, содержащую каждую строку S_i как префикс. Эта S_∞ и есть строка Туэ — Морса (иногда вместо a и b используют 0 и 1).

(a) **(+0.5)** Найдите значение $S_\infty[k]$ (k-й символ строки S_∞) за $\mathcal{O}(\log k)$.

(b) **(+0.25)** Докажите, что $h_{2^{64},x}(S_n) = h_{2^{64},x}(S_{n+2})$ для любого $n \geq 6$ и для любого *чётного* основания полиномиального хеширования x.

Тут строка Туэ — Морса не по делу, такую пару строк для чётного x придумать очень просто.

(c) **(+0.75)** Докажите, что $h_{2^{64},x}(S_n) = h_{2^{64},x}(\neg S_n)$ для любого $n \geq 70$ и для любого *нечётного* основания полиномиального хеширования x.

Это уже интересно, но всё-таки такие строчки S_n получаются слишком длинными.

(d) **(+0.5)** Докажите, что $h_{2^{64},x}(S_n) = h_{2^{64},x}(\neg S_n)$ для любого $n \geq 11$ и для любого *нечётного* основания полиномиального хеширования x.

Длина строчки S_{11} — всего лишь 2048 символов.

12 Ахо — Корасик и теория чисел

12.1 Практика

1. `[3]` **Общая подстрока**
   Найти наибольшую общую подстроку двух строк. $\mathcal{O}(n \log n)$.

---

2. **Словари offline**
   Даны словарь (конечное множество слов) и текст.
   a) Для каждого слова из словаря определить, входит ли оно в текст.
   b) Для каждого слова из словаря найти число вхождений в текст.

3. **Словари online**
   Даны словарь (конечное множество слов) и текст. Обновлять ответ `online` при добавлении символа в конец текста.
   a) Пересчитать суммарное число вхождений слов из словаря в текст за $\mathcal{O}(1)$.
   b) Пересчитать множество всех вхождений слов из словаря в текст за $\mathcal{O}(1+|\Delta A|)$, где ΔA – приращение ответа после добавления очередного символа. Вхождение слова — это номер слова и позиция в тексте, с которого его можно считать.

4. **Избегаемость шаблонов («Вирусы»)**
   Дан словарь слов суммарной длины L над алфавитом Σ. За время $\mathcal{O}(L \cdot |\Sigma|)$ определите, существует ли бесконечная строка, не содержащая ни одно словарное слово как подстроку.

5. **XOR** $\to \max$
   Дан массив a длины n. Найдите пару a_i, a_j: $a_i \,\hat{}\, a_j = \max$. Время $\mathcal{O}(n \log M)$, где $M = \max(a_1, \ldots, a_n)$.

6. **XOR** $\geq k$
   Дан массив a длины n и число k. Длина чисел в массиве $\mathcal{O}(1)$. За время $\mathcal{O}(n)$ посчитайте количество
   a) пар индексов таких, что побитовый `XOR` элементов по этим индексам $\geq k$.
   b) отрезков последовательности, побитовый `XOR` всех чисел из которых $\geq k$.

---

7. **Решето за линию**
   Перед вами решето Эратосфена, работающее за линейное время.

```
vector<int> primes, d(n + 1, -1);
for (int y = 2; y <= n; ++y)
  if (d[y] == -1)
    d[y] = primes.size(), primes.push_back(y);
  for (int i = 0; i <= d[y] && y * primes[i] <= n; ++i)
    d[y * primes[i]] = i; // x = y * primes[i], i = d[x]
```

   Докажите корректность и оценку времени работы.

12.2 Домашнее задание

1. **(0.5)** Докажите, что алгоритм Евклида работает за $\mathcal{O}(\log(\min(a, b) + 1) + 1)$ арифметических операций.

2. **(1.5) Длинный `gcd`**
   На лекции мы оценили количество арифметических операций, выполняемых алгоритмом Евклида, считая, что они выполняются за $\mathcal{O}(1)$. Оценим время работы более точно.

   Даны два числа: $a > 0$ длиной n и $b > 0$ длиной m. Мы исключаем случаи $a = 0$ и $b = 0$, поскольку они тривиальные, но из-за них $\mathcal{O}$-оценки становятся более громоздкими.

   Для определённости зафиксируем, что мы храним их в десятичной системе счисления, и длина числа — это количество десятичных цифр в нём. Иными словами, $n = \lceil \log_{10}(a+1) \rceil = \Theta(\log a)$, $m = \lceil \log_{10}(b + 1) \rceil = \Theta(\log b)$.

   Длинные числа можно складывать и вычитать за время $\mathcal{O}(\max(n, m))$, умножать и делить за время $\mathcal{O}(nm)$ — достаточно выполнять эти операции «в столбик». (Умножать и делить можно и быстрее, но нам пока не надо.)

   (a) **(0.25)** Покажите, что рассмотренный на лекции алгоритм Евклида работает за $\mathcal{O}(nm \min(n, m))$ (то есть его сложность, на самом деле, кубическая относительно длин чисел!).

   (b) **(1.25)** Докажите, что:

   - если a и b чётные, то $\gcd(a, b) = 2 \gcd(a/2, b/2)$;
   - если a нечётное и b чётное, то $\gcd(a, b) = \gcd(a, b/2)$.

   Используя эти факты, придумайте алгоритм вычисления `gcd` за $\mathcal{O}(\max(n, m)^2)$.

3. **(1) Оптимизируем решето**
   Добавим оптимизации в решето Эратосфена, которое было рассмотрено на лекции.

```
for (int i = 2; i * i <= n; i++)
  if (is_prime[i])
    for (int j = i * i; j <= n; j += i)
      is_prime[j] = 0;
```

   Докажите, что этот код всё ещё корректен. Отметим, что время работы останется всё также $\mathcal{O}(n \log \log n)$.

4. **(1.5 + 0.5) Количество строк**
   Дан словарь и число n. Посчитайте количество строк длины n над алфавитом $\{a, b\}$, которые не содержат ни одного словарного слова, как подстроку. Так как ответ может быть очень большим, выведите количество по модулю $M = 10^9 + 9$.

   (a) **(1.5)** Время и память $\mathcal{O}(nL)$, где L – суммарная длина слов в словаре.

   (b) **(+0.5)** `(*)` Время $\mathcal{O}(nL)$, память $\mathcal{O}(L)$.

13 Высшая арифметика

В дверь постучали 64 раза.
— Восемь осьминогов, — усмехнулся уже подготовленный Штирлиц.
— Не догадался, — подумали сороконожка и три осьминога.

13.1 Практика

1. **Чудеса в решете**
   Дано $n \le 10^6$. Для каждого x от 1 до n узнайте всё про него.
   a) Простое ли?
   b) Самый маленький простой делитель.
   c) Количество делителей.
   d) `(s)` Сумму делителей.
   e) `(s)` $\varphi(x)$.

2. **Расширенный Евклид**
   a) Докажите, что $\max|x_i| \le |b|$ и $\max|y_i| \le |a|$. В частности, если расширенного Евклида запустить для a и b типа `int64`, вычисления поместятся в тот же тип.
   b) Найдите класс решений диофантового уравнения $ax + by = c$.
   c) Найдите класс решений уравнения $ax \equiv b \pmod{m}$.
   d) `(s)` Докажите, что в строке $ax_i + by_i = r_i$ выполнено $(x_i, y_i) = 1$.
   e) `(s)` Найдите $x, y\colon ax + by = c, |x| + |y| \to \min$

3. **Инициация**
   В некоторых странах Европы нумерация этажей отличается от привычной нам. Например, в пятиэтажном доме в Германии этажи могут нумероваться так: Erdgeschoss («земляной этаж»), первый этаж, второй этаж, третий этаж, четвёртый этаж.

   Начинающий музыкант Миф катается на лифте n-этажного здания. В лифте оказалось только три кнопки: «подняться на a этажей», «подняться на b этажей» и «вернуться на Erdgeschoss» ($a, b > 0$). Сможет ли он попасть на этаж с номером c ($0 \le c \le n-1$)?

4. **Взлом RSA**
   a) Пусть $n = pq$ (p и q простые), известно $\varphi(n)$, разложите n на множители. Таким образом, задача нахождения $\varphi(n)$ не проще задачи разложения на множители (и, очевидно, не сложнее, поскольку, зная разложение, мы легко вычислим $\varphi(n)$).
   b) Пусть у нас есть «волшебный» оракул. Для любого открытого ключа (n, e) оракул может взломать 1% из всех возможных зашифрованных сообщений. Придумайте алгоритм, который взламывает любое сообщение. Матожидание времени работы $\mathcal{O}(\mathrm{poly}(\log n))$.

---

5. **Блочное решето**
   a) Найти все простые числа на $[n^2, n^2 + n]$ за $\mathcal{O}(n \log\log n)$.
   b) Найти все простые на $[1, n]$ за $\mathcal{O}(n \log\log n)$ с $\mathcal{O}(\sqrt{n})$ памяти.

6. **RSA и простота**
   Пусть оказалось, что сообщение, шифруемое RSA, не взаимно просто с n. Сломается ли процедура шифрования/дешифрования? Чем плохо такое сообщение?

7. **Степень вхождения**
   Пусть $m \in \mathbb{N}$, $p \in \mathbb{P}$. Представим m в виде $m = p^k s$, где s не делится на p. Такое k называют *степенью вхождения* p в число m и иногда обозначают $\mathrm{ord}_p(m)$.

   Дано число n и простое p. Найдите $\mathrm{ord}_p(n!)$ за время $\mathcal{O}(\log n)$.

8. **Цешки по модулю**
Для заданных n, k и простого p посчитайте $\binom{n}{k} \bmod p$. Учтите, что p может быть меньше, чем n. Время $\mathcal{O}(n)$. Можно считать, что все операции в $\mathbb{Z}_p$ выполняются за $\mathcal{O}(1)$.

9. **Магия**
Поймите, что делает код:

```
f[1] = 1;
for (int i = 2; i < p; i++)
  f[i] = (p - f[p % i]) * (p / i) % p;
```

13.2 Домашнее задание

x называется нетривиальным делителем n, если n делится на x и $1 < x < n$. Нетривиальное разложение числа n на множители — это представление n в виде $n = xy$, где $x, y > 1$.

1. **(1) Магия**
   Докажите, что следующий код для простого числа p за время $\mathcal{O}(p)$ находит обратные элементы ко всем числам от 1 до $p - 1$.

```
f[1] = 1;
for (int i = 2; i < p; i++)
  f[i] = (p - f[p % i]) * (p / i) % p;
```

2. **(1) Факторизация через gcd**
   Даны числа n, a и b. Оказалось, что $a^2 \equiv b^2 \bmod n$ и $a \not\equiv \pm b \bmod n$. Найдите нетривиальное разложение n на множители (не обязательно простые). Время $\mathcal{O}(\text{poly}(\log n))$.

3. **(1) Эратосфен и гладкость**
   Обозначим i-е по возрастанию простое число, как p_i. Назовём число b-гладким, если все его простые делители не превосходят p_b.

   Дано n. Для каждого $b \le n$ найдите количество b-гладких чисел в множестве $\{1, 2, \ldots, n\}$. Время $\mathcal{O}(n \log \log n)$.

4. **(1.25) RSA и факторизация**
   Известны открытый ключ $(n, 3)$ и закрытый ключ (n, d) системы `RSA`. Известно, что n — произведение двух различных простых. Разложите n на множители. Время $\mathcal{O}(\text{poly}(\log n))$.

5. **(+1) (*) Рекурсивная факторизация**
   Пусть дана процедура $fact(n)$, работающая за время $\mathcal{O}(n^\alpha)$ ($\alpha \le 0.5$), которая возвращает:

   - любой нетривиальный (не обязательно простой) делитель n, если n составное;
   - -1, если n простое.

   Покажите, как разложить n на простые множители за $\mathcal{O}(n^\alpha)$.

14 Оценки для хешей, BST и AVL

14.1 Практика

Оценки для хешей

1. **Модуль в хешах — 1**
   Задача: $q \le 10^6$ раз проверить на равенство две подстроки строки s длины $|s| \le 10^6$. Хватит ли 32-битного целочисленного типа для хранения хеша?

2. **Модуль в хешах — 2**
   Мы считаем число различных подстрок хешами за $\mathcal{O}(n^2)$, $n \le 1000$. Хватит ли 32-битного целочисленного типа для хранения хеша?

Бинарные деревья поиска

> Идут лесом, поют куролесом, несут деревянный пирог с мясом.
>
> Отгадка

3. **Симметричный обход BST** (in-order traversal, LNR traversal)
   Выведите все добавленные в BST элементы в отсортированном порядке. Время $\mathcal{O}(n)$.

4. **Операции с BST**
   Реализуйте за время $\mathcal{O}(h)$ в BST:

   (a) `next(v)`: по данному узлу v найти узел u такой, что $u.x > v.x$ и $u.x$ минимально.

   (b) `lower_bound(x)`: по данному значению x найти узел v такой, что $v.x \ge x$ и $v.x$ минимально.

   (c) `kth_element(k)` — k-я порядковая статистика: по данному k найти узел v такой, что в симметричном обходе он будет на k-м месте.

   (d) `del(x)`: по данному x удалить из дерева узел v такой, что $v.x = x$.
   Подсказка: если у v не больше одного ребёнка, всё просто; если два, рассмотрите `next(v)`.

5. **Минимум за $\mathcal{O}(1)$**
   Как находить минимум в BST за $\mathcal{O}(1)$?

6. **Высота AVL-дерева**
   Докажите, что высота любого AVL-дерева есть $\mathcal{O}(\log n)$.

   Подсказка: обозначим за s_h минимальный размер дерева с высотой h. Напишите рекуррентное соотношение на s_h, оцените скорость роста s_h относительно h.

7. **Удаление из AVL-дерева**
   Реализуйте `del(x)` в AVL-дереве за время $\mathcal{O}(\log n)$.

8. **Объединение AVL-деревьев**
   Пусть вам даны два AVL-дерева T_1 и T_2. Придумайте, как построить AVL-дерево T, являющееся объединением деревьев T_1 и T_2 за время $\mathcal{O}(height(T_1) \times size(T_2))$, если $size(T_1) \ge size(T_2)$.

---

9. **Операции над AVL**

   a) Пусть `root.l.h = root.r.h + 3`. Как перебалансировать дерево за $\mathcal{O}(1)$?
   b) Пусть `root.l.h = root.r.h + k`. Как перебалансировать дерево за $\mathcal{O}(k)$?
   c) Merge за $\mathcal{O}(\log n)$ (два способа).
   d) Split за $\mathcal{O}(\log^2 n)$. `(*)` за $\mathcal{O}(\log n)$.
   e) Уменьшите количество дополнительной информации до двух бит на вершину.

10. **Быстрый отсортированный массив** Нужно быстро выполнять запросы:
    - `get_ith_element(i)` (i-й по величине элемент);
    - `get_position(x)` (номер элемента в отсортированном массиве);
    - `add(i, x, y)` (вставить x на i-ю позицию; гарантируется, что $S[i-1] < x < S[i]$);
    - `del(i)` (удалить элемент на i-й позиции).
11. **Быстрый массив**
    Нужно быстро выполнять запросы: `get(i)`, `set(i, y)`, `insert(i, y)` (вставить x после i-го элемента), `del(i)`, `rotate(k)` (циклический сдвиг на k элементов).
12. **Модификации отрезка**
    a) Запросы: `insert(i, x)`, `del(i)`, `get_sum(l, r)`, `set(l, r, value)`.
    b) Добавим запросы `reverse(l, r)` и `rotate(k)`.
13. **k-min**
    Как вывести k минимальных элементов в AVL-дереве за $\mathcal{O}(k)$?
    А для произвольного BST?

14.2 Домашнее задание

1. **(0.75) Построить AVL**
   Дан отсортированный массив. Построить AVL-дерево за $\mathcal{O}(n)$.

2. **(0.5) Объединение AVL-деревьев**
   Пусть вам даны два AVL-дерева T_1 и T_2. Придумайте, как построить AVL-дерево T, являющееся объединением деревьев T_1 и T_2 за время $\mathcal{O}(size(T_1) + size(T_2))$.

3. **(0.75) Номер по элементу**
   Реализуйте за время $\mathcal{O}(h)$ в BST операцию `get_position(x)`: по данному значению x узнать его позицию в симметричном обходе. Иными словами, эта операция обратная к поиску порядковой статистики. Ключи в дереве можно считать различными.

4. **(0.75) Поворот не туда**
   Покажите, что любые два корректных дерева поиска, построенные на одном и том же множестве ключей, можно получить друг из друга последовательностью поворотов. Ключи в множестве можно считать различными.

5. **(+0.75)** `(*)` **Восстановление по прямому обходу**
   Дан прямой обход BST (pre-order, NLR): сначала выписали корень, потом рекурсивно левое поддерево, потом рекурсивно правое поддерево. За $\mathcal{O}(n)$ восстановить BST.

6. **(+1)** `(*)` **Быстрое несбалансированное BST**
   Рассмотрим обычное несбалансированное BST. Если делать добавление в лоб спуском вниз, время работы на один запрос добавления может быть $\Omega(n)$. В процессе мы можем поддерживать для каждой вершины ссылки на детей и отца, а также глубину вершины (расстояние до корня). Научитесь делать ту же самую работу — поддерживать для каждой вершины детей, отца, глубину — быстрее, а именно за $\mathcal{O}(\log n)$ на один запрос добавления.