Практика по алгоритмам

Алексей Лапенок, Александр Мишунин *

Весна, 2021

Содержание

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

1 Задачи RMQ & LCA

1.1 Практика

1. Вам дано дерево из одной вершины. Придумайте, как online отвечать на следующие запросы:

   - Подвесить новую вершину к дереву.
   - `LCA` двух вершин.

   (a) $\mathcal{O}(1)$ на подвешивание, $\mathcal{O}(h)$ на `LCA` (h — текущая высота).
   (b) $\langle \mathcal{O}(\log n), \mathcal{O}(\log n) \rangle$ (n — число добавлений; для простоты считайте, что n известно заранее).
   (c) $\langle \mathcal{O}(1), \mathcal{O}(\log n) \rangle$ (используйте скошенный список).

2. Придумайте online static `RMQ` за

   (a) $\mathcal{O}(n)$ на предподсчет, $\mathcal{O}(\log n)$ на запрос.
   (b) $\mathcal{O}(n \log n)$ на предподсчет, $\mathcal{O}(1)$ на запрос.
   (c) $\mathcal{O}(n \log \log n)$ на предподсчет, $\mathcal{O}(1)$ на запрос.
   (d) $\langle \mathcal{O}(n \log^* n), \mathcal{O}(\log^* n) \rangle$.
   (e) ± 1 `RMQ` за $\langle \mathcal{O}(n), \mathcal{O}(1) \rangle$.

3. Дано подвешенное дерево из n вершин с весами на ребрах. Online-запросы:

   (a) сумма весов на пути из a в b. $\langle \mathcal{O}(n \log n), \mathcal{O}(1) \rangle$.
   (b) минимальный вес на пути из a в b. $\langle \mathcal{O}(n \log n), \mathcal{O}(\log n) \rangle$.

4. Вам дано подвешенное дерево из n вершин. Нужно online отвечать на запросы:

   - Найти `LCA` двух вершин,
   - Переподвесить дерево за новую вершину.

   Время $\langle \mathcal{O}(n \log n), \mathcal{O}(1) \rangle$.

5. Сведите задачу `RMQ` к `LCA` с помощью декартова дерева за $\mathcal{O}(n)$.

6. В последовательности длины n из различных целых чисел с помощью дерева отрезков найдите наибольшую возрастающую подпоследовательность за $\mathcal{O}(n \log n)$

7. Дан массив из нулей и единиц. Запросы: “поменять элемент”, “найти ближайшую слева/справа единицу к позиции i”. $\langle \mathcal{O}(n), \mathcal{O}(\log n) \rangle$, online.

8. Дана таблица $n \times n$ с целыми числами. Поступают запросы:

   - Изменить значение в ячейке (i, j),
   - Посчитать сумму чисел в прямоугольнике $((l, t), (r, b))$.

   $\langle \mathcal{O}(n^2 \log n), \mathcal{O}(\log^2 n) \rangle$, online.

9. k-инверсией в перестановке p называется набор индексов $i_1 < i_2 < \ldots < i_k$, такой, что $p[i_1] > p[i_2] > \ldots > p[i_k]$. Найти число k-инверсий за $\mathcal{O}(nk \log n)$.

10. Вывести все числа $\geq X$ на отрезке $[L, R]$ массива. Static online

    - $\langle \mathcal{O}(n \log n), \mathcal{O}(\log^2 n + k) \rangle$ (k — размер ответа).
    - $\langle \mathcal{O}(n \log n), \mathcal{O}(\log n + k) \rangle$

11. Дерево с весами в вершинах. Нужно научиться online отвечать на запрос $count(a, b, x)$ — количество вершин на пути из a в b, у которых вес $\geq x$, если:

    (a) Веса не меняются. $\mathcal{O}(\log n)$.

    (b) Теперь веса меняются. Запрос за $\mathcal{O}(\log^2 n)$, изменение веса за $\mathcal{O}(\log^2 n)$.

1.2 Домашнее задание

1. Дано дерево из одной вершины. Требуется уметь отвечать online за $\mathcal{O}(\log n)$ на запросы:

   - подвесить новую вершину u к вершине дерева v,
   - вернуть диаметр дерева.

   Диаметр дерева — длина самого длинного простого пути в дереве.

2. Дан ориентированный граф, в котором исходящая степень каждой вершины равна единице. Запросы online: из вершины v сделать k шагов вперед и вернуть куда пришли.

   (a) Предподсчет: $\mathcal{O}(n \log k_{\max})$, время на запрос: $\mathcal{O}(\log k)$.

   (b) Предподсчет: $\mathcal{O}(n \log n)$, время на запрос: $\mathcal{O}(\log \min(k, n))$.

3. Дана скобочная последовательность из круглых скобок длины n. Запросы:

   - является ли отрезок $[L, R]$ правильной скобочной последовательностью,
   - изменить i-ю скобку.

   $\mathcal{O}(n)$ на предподсчет, $\mathcal{O}(\log n)$ на запрос, online.

4. Попробуем модифицировать идею `SparseTable` так, чтобы она работала для произвольных ассоциативных функций: предложите способ выделить $\mathcal{O}(n \log n)$ отрезков в массиве размера n так, что любой отрезок $[L, R]$ можно было представить в виде объединения $\mathcal{O}(1)$ непересекающихся выделенных отрезков. Заметим, что дерево отрезков выделяет $\mathcal{O}(n)$ отрезков, и любой отрезок представляется как объединение $\mathcal{O}(\log n)$ из них.

Дополнительные задачи

5. Придумайте, как в задаче 4 домашнего задания обойтись $\mathcal{O}(n \log \log n)$ отрезков.

2 AVL-деревья

2.1 Практика

1. 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:
   (a) $\mathcal{O}(height(T_1) \times size(T_2))$, если $size(T_1) \geq size(T_2)$.
   (b) $\mathcal{O}(size(T_1) + size(T_2))$
2. Дано двоичное дерево, в котором AVL-свойство выполнено везде, кроме корня.
   (a) Пусть `root.l.h = root.r.h + 3`. Как восстановить AVL-свойство всюду за $\mathcal{O}(1)$?
   (b) Пусть `root.l.h = root.r.h + k`. Как восстановить AVL-свойство всюду за $\mathcal{O}(k)$?
3. Пусть вам даны два AVL-дерева T_1 и T_2 и элемент k, при этом все ключи из T_1 меньше k, а из T_2 больше k Придумайте, как построить AVL-дерево T, являющееся объединением деревьев T_1 и T_2 и содержит элемент k за время $\mathcal{O}(size(T_1) + size(T_2))$
4. Придумайте `split` для AVL-деревьев за $\mathcal{O}(\log^2 n)$.
5. Придумайте `merge` для AVL-деревьев за $\mathcal{O}(\log n)$.
6. Докажите, что придуманный `split` на самом деле работает за $\mathcal{O}(\log n)$.
7. Пусть дано AVL-дерево T, ключ k и число n. Придумайте алгоритм, который выведет n элементов, следующих за k, за время $\mathcal{O}(h(T) + n)$.
8. Придумайте структуру данных, позволяющую online за $\mathcal{O}(\log n)$ отвечать на следующие запросы:
   - добавить элемент x
   - удалить элемент x
   - найти i-ый по порядку элемент
   - найти порядок элемента x
9. Запросы online за $\mathcal{O}(\log n)$:
   - добавить пару $\langle x, y \rangle$
   - удалить пару $\langle x, y \rangle$
   - посчитать сумму y по всем парам таким, что $l \leq x \leq r$
10. Запросы online за $\mathcal{O}(\log n)$:
    - добавить пару $\langle x, y \rangle$
    - посчитать сумму y по всем парам таким, что $l \leq x \leq r$
    - посчитать сумму x по всем парам таким, что $l \leq y \leq r$
11. Запросы online за $\mathcal{O}(\log n)$:
    - `add(i, x)` — вставить x на позицию i, все элементы после него сдвигаются на 1 вправо
    - `del(i)` — удалить элемент на позиции i, все элементы после него сдвигаются на 1 влево
    - `sum(l, r)` — сумма всех x, для которых $l \leq i \leq r$
    - `add(l, r, value)` — добавить *value* ко всем x, для которых $l \leq i \leq r$
12. Уменьшите количество дополнительной информации в AVL-дереве до двух бит на вершину.
13. Покажите, что любые два корректные дерева поиска, построенные на одном и том же множестве ключей, можно получить друг из друга последовательностью поворотов.

2.2 Домашнее задание

1. Покажите, что:

   (a) Добавление в AVL-дерево требует $\mathcal{O}(1)$ вращений.

   (b) Удаление из AVL-дерева требует $\Omega(\log n)$ вращений в худшем случае.

2. Вершина дерева является *единственным ребёнком*, если у ее родителя только один ребёнок (корень единственным ребёнком не является). Определим для дерева степень одиночества LR таким образом:

$$LR(T) \;=\; \frac{\text{Количество единственных детей в } T}{\text{Количество вершин в } T}$$

   (a) Покажите, что в любом непустом AVL-дереве T $LR(T) \leq \frac{1}{2}$.

   (b) Правда ли, что если для бинарного дерева T верно $LR(T) \leq \frac{1}{2}$, то $h(T) = \mathcal{O}(\log size(T))$?

   (c) Правда ли, что существуют $0 < \alpha < \beta < \frac{1}{2}$ такие, что:

   - существует хотя бы одно бесконечное семейство бинарных деревьев $\{T_i\}_{i\in\mathbb{N}}$, где все $\alpha \leq LR(T_i) \leq \beta$.
   - В максимальном по включению таком семействе $h(T_i) = \mathcal{O}(\log size(T_i))$.

3. Пусть в обычное несбалансированное бинарное дерево поиска добавляются различные элементы в порядке $x_1, x_2, \ldots, x_n$. Придумайте алгоритм, поддерживающий для каждой вершины её глубину и ссылки на родителя и детей и обновляющий эту информацию при добавлении элемента в дерево за $\mathcal{O}(\log n)$.

   Пояснение: если делать добавление наивным спуском вниз, на запрос может уйти $\Omega(n)$ времени; требуется обрабатывать запросы быстрее.

4. Придумайте, как реализовать структуру данных, поддерживающую следующие операции на последовательности из n чисел:

   - Обмен местами последовательных пар соседних чисел на заданном отрезке четной длины (пример: $[1, 2, 3, 4, 5, 6], (2, 5) \to [1, 3, 2, 5, 4, 6]$),
   - Вывод числа на заданной позиции.

   Время работы — $\mathcal{O}(\log n)$ на запрос.

Дополнительные задачи

5. Напишите на каком-нибудь строго типизированном функциональном языке с зависимыми типами реализацию исправления произвольного дисбаланса и добавления в AVL-дерево так, чтобы типизацией гарантировался:

   (a) AVL-инвариант.

   (b) AVL-инвариант и порядок на ключах.

6. Постройте тест, на котором «недо-AVL-дерево», которое делает только малые вращения, делает $\omega(n \log n)$ операций после n запросов.

   Формально: изначально дерево пусто, нужно n раз вызвать `add(root,` x_i`)` для некоторой последовательности x_i, что суммарное время работы $\omega(n \log n)$.

   Либо докажите, что такого теста нет.

3 Splay и декартовы деревья и неявные ключи

3.1 Практика

1. Пусть есть произвольное `splay`-дерево, построенное на ключах $1, 2, 3 \dots n$. Покажите, что в результате последовательных вызовов `splay(1)`, `splay(2)`, ... `splay(n)` получится бамбук.

2. Придумайте структуру данных, реализующую список длины n с online-операциями за $\mathcal{O}(\log n)$:

   - `read(i) / write(i, x)` — прочитать/записать элемент в позиции i.
   - `move(l, r, k, right / left)` — переместить подотрезок на k позиций влево или вправо. Гарантируется, что подотрезок не вылезет за границы массива.
   - `rev(l, r)` — развернуть отрезок.

3. Задача "вставка ключа": изначально все ячейки пусты, требуется online за $\mathcal{O}(\log n)$ обрабатывать запросы:

   - `Insert(i, x)` — вставить x в ячейку i. Если i-я ячейка занята, все ячейки, начиная с i-й, сдвигаются вправо. Ячейка i не обязательно является занятой или соседней с занятой
   - Узнать элемент на позиции i

   Пример: $(5, 1); (5, 2); (5, 3); (1, 7); (1, 8); (2, 9) \to 8, 9, 7, 0, 3, 2, 1$.

4. Запросы online за $\mathcal{O}(\log n)$:

   - `add(i, x)` — вставить x на позицию i, все элементы после него сдвигаются на 1 вправо
   - `del(i)` — удалить элемент на позиции i, все элементы после него сдвигаются на 1 влево
   - `sum(l, r)` — сумма всех x, для которых $l \le i \le r$
   - `add(l, r, value)` — добавить *value* ко всем x, для которых $l \le i \le r$

5. Пусть в начале есть много `splay`-деревьев суммарного размера n, над которыми затем алгоритм делает m операций `split` и `merge`. Правда ли, что суммарное время работы операций с деревьями можно оценить как $\mathcal{O}((m + n) \log n)$, если:

   (a) деревья персистентные.

   (b) деревья эфемерные.

6. Мальчик Петя взял пары (x_i, y_i) и построил на них декартово дерево. Приведите пример, на котором полученное дерево не является статически оптимальным для ключей x_i и их частот y_i.

7. Нарисуйте все Декартовы деревья, которые могут получится в результате операции `merge(бамбук идущий влево-вниз, вершина)` и `merge(бамбук идущий вправо-вниз, вершина)`.

8. Пусть приоритеты случайны. Покажите, что наличие одинаковых ключей не портит оценку на матожидание высоты Декартова дерева для последовательности стандартных операций `insert`.

9. (a) Попробуем улучшить константу у стандартного `insert` следующим образом: сначала будем спускаться по дереву, пока не встретим узел с меньшим, чем у нового, приоритетом, и уже этому поддереву сделаем `split` и подвесим две части к новому узлу. Покажите, что результате всегда получится корректное Декартово дерево.

   (b) Придумайте аналогичный `delete` через спуск и один `merge`.

   (c) Пусть теперь приоритеты случайны, а ключи могут быть одинаковыми. Заметим, что при спуске у нас есть выбор, в какое поддерево идти при равенстве ключей. Влияет ли этот выбор на оценку матожидания высоты дерева?

10. Попробуем построить сливаемую кучу по минимуму на основе дерева следующим образом: пусть каждая вершина дерева хранит один ключ и односвязный список ссылок на дочерние поддеревья. На ключах выполняется свойство кучи. Операции будем выполнять следующим образом:

    - *create-heap* — создать пустое дерево
    - *find-min* — вернуть ключ, хранящийся в корне дерева
    - *merge* — добавить ссылку на дерево с большим (по ключу) корнем в начало списка детей корня дерева с меньшим (если оба дерева не пустые)
    - *insert* — *merge* с деревом из одного элемента
    - *delete-min* — удалить корень и слить его детей таким образом: идем по списку слева направо, сливая детей попарно (1-2, 3-4, 5-6 и т. д.). Потом идем справа налево и сливаем все деревья в одно. Например, из списка детей $[c_1, c_2, c_3, c_4, c_5]$ получится дерево:

    $$merge(merge(c_1, c_2), merge(merge(c_3, c_4), c_5))$$

    Докажите, что любая корректная последовательность из n таких операций работает за $\mathcal{O}(n \log n)$ (в начальный момент куч нет). Подсказка: это должно быть как-то связано со `splay`-деревьями.

11. Придумайте модификацию `splay`-дерева, которая умеет делать `splay(key)` за один проход используя $\mathcal{O}(1)$ дополнительной памяти и при этом не хранит ссылки на предков.

12. Покажите, что любые два корректные дерева поиска, построенные на одном и том же множестве ключей, можно получить друг из друга последовательностью поворотов.

Дополнительные задачи

13. Пусть K — множество ключей, а $is\_root(x, \tau)$ — предикат, определяющий является ли ключ x корнем дерева τ. Назовем `RBST[`K`]` такое распределение $\mathcal{R}$ на двоичных деревьях на K, что:

    - Распределение на пустом дереве — `RBST[`$\varnothing$`]`.
    - $\forall k \in K\colon \underset{\tau \in \mathcal{R}}{\mathbf{Pr}}\left(is\_root(k, \tau)\right) = \frac{1}{|K|}$
    - Для любого $k \in K$ условное распределение $\underset{\tau \in \mathcal{R}}{\mathbf{Pr}}\left(\cdots | is\_root(k, \tau)\right)$ индуцирует на поддеревьях корня независимые распределения, являющиеся `RBST`.

    (a) Докажите, что `RBST[`K`]` единственно.

    (b) Докажите, что результат вставки равномерно случайной перестановки K в обычное дерево — `RBST[`K`]`.

    (c) Докажите, что Декартово дерево на K с равномерно случайными приоритетами из $(0, 1)$ — `RBST[`K`]`.

    (d) Докажите, что результат `split` по любой константе на `RBST` является парой независимых `RBST`.

    (e) Докажите, что `merge` двух независимых `RBST`, который выбирает корень случайно из корней двух аргументов с вероятностью, пропорциональной размерам аргументов, возвращает `RBST`.

    (f) Покажите, что `merge` двух зависимых `RBST` может не быть `RBST`.

    (g) Докажите верхнюю оценку $3 \log_2 |K| + \mathcal{O}(1)$ на матожидание глубины `RBST`.

14. Пусть `splay`-дерево поддерживает множество S. В серии из m запросов каждый элемент $x_i \in S$ был запрошен $p_i m$ раз; $0 < p_i \leq 1$ и $\sum_i p_i = 1$. Докажите, что `splay`-дерево обработало все запросы за время $\mathcal{O}(m \cdot \left[1 + \sum_i p_i \cdot \log \frac{1}{p_i}\right])$.

3.2 Домашнее задание

1. Придумайте структуру данных, поддерживающую упорядоченный по значению список S целых чисел, которая умеет отвечать на запросы:

   - `insert(x)` — вставить x в S, если его там не было.
   - `delete(x)` — удалить x из S, если он там был.
   - `S[k]` — вернуть k-тый по порядку элемент из S.
   - `max(l, r)` — найти $\max_{l \le j < k \le r} |S[j] - S[k]|$. Гарантируется $r - l \ge 1$.
   - `min(l, r)` — найти $\min_{l \le j < k \le r} |S[j] - S[k]|$. Гарантируется $r - l \ge 1$.

   Каждый запрос должен обрабатываться за $\mathcal{O}(\log |S|)$.

2. Докажите, что существует такое $N \in \mathbb{N}$, что для любого множества различных ключей S, $|S| \ge N$ любое бинарное дерево поиска на нём можно получить из любого другого дерева на S последовательностью операций `splay`. (Подсказка: покажите, что, если x — лист, то `splay` любой вершины, кроме x, оставляет x листом)

3. Даны число K и изначально пустая последовательность. Вам поступает n запросов, каждый одного из двух типов:

   - `append(x)` — дописать элемент x в конец последовательности
   - `rev()` — развернуть K последних элементов последовательности (если в данный момент их всего меньше K, то развернуть всю последовательность).

   Вам нужно один раз после всех запросов вывести получившуюся последовательность.

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

4. Пусть приоритеты случайны, а ключи все разные. Обозначим за x_k вершину Декартова дерева, содержащую k-тый по порядку ключ. Всего в дереве n вершин.

   (a) Пусть $1 \le i \le j \le k \le n$. Найдите вероятность того, что x_j — общий предок x_i и x_k.
   (b) Пусть $1 \le i \le k \le n$. Найдите матожидание длины пути между x_i и x_k.

   Ответ достаточно выразить через комбинацию гармонических чисел $H_n = \sum\limits_{d=1}^{n} \frac{1}{d}$.

Дополнительные задачи

5. Придумайте аналог Декартова дерева со случайными приоритетами для хранения точек на плоскости с операциями `splitX`, `mergeX`, `splitY`, `mergeY`. Все операции должны работать за $o(n)$.

6. `C-c-c-combo!`
   Напишите псевдокод для структуры, умеющей отвечать на запросы:

   - `insert(i, x)`, $0 \le i \le L$ (L — текущая длина) — вставить x в позицию i. Все элементы в i и правее сдвигаются на 1 вправо.
   - `delete(i)` — удалить элемент из позиции i. Все правее сдвигается на 1 влево.
   - `add(l, r, value)` — добавить *value* ко всем x, для которых $l \le i \le r$
   - `set(l, r, value)` — установить в *value* все x, для которых $l \le i \le r$
   - `sum(l, r)` — сумма всех x, для которых $l \le i \le r$
   - `reverse(l, r)` — изменить порядок всех x, для которых $l \le i \le r$, на обратный

   В начальный момент структура пустая. Время работы на запрос $\mathcal{O}(\log L)$, online.

4 Хеширование строк

4.1 Практика

Все алгоритмы в этой серии задач вероятностные, то есть могут ошибаться с вероятностью, не превышающей $\varepsilon = \Theta(1)$.

1. Сколько существует различных функций из $\{0,1\}^n$ в $\{0,1\}^m$?
2. Придумайте для строки длины n предподсчет за $\mathcal{O}(n)$, позволяющий:
   (a) Вычислить за $\mathcal{O}(1)$ полиномиальный хеш любой подстроки.
   (b) Лексикографически сравнить любые две подстроки за $\mathcal{O}(\log n)$.
3. Пусть есть семейство полиномиальных хеш-функций $\mathcal{H}$ по простому модулю m со случайной точкой x. Покажите, что для любой пары строчек s_1, s_2 длины не более n:
$$\mathop{\mathbf{Pr}}_{h \in \mathcal{H}} [\ h(s_1) = h(s_2)\ ] \leq \frac{n}{m}$$
4. Найдите все периоды строки. $\mathcal{O}(n)$.
5. (a) Найти число различных подстрок в строке. $\mathcal{O}(n^2)$.
   (b) Найти подстроку данной строки, встречающуюся максимальное число раз.
6. Найти k-й в лексикографическом порядке суффикс в строке.
   (a) $\mathcal{O}(n \log^2 n)$.
   (b) $\mathcal{O}(n \log n)$.
7. Определим следующую последовательность строк:
   - $S_1 = 0$
   - $S_2 = 01$
   - $S_3 = 0110$
   - $S_n = S_{n-1}(\neg S_{n-1})$

   Заметим, что каждая строка является префиксом всех следующих. Обозначим как S_∞ строку, содержащую каждую S_i как префикс. Докажите, что если результат полиномиального хеширования вычисляется по модулю 2^{64}, то вне зависимости от выбранной (нечетной) точки $hash(S_{12}) = hash(\neg S_{12})$.
8. Научиться искать образец в строке, если допустимо различие в один символ между образцом и найденной подстрокой. $\mathcal{O}(n \log m + m)$.
9. Найти наибольшую по длине строку, которая дважды без перекрытий встречается в заданной строке. $\mathcal{O}(n \log n)$.
10. (a) Найти количество подпалиндромов строки. $\mathcal{O}(n \log n)$.
    (b) Найти максимальный подпалиндром строки. $\mathcal{O}(n)$.

Дополнительные задачи

11. Дано начальное состояние в игре «Жизнь» в виде битовой матрицы $N \times N$. Нас интересует, сколько останеться живых клеточек на поле после большого количества итераций $T \gg N$. Для простоты можно считать N и T степенями двойки.

    (a) Определим функцию e, сопоставляющую тайлу размера $2^n \times 2^n$ кусок размера $2^{n-1} \times 2^{n-1}$ из его середины на 2^{n-2} шагов в будущем. Можно ли написать рекуррентное соотношение для e?

    (b) Придумайте, как с помошью хеширования решить нашу задачу быстрее, чем наивной симуляцией.

4.2 Домашнее задание

Определение. *Семейство хеш-функций $\mathcal{H} = \{h : X \to Y\}$ называется*

- *универсальным, если:*

$$\forall x_1, x_2 \in X, x_1 \neq x_2 : \underset{h \in \mathcal{H}}{\boldsymbol{Pr}} [\, h(x_1) = h(x_2) \,] \leq \frac{1}{|Y|}$$

- *k-независимым, если для любых различных $x_1, x_2, \cdots, x_k \in X$, для любых, возможно, совпадающих $y_1, y_2, \cdots, y_k \in Y$ выполняется:*

$$\underset{h \in \mathcal{H}}{\boldsymbol{Pr}} \left[\bigwedge_{i=1}^{k} h(x_i) = y_i \right] = \frac{1}{|Y|^k}$$

1. (a) Докажите, что любое 2-независимое семейство хэш-функций является универсальным.
   (b) Докажите, что любое $k+1$-независимое семейство хэш-функций является k-независимым.
2. Придумайте строку длины n над алфавитом $\{0, 1\}$, в которой $\Omega(n^2)$ различных подстрок.
3. Даны две строки. Найдите их наибольшую общую подстроку. $\mathcal{O}(n \log n)$.
4. Даны строка s длины n и число k. Найдите в s за время $\mathcal{O}(n)$ самую длинную подстроку, представимую как степень некоторой строки x, для которой $|x| = k$ (степенью строки называют её конкатенацию с собой несколько раз).
5. *(только группа Лапенка)* Число $p < |s|$ является периодом строки s, если для любого i $s[i] = s[i + p]$. Дана строка. Найти все ее периоды. $\mathcal{O}(n)$.
6. *(только группа Мишунина)* Найти подстроку в тексте. При сравнении строк, если несовпадений символов было не более k, строки считаются равными. $\mathcal{O}(nk \log n)$.

Дополнительные задачи

7. Напишите на `Haskell` определение бесконечного списка `Int`-ов, представляющего собой последовательность Туэ-Морса (последовательность из задачи 7 практики) S_∞, такое, что:
   - определение содержит не более 128 символов
   - можно использовать стандартные функции из модулей `Prelude` и `Data.List`
   - вычисление в нормальную форму `genericTake n` от списка занимает время $\mathcal{O}(n)$.
8. Пусть дана w-разрядная `RAM` машина (т.е. машина, которая умеет производить арифметические операции и чтение/запись по адресу с w-битными словами за $\mathcal{O}(1)$). Для простоты можно считать w степенью двойки. Придумайте для такой машины детерминированный предподсчет за $\mathcal{O}(w)$ операций, который позволит по w-битному числу вида 2^k восстанавливать k за $\mathcal{O}(1)$ операций.

5 Универсальные семейства

5.1 Практика

1. Является ли семейство из одной идеальной хеш-функции:
   (a) универсальным?
   (b) 2-независимым?
2. Пусть случайно и равновероятно выбрана функция $f : \mathbb{F}_2^n \to \mathbb{F}_2^m$. Найдите вероятность события
   (a) $f(x) = y$ для фиксированных x и y,
   (b) $f(x_1) = f(x_2)$ для фиксированных x_1 и x_2,
   (c) $f(x_1)$ отличается от $f(x_2)$ ровно в одном бите.
3. (a) Для каких n и m семейство всех функций $\mathbb{F}_2^n \to \mathbb{F}_2^m$ является универсальным?
   (b) ... 2-независимым?
   (c) Может ли существовать конечное универсальное семейство функций $\{X \to \mathbb{F}_2^m\}$, если X бесконечно?
4. В вашем распоряжении есть пара 2-независимых семейств хеш-функций $\mathcal{A} = \{f : A \to \mathbb{F}_2^n\}$ и $\mathcal{B} = \{g : B \to \mathbb{F}_2^n\}$. Постройте (и докажите, что построено правильно) 2-независимое семейство хеш-функций:
   (a) $\{h : A \cup B \to \mathbb{F}_2^n\}$ при условии $A \cap B = \varnothing$,
   (b) $\{h : A \times B \to \mathbb{F}_2^n\}$.
5. Построим хеш-функцию $h : \mathbb{F}_2^n \to \mathbb{F}_2^k$ для $k < n$. Зафиксируем матрицу A ранга k над полем $\mathbb{F}_2$ и вектор $b \in \mathbb{F}_2^k$. Пусть
$$h(x) := A \cdot x + b.$$
   (a) Найдите I максимальной мощности такое, что $|h(I)| = 1$. Какова мощность I?
   (b) Постройте (и докажите, что построено правильно) 2-независимое семейство хеш-функций $\mathcal{H} = \{h_i : \mathbb{F}_2^n \to \mathbb{F}_2^k\}_i$.
6. Пусть p — простое. Определим семейство $\mathcal{H} = \{f : \mathbb{F}_p^n \to \mathbb{F}_p\}$ следующим образом:
$$f(x) = b + \sum_k a_k x_k \mod p$$
   где b и a_k выбираются равновероятно независимо из $[0 \dots p-1]$. Покажите, что получилось 2-независимое семейство.
7. Пусть p — простое. Определим хеш-функцию $h_{a_0,\dots,a_{k-1}}(x) := \sum_{i=0}^{k-1} a_i \cdot x^i \mod p$. Покажите, что семейство $\mathcal{H} = \{h_{a_0,\dots,a_{k-1}} \mid a_i \in \mathbb{Z}_p\}$ является k-независимым.

Дополнительные задачи

8. Есть множество X и 2-независимое семейство хеш-функций $\mathcal{H} = \{f : X \to \mathbb{F}_2^n\}$. Определим похожесть двух множеств $A \subset X$ и $B \subset X$ как
$$d(A, B) = \frac{|A \cap B|}{|A \cup B|}.$$

Пусть $2^n \gg |A \cup B|^2$, заметим, что:

$$d(A, B) \simeq \underset{h \in \mathcal{H}}{\mathbf{Pr}} \left[\min h(A) = \min h(B)\right].$$

Случайно выбрав из $\mathcal{H}$ k хеш-функций, мы можем вероятностно оценить $d(A, B)$ с точностью порядка $\frac{1}{\sqrt{k}}$ за $\mathcal{O}(k(|A| + |B|))$ вычислений хеш-функций.

(a) Докажите, что написанное выше правда (на самом деле не совсем).

(b) Придумайте, как добиться того же результата с одной хеш-функцией и $\mathcal{O}(|A| + |B|)$ вычислений. Возможно, придется ввести дополнительные требования на семейство.

(c) Придумайте реализацию предыдущего пункта за один проход, $\mathcal{O}(|A| + |B| + k \log k)$ времени и $\mathcal{O}(k)$ памяти (считая, что хеш вычисляется за $\mathcal{O}(1)$).

9. Даны два множества A и B, на которых построены фильтры Блума $\mathcal{A}$ и $\mathcal{B}$ соответственно с одинаковыми размерами и хеш-функциями.

(a) Правда ли, что фильтр Блума, построенный на $A \cup B$ совпадает с $\mathcal{A}|\mathcal{B}$?

(b) Правда ли, что фильтр Блума, построенный на $A \cap B$ совпадает с $\mathcal{A}\&\mathcal{B}$?

(c) Правда ли, что фильтр Блума $\mathcal{A}\&\mathcal{B}$ отвечает положительно на все элементы $A \cap B$?

10. Придумайте способ эффективно удалять ранее добавленные элементы из фильтра Блума (вероятно, потребуется больше в $\log n$ раз памяти).

5.2 Домашнее задание

1. В вашем распоряжении есть 2-независимое семейство хеш-функций $\mathcal{A} = \{f : A \to \mathbb{F}_2^n\}$. Постройте (и докажите, что построено правильно) 2-независимое семейство хеш-функций, которое:

   (a) будет отправлять списки из элементов типа A в $\mathbb{F}_2^n$,

   (b) будет отправлять мультимножества из элементов типа A в $\mathbb{F}_2^n$.

2. Пусть модуль m, по которому берется многочлен в полиномиальном хешировании не фиксирован, а выбирается случайно равновероятно из некоторого конечного набора простых чисел. Пусть константа x для полиномиального хеширования выбирается равновероятно из множества $\{1, 2, \ldots, m-1\}$. Покажите, что такое семейство полиномиальных хеш-функций не является универсальным.

3. Равномерной называется хеш-функция, у которой у каждого значения одинаковое количество прообразов. Пусть есть пара множеств A, B и две равномерные хеш-функции $f : A \to [n]$ и $g : B \to [n]$, где n — натуральное число. Постройте равномерную хеш-функцию с областью значений $[n]$ для:

   (a) Упорядоченных пар (A, B),

   (b) Неупорядоченных пар $\{A, A\}$, если n — нечётное.

   (c) Покажите, что при четном n соответствующей хеш-функции для $\{A, A\}$ может не существовать.

Дополнительные задачи

4. Пусть дана хэш-таблица размера n с хэш-функцией $h : K \to [n]$. На вход поступает n ключей. Будем предполагать, что хэш-функция отправляет каждый ключ в каждую ячейку независимо с равной вероятностью. Коллизии разрешаются с помощью односвязных списков, цепочек. Посчитаем максимальную длину цепочки.

   (a) Зафиксируем хэш-значение x. Доказать, что вероятность, что k ключей будут иметь хэш x составляет
   $$Q_k = \binom{n}{k} \cdot \left(\frac{1}{n}\right)^k \cdot \left(1 - \frac{1}{n}\right)^{n-k}.$$

   (b) Пусть P_k – вероятность максимальной цепочки иметь длину k. Доказать, что $P_k \leq n \cdot Q_k$.

   (c) Вывести из Стрилинга, что $Q_k < (\frac{e}{k})^k$.

   (d) Показать, что для некоторого $c > 1$ верно $Q_k \leq \frac{1}{n^3}$ при $k \geq c \cdot \frac{\log n}{\log \log n}$.

   (e) Доказать, что мат.ожидание длины цепочки не превосходит
   $$\mathbf{Pr}\left[M > c \cdot \frac{\log n}{\log \log n} \right] \cdot n + \mathbf{Pr}\left[M \leq c \cdot \frac{\log n}{\log \log n} \right] \cdot c \cdot \frac{\log n}{\log \log n},$$
   где M – максимальная длина цепочки. Обратите внимание, что M – случайная величина. Вывести оценку сверху $\mathcal{O}(\frac{\log n}{\log \log n})$.

6 Алгебра и криптография

6.1 Практика

1. Дано число n, a и b. Известно, что $a^2 = b^2 (\mathrm{mod}\ n)$, но $a \neq \pm b\ (\mathrm{mod}\ n)$. Найдите нетривиальное разложение n на множители.

2. (a) Заметим, что $gcd(2x, 2y) = 2gcd(x, y)$ и, если x — нечетное число, то $gcd(x, 2y) = gcd(x, y)$. Придумайте `gcd` за $\mathcal{O}(n^2)$.
   (b) Придумайте нерекурсивную версию расширенного Евклида.
   (c) Докажите, что для тождества Безу $Ax + By = gcd(x, y)$, которое вернет расширенный Евклид, если $x, y > 0$, то $|A| \leq y$ и $|B| \leq x$.

3. Дана матрица A размера $n \times m$ над конечным полем, $n \geq m$. Найдите матрицу B размера $m \times n$ такую, что $BA = I_{m \times m}$, или установите, что такой не существует.

4. Найдите базис ядра матрицы над конечным полем.

5. Дано множество натуральных чисел A и параметр b. Известно, что каждое число из A раскладывается в произведение степеней первых b простых чисел (т.е. $\forall a \in A\ a = \prod_{i=1}^{b} p_i^{\alpha_i}$). Требуется найти непустое подмножество A, числа из которого которые в произведении дадут квадрат некоторого натурального числа. Придумайте полиномиальный от b и $|A|$ алгоритм, для поиска такого подмножества.

6. Перед нами стоит задача: получить случайное простое число в диапазоне от 1 до n. Решать ее будем так: выбираем случайное число равновероятно из диапазона и запускаем тест Миллера-Рабина k раз. Если все k тестов прошли, то возвращаем число, иначе — повторяем весь процесс сначала. Этот метод не так плох как кажется, т.к. простые числа достаточно часты (среди чисел от 1 до n примерно $\frac{n}{\ln n}$ простых чисел). Известно, что любое составное число проваливает тест Миллера-Рабина с вероятностью $\frac{3}{4}$, и повторные запуски теста независимы друг от друга. Какое минимальное k нужно выбрать, чтобы получить простое число с вероятностью $\geq 90\%$?

7. Есть множество $A \subseteq [n]$ и k ячеек, каждая из которых умеет хранить $\mathcal{O}(\log n)$ бит информации. В каждый момент времени максимум t из k ячеек могут оказаться недоступны (мы можем узнать про ячейку, доступна ли она, и, если доступна, прочитать данные). Требуется организовать такой способ хранения информации, чтобы в любой момент времени можно было востановить множество A.
   (a) $k = (t + 1) \cdot |A|$.
   (b) $k = t + |A|$.

8. Вам дана система линейных уравнений по модулю n. Решите данную систему при условии, что:
   (a) $n = pq$, p, q — разные простые.
   (b) $n = p^k$, p — простое.
   (c) для произвольного n.

9. Допустим, случайно оказалось, что сообщение, шифруемое RSA, не взаимно просто с n. Сломается ли процедура шифрования/дешифрования? Чем плохо такое сообщение?

10. В d-мерном пространстве над $\mathbb{Q}$ даны точка и набор из k векторов — базис подпространства. Найдите расстояние от точки до подпространства.

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

Дополнительные задачи

12. Дан набор векторов с весами, оболочка векторов совпадает со всем пространством. Выбрать из данных векторов базис минимального суммарного веса.

13. Дана матрица A размера 2×2 над кольцом целых чисел. Придумайте алгоритм, представляющий A в виде $A = LDR$, где D — диагональная матрица, а L и R — обратимые.

14. Аня решила послать приглашение на секретную вечеринку Боре, Ване и Гоше. Аня разослала им одинаковый текст приглашения M закодированный с помощью RSA. У Вани, Бори и Гоши выбраны различные n, но ключ e у всех одинаковый: $e = 3$. Придумайте, как Дима сможет узнать, где будет происходить секретная вечеринка, если ему доступны все три шифрованных приглашения и открытые ключи.

6.2 Домашнее задание

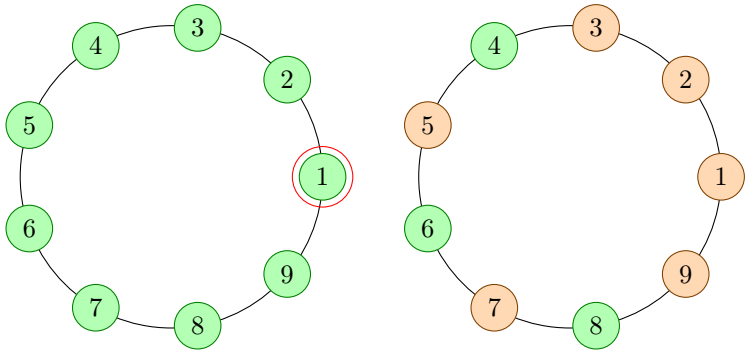
1. Дана матрица A размера $n \times m$. Найти такое представление $A = B \cdot C$, что B имеет размер $n \times k$ и k минимально. $\mathcal{O}(nm(n+m))$. (Подсказка: $k = rank(A)$)

2. Известны открытый ключ $(n, 3)$ и закрытый ключ (n, d) системы `RSA`, при этом n — произведение двух разных простых. Разложите n на множители. $\mathcal{O}(poly(\log n))$.

3. Алена отправила сообщение m, зашифрованное через `RSA`, двум людям. Для каждого человека определено свое e_i, но везде одинаковое $n = pq$, Оказалось, что e_i взаимно простые. Найдите сообщение Алены за $\mathcal{O}(poly(\log n))$.

4. На кольцевой стоят n светофоров и хитро перемигиваются. Свет на каждом светофоре зажигается в таком порядке: зелёный, желтый, красный, снова зелёный и так по кругу. Все светофоры пронумерованы от 1 до n. Если переключить светофор с номером i, то следующие k_i светофоров с шагом a_i тоже автоматически поменяют свой цвет. (Ниже приведен пример, когда переключается первый светофор, $k_1 = 5$, $a_1 = 2$.)

   Приведите алгоритм, который по заданному начальному состоянию светофоров и всем k_i, a_i за время $\mathcal{O}(n^3)$ определяет, возможен ли «зелёный коридор» — ситуация, когда все светофоры переключены на зелёный.



5. У Винни-Пуха есть бесконечные запасы каждого из k видов горшочков мёда. Каждый вид горшочков характеризуется целым числом — сколько дней нужно Винни-Пуху, чтобы съесть весь мёд из одного такого горшка. Винни-Пух уже провёл серию экспериментов вида “взять несколько горшочков мёда, записать день недели, когда эксперимент начался, есть мёд, пока тот не закончится, записать день недели, когда эксперимент закончился” (в наборе может быть несколько горшочков одного вида, в неделе 7 дней). По понятным причинам Винни очень любит экспериментировать. А Кролик не любит мёд, но любит предсказывать будущее. Помогите Кролику определить, можно ли, зная результаты первых k экспериментов и набор горшков и день начала для $(k+1)$-го, предсказать результаты $(k+1)$-го эксперимента.

   (a) Каждый раз Винни берёт произвольные наборы горшков. $\mathcal{O}(k^3)$

   (b) Каждый раз Винни берёт горшки не более чем двух разных типов. $\mathcal{O}(k)$.

6. Для заданного n и простого p найти число таких $k \in [0, n]$, что $\binom{n}{k} = 0 \pmod p$. Можно считать, что все операции по модулю p происходят за $\mathcal{O}(1)$. Придумайте алгоритм, который работает за:

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

   (b) $\mathcal{O}(\text{poly}(\log n))$ (Подсказка: докажите, что $(1+x)^{p^i} = 1 + x^{p^i} \pmod p$, представьте числа в p-ичной системе счисления и попробуйте упростить $(x+1)^n$)

Дополнительные задачи

7. Рассмотрим матрицу $A \in \{0,1\}^{n \times m}$. Для произвольных i и j ($1 \le i \le n$, $1 \le j \le m$) разрешается поменять все значения в строке i и столбце j на противоположные (значение на пересечении строки и столбца меняется). Требуется получить нулевую или единичную матрицу. Придумайте алгоритм за $o((nm)^3)$.

8. В распоряжении взломщиков оказался волшебный оракул, способный для любого открытого ключа (n, e) взломать 1% из возможных зашифрованных протоколом `RSA` сообщений (т.е. детерминированная функция, которая за $\mathcal{O}(1)$ успешно сопоставляет набору (n, e, y) такое число m, что $y \equiv m^e \pmod{n}$, на случайно выбранной сотой части всех возможных входов). Придумайте алгоритм, который взламывает любое сообщение со средним временем работы $\mathcal{O}(poly(\log n))$.

9. Покажите, что задача факторизации принадлежит классу `coNP` (т.е. что задача проверки на простоту принадлежит `NP`).

7 FFT

7.1 Практика

1. Пусть вам известно `FFT`$([a_1, a_2, \cdots, a_n])$. Вычислите `FFT`$([a_2, a_3, \cdots, a_{n+1}])$ за линию.
2. За какое время можно посчитать 2^n в десятичной системе счисления, используя `FFT`?
3. Вам даны две последовательности $\{a_i\}$ и $\{b_i\}$. Их свёрткой называется последовательность:

$$c_i = \sum_{0 \le k \le i} a_k b_{i-k}$$

   Придумайте, как посчитать первые n членов свертки двух последовательностей за $\mathcal{O}(n \log n)$.
4. Даны два n-мерных вектора a и b. Посчитайте скалярные произведения a со всеми циклическими сдвигами b. $\mathcal{O}(n \log n)$.
5. Даны текст t и шаблон p над алфавитом размера k. $|t| \ge |p|$.
   (a) Для каждого из $|t| - |p| + 1$ наложений p на t узнать количество ошибок. $\mathcal{O}(k|t| \log |t|)$.
   (b) Предыдущий пункт с дополнительным условием: в тексте и в шаблоне допустимы символы «?», означающие совпадение с любым символом.
6. Дано уравнение: $x^n + y^n \equiv z^n \pmod m$. Найдите количество его решений за $\mathcal{O}(m \log (n + m))$.
7. Дана чёрно-белая (без оттенков серого) картинка размера $n \times n$ и образец размера $k \times k$, $k \le n$. Найдите наилучшее вхождение образца в картинку за $\mathcal{O}(n^2 \log n)$. Наилучшим называется вхождение с минимальным суммарным количеством различных пикселей.
8. Эльф Леголас — пессимист. Начиная со своего сотого дня рождения каждому дню своей жизни он присваивает рейтинг уныния — действительное число от 0 до 1. Жизнь Леголаса была печальна, но насыщенна: из года в год одни неприятности уступали место другим. Однако последние k лет у Леголаса появилось стойкое ощущение Déjà vu: ему кажется, что происходящие с ним невзгоды сильно напоминают один из предыдущих отрезков его страдальческой жизни, — но вот какой именно, он вспомнить не может. Воспользовавшись дневником рейтинга уныния, помогите Леголасу найти такой период в его жизни, что среднеквадратичное отличие рейтинга уныния каждого дня от соответствующего дня последних k лет минимально. Время $\mathcal{O}(N \log N)$, где N — возраст Леголаса.
9. Даны n предметов и запросы «можно ли набрать вес w_i, используя только предметы с номерами от l_i до r_i». При этом все $w_i \le s$. Сделайте предподсчёт за $\mathcal{O}(ns \log s)$ так, чтобы на запрос можно было бы в online ответить за $\mathcal{O}(s \log s \log n)$.
10. (a) Пусть дан многочлен $p(x)$. Определим многочлен $R(p(x))$ как «разворот» $p(x)$: коэффициент при константе меняется с коэффициентом при $x^{deg(p(x))}$, коэффициент при x меняется с коэффициентом при $x^{deg(p(x))-1}$, и так далее. Докажите, что $R(p_1(x)p_2(x)) = R(p_1(x))R(p_2(x))$.
    (b) Пусть p_1 при делении на p_2 дает неполное частное q и остаток r. Докажите, что $R(p_1) \equiv R(p_2)R(q) \pmod{x^{deg(p_1)-deg(p_2)}}$.
    (c) Придумайте, как найти $R(p_2)^{-1} \pmod{x^k}$ за $\mathcal{O}(n \log n)$.
    (d) Придумайте алгоритм, реализующий деление многочленов степени не более n с остатком за $\mathcal{O}(n \log n)$

7.2 Домашнее задание

1. Дан массив из n целых чисел, по модулю не превосходящих N. За $\mathcal{O}(n + N \log N)$ определите, содержит ли он три (не обязательно различных) числа, дающих в сумме 0.

2. (только группа Мишунина) Найдите количество `AVL`-деревьев высоты h из n вершин по простому модулю p за:

   (a) $\mathcal{O}(hn \log n)$.

   (b) $\mathcal{O}(hN(h))$. Здесь $N(h)$ — максимальное количество вершин в `AVL` дереве высоты h.

   Считайте, что в поле вычетов по модулю p вам известен примитивный корень из единицы достаточной степени.

3. Придумайте, как свести вычисление `FFT` последовательности размера $n = n_1 n_2$ к:

   (a) n_1 вычислениям `FFT` от последовательностей размера n_2 и $\mathcal{O}(n_1^2 n_2)$ дополнительных арифметических операций.

   (b) n_1 вычислениям `FFT` от последовательностей размера n_2, n_2 вычислениям `FFT` от последовательностей размера n_1 и $\mathcal{O}(n_1 n_2)$ дополнительных арифметических операций.

4. Заданы картинка a и образец p в виде матриц вещественных чисел из $[0, 1]$ размерами $n \times n$ и $k \times k$ соответственно ($n \geq k$). Требуется найти позицию (x, y), $0 \leq x \leq n - k$, $0 \leq y \leq n - k$, для которой:

   $$\sum_{i=0}^{k-1} \sum_{j=0}^{k-1} (p_{i,j} - a_{(x+i),(y+j)})^2 \to \min$$

   за время $\mathcal{O}(n^2 \log n)$.

5. Даны строка s длины n и шаблон, в котором могут встречаться вопросительные знаки. Найти подстроку s, точно подходящую под шаблон, за $\mathcal{O}(n \log n)$. Алфавит — **не** константа!

Дополнительные задачи

6. (только группа Лапенка) Найдите количество `AVL`-деревьев высоты h из n вершин по простому модулю p за:

   (a) $\mathcal{O}(hn \log n)$.

   (b) $\mathcal{O}(hN(h))$. Здесь $N(h)$ — максимальное количество вершин в `AVL` дереве высоты h.

   Считайте, что в поле вычетов по модулю p вам известен примитивный корень из единицы достаточной степени

7. Даны строка длины n и шаблон, в котором могут встречаться звёздочки и вопросительные знаки. Проверьте, подходит ли строка целиком под шаблон, за $\mathcal{O}(n\sqrt{n} \log n)$. Алфавит — **не** константа!

8. Перевод из системы счисления в другую быстрее квадрата.

9. Интерполяция в произвольных точках быстрее квадрата.

8 Линейное программирование

8.1 Практика

1. Будем называть *стандартной формой* задачи линейного программирования ограничения $Ax \leq b; x \geq 0$, и *канонической формой* ограничения $Ax = b; x \geq 0$. Приведите задачу в стандартной форме к канонической и задачу в канонической форме к стандартной.

2. Сведите к задаче линейного программирования следующую задачу:

$$\min_{1 \leq i \leq p} \left[\sum_{j=1}^{n} c_{ij} x_j \right] \to \max$$

   При ограничениях:

$$\sum_{j=1}^{n} a_{ij} x_j = b_i, i \in [1 \dots m]; \quad x_i \geq 0, i \in [1 \dots n]$$

3. Пусть

$$\begin{array}{ll} c^\intercal x \to \max & b^\intercal y \to \min \\ Ax \leq b & A^\intercal y \geq c \\ x \geq 0 & y \geq 0 \end{array}$$

   пара из прямой и дуальной задач с оптимальными решениями $x^\star$ и $y^\star$ соответственно.

   (a) Покажите, что дуальной задачей к дуальной является прямая.

   (b) (weak duality) Покажите, что $c^\intercal x \leq b^\intercal y$.

   (c) (complementary slackness) Strong duality утверждает, что $c^\intercal x^\star = b^\intercal y^\star$. Покажите, что условия $y_i^\star > 0$ и $(Ax^\star)_i < b_i$ никогда не выполняются вместе.

   (d) Покажите, что для задачи

$$c^\intercal x \to \max$$
$$Ax \leq b + \Delta b$$
$$x \geq 0$$

   оптимальное значение целевой функции ограничено сверху $c^\intercal x^\star + \Delta b^\intercal y^\star$.

4. Рассмотрим *матричную игру с нулевой суммой*: дана матрица A размера $n \times m$, первый игрок выбирает $i \in [1 \dots n]$, второй $j \in [1 \dots m]$. Выигрыш первого игрока составляет $A_{i,j}$, второго $-A_{i,j}$.

   (a) Пусть стратегия первого (второго) игрока — распределение вероятностей на строчках (столбцах). Первый игрок пытается максимизировать матожидание своего выигрыша в предположении, что второй игрок знает его стратегию. Сведите поиск оптимальной стратегии для первого игрока к задаче линейного программирования.

   (b) Аналогично для второго игрока.

   (c) Постройте дуальные к полученным задачам.

   (d) Покажите, что в данной игре существует *равновесие по Нэшу*. Пара стратегий a и b называется равновесием по Нэшу, если никому не выгодно отказываться от своей стратегии, то есть если первый игрок сменит стратегию a на c, а второй продолжит придерживаться стратегии b, то результат первого не улучшится; аналогично для второго.

5. Постройте дуальную к задаче в канонической форме.

6. Рассмотрим задачу линейного программирования в канонической форме с матрицей A. Докажите, что точка x является вершиной полиэдра системы тогда и только тогда, когда столбцы матрицы A, индуцированные переменными, положительными в точке x, линейно независимы.

7. Матрица M тотально унимодулярна, если любой ее минор равен либо нулю, либо ± 1. Докажите, что если матрица задачи линейного программирования в канонической форме тотально унимодулярна, а вектор целый, то полиэдр данной задачи — целый.

8. Пусть полиэдр задачи линейного программирования целый. Докажите, что решение задачи целочисленного линейного программирования (максимум $c^T \cdot x$) с данными ограничениями совпадает с решением задачи линейного программирования.

9. Сведите к задаче линейного программирования задачу:

$$\frac{\sum c_i x_i + \alpha}{\sum d_i x_i + \beta} \to \min$$

При ограничениях:

$$\sum_{j=1}^{n} a_{ij} x_j = b_i, i \in [1 \dots m]; \quad x_i \geq 0, i \in [1 \dots n]$$

Для простоты будем считать, что знаменатель целевой функции положителен в ограниченной области.

10. Рассмотрим задачу о максимальном паросочетании в графе.

    (a) Сформулируйте эту задачу, как задачу целочисленного линейного программирования.

    (b) Покажите, что в общем случае полиэдр данной задачи может быть нецелым.

    (c) Докажите, что в случае двудольного графа полиэдр полученной задачи целый, но нецелочисленное решение может оказаться бесполезным для нахождения самого паросочетания.

    (d) Какая задача является двойственной к данной?

    (e) ** Пусть полиэдр не целый. Докажите, что если добавить к изначальным условиям такие: «для любого нечетного подмножества вершин U количество индуцированных этим подмножеством ребер не превосходит $\lfloor \frac{|U|}{2} \rfloor$», то полиэдр получится целым.

11. Пусть Вася умеет решать системы линейных неравенств из k уравнений от n неизвестных за время $LN(n, k)$. Васе дали задачу линейного программирования в стандартной форме с целочисленной матрицей A размера $n \times m$. Докажите, что он сумеет решить ее за $LN(n, m+1)n \log(Mn)$, где M — максимальное число, встречающееся в матрице A, векторе или в минимизируемом функционале.

8.2 Домашнее задание

1. Сведите к задаче линейного программирования задачу:

$$\sum_{i=1}^{p} \left| \sum_{j=1}^{n} c_{ij} x_j - d_i \right| \to \min$$

При ограничениях:

$$\sum_{j=1}^{n} a_{ij} x_j = b_i, i \in [1 \dots m]$$

$$x_i \geq 0, i \in [1 \dots n]$$

2. 9 вопросов: приведите пример (или докажите, что такого не бывает), когда прямая задача ЛП {несовместна, неограничена, имеет конечное решение} и дуальная задача {несовместна, неограничена, имеет конечное решение}.

3. Пусть у нас задан орграф $G = (V, E)$ с двумя выделенными различными вершинами $s, t \in V$, для каждого ребра e которого задано вещественное неотрицательное число c_e — его пропускная способность.

   s-t потоком называется функция $f : E \to \mathbb{R}^+$ такая, что:

$$\forall e \in E : 0 \leq f_e \leq c_e$$

$$\forall v \in V \setminus \{s, t\} : \sum_{e=(u,v)} f_e - \sum_{e=(v,u)} f_e = 0$$

   Величиной потока называется:

$$|f| = \sum_{e=(s,u)} f_e - \sum_{e=(u,s)} f_e$$

   s-t разрезом называется пара $(S, T = V \setminus S)$ подмножеств V такая, что $s \in S$ и $t \in T$. Весом разреза называется величина:

$$\sum_{e=(u \in S, v \in T)} c_e$$

   (a) Сведите задачу нахождения максимального s-t потока к задаче линейного программирования в стандартной форме. (*подсказка*: добавьте фиктивное ребро $t \to s$).

   (b) Для ориентированного графа сведите задачу о минимальном s-t разрезе к задаче линейного программирования в стандартной форме. «Сведите» здесь значит, что по любому готовому решению задачи линейного программирования вы должны уметь восстановить какой-то минимальный разрез за $\mathcal{O}(V)$.
   (*подсказка*: назначьте каждой вершине v переменную $0 \leq y_v \leq 1$; у любого целочисленного решения задачи $\{y_v\}$ должны кодировать минимальный разрез. Затем покажите, что любое нецелочисленное решение представляется в виде выпуклой комбинации целочисленных.)

   (c) Покажите, что полученные задачи линейного программирования дуальны друг другу.

Дополнительные задачи

4. (a) Покажите, что матрица инцидентности произвольного орграфа (с $+1$ и -1) тотально унимодулярна.

(b) Покажите, что вершины полиэдра (все, не только оптимальные решения) задачи линейного программирования о минимальном разрезе целочисленны.

5. Придумайте, как написать линейную программу, для поиска максимума такого функционала:

$$\sum_{i,j\in[1..n]} |c_{i,j}(x_i - x_j)|$$

при условиях:

$$Ax = b$$

$$\forall i : x_i > 0$$

6. Найдите равновесие Нэша в игре «Volunteer's dilemma» на n игроков: стратегия каждого игрока — вероятность, с которой он вызывается добровольцем. Если в игре не вызвалось ни одного добровольца, все получают 0, иначе добровольцы получают по c_1, а все остальные — по c_2 ($0 < c_1 < c_2$).

9 Потоки и разрезы

9.1 Практика

1. Дан двудольный граф. Каждой вершине сопоставлено число a_v.
   (a) Выберите максимальное количество рёбер так, чтобы степени вершин были не более 1.
   (b) Выберите максимальное количество рёбер так, чтобы степени вершин были не более a_v.

2. Даны девочки, мальчики и собачки. Для каждой пары "мальчик, девочка" известно, хочет ли девочка дружить с мальчиком. Для каждой пары "собачка, девочка" известно, нравится ли собачка девочке. Нужно максимальному количеству девочек выделить по мальчику и собачке так, что:
   - Каждый мальчик не более чем с одной девочкой.
   - Каждая собачка не более чем у одной девочки.
   - Тройки гармоничны: девочка и хочет дружить с выбранным ей мальчиком, и собачка ей нравится.

3. Дан неориентированный граф. Необходимо ориентировать его так, чтобы максимальная исходящая степень была минимальна.
   (a) $\mathcal{O}(E^2 \log V)$
   (b) $\mathcal{O}(E^2)$

4. По правилам футбольного турнира в каждом матче должна победить одна из команд, то есть, не бывает 'ничьих'. Вам дана матрица уже сыгранный матчей. Можно ли так доиграть турнир, чтобы каждая команда выиграла заданное число раз? Каждая команда играет с каждой, для каждой команды известно, сколько игр она выиграла.

5. Рассмотрим ориентированный граф. За одно действие можно удалить все входящие в вершину i ребра за стоимость $a_i \geq 0$, или все исходящие из вершины i рёбра за стоимость $b_i \geq 0$. Необходимо удалить все рёбра графа за минимальную стоимость.

6. Каждой вершине ориентированного графа сопоставлено число (не обязательно положительное) — её вес. Найдите замкнутое подмножество вершин максимальной суммарной стоимости. Подмножество вершин называется замкнутым, если из него не исходят рёбра в другую часть графа.

7. Есть ориентированный граф с начальной и конечной вершинами. В начальной вершине есть K грузовиков. Грузовикам нужно попасть в конечную вершину. Время дискретно. За единицу времени каждый грузовик или стоит на месте, или перемещается в одну из соседних вершин. В любой вершине могут одновременно стоять несколько грузовиков. По любому из рёбер в каждый момент времени должен ехать не более чем один грузовик. Минимизируйте время, когда все грузовики окажутся в конечной вершине.
   (a) $\mathcal{O}(poly(V, E, K))$
   (b) $\mathcal{O}(K(V + K)E)$

8. Найдите максимальную по числу вершин двудольную клику (полный двудольный подграф) в двудольном графе за $\mathcal{O}(VE)$.

9. За один ход можно покрасить один произвольный вертикальный или горизонтальный отрезок матрицы $n \times m$ в белый цвет, мазки могут перекрываться. Приведите чёрную матрицу к заданному виду за минимальное число ходов. $\mathcal{O}(poly(n, m))$.

10. Найдите ориентированный граф с целочисленными пропускными способностями, на которых детерминированный алгоритм Форда-Фалкерсона на основе `DFS` с фиксированным порядком просмотра рёбер в каждой вершине, **пропускающий максимум по дополняющему пути**, работает за экспоненту от V.

11. Есть клеточное поле $n \times m$. Некоторые клетки свободны, некоторые заняты горами. Есть два замка. Нужно построить в некоторых клетках стены, так чтобы нельзя было пройти от одного замка к другому (нельзя ходить по горам и стенам). Какое минимальное число клеток надо застроить?

12. Докажите реберную теорему Менгера: минимальное число ребер, которые необходимо удалить в графе, чтобы из s в t не было пути, равно максимальному числу реберно непересекающихся путей из s в t.

13. Докажите вершинную теорему Менгера: минимальное число вершин, которые необходимо удалить в графе, чтобы из s в t не было пути, равно максимальному числу вершинно непересекающихся путей из s в t (s и t удалять нельзя).

9.2 Домашнее задание

1. Разбейте вершины ориентированного графа на циклы. Т.е. каждая вершина должна быть покрыта ровно одним циклом. Либо скажите, что это невозможно. $\mathcal{O}(EV)$.

2. Дан граф и выделенные вершины s, t. Нужно проверить, правда ли существует единственный минимальный s-t разрез.

   (a) $\mathcal{O}(poly(V, E))$

   (b) $\mathcal{O}(E)$ при условии, что нам уже известен максимальный поток (сам поток, не только его величина).

3. В неориентированном графе без кратных рёбер необходимо удалить минимальное число рёбер так, чтобы увеличилось количество компонент связности.

   (a) $\mathcal{O}(V \cdot Flow)$, где $Flow$ — время работы какого-нибудь алгоритма поиска максимального потока.

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

4. Даны n рабочих и m работ и матрица умений: "какой рабочий какие работы умеет делать". Необходимо максимально равномерно распределить работы между рабочими. Другими словами, каждой работе сопоставить рабочего, который умеет делать эту работу, и минимизировать $\max\limits_{i=1..n} k_i$, где k_i – количество работ, выданных i-му рабочему. $\mathcal{O}(nm^2)$.

5. Есть ориентированный граф с начальной и конечной вершинами. В начальной вершине есть K грузовиков. Грузовикам нужно попасть в конечную вершину. Время дискретно. За единицу времени каждый грузовик или стоит на месте, или перемещается в одну из соседних вершин. В любой вершине могут одновременно стоять несколько грузовиков. По любому из рёбер в каждый момент времени должен ехать не более чем один грузовик. Минимизируйте время, когда все грузовики окажутся в конечной вершине.

   (a) $\mathcal{O}(poly(V, E, K))$

   (b) $\mathcal{O}(K(V + K)E)$

Дополнительные задачи

6. Есть заказы и инструменты. Для каждого заказа известен список инструментов, который нужен, чтобы его выполнить. Каждый инструмент сделан умелыми японскими рабочими, поэтому бесконечно прочный, его можно один раз купить и много раз использовать. У каждого инструмента есть цена p_i. У каждого заказа есть прибыль, которую можно получить, выполнив заказ. Вы – бедный китайский рабочий. У вас изначально нет инструментов, но зато вы можете под нулевой процент в банке взять сколь угодно большой кредит, чтобы купить инструментов.

   (a) Вопрос: какую максимальную прибыль вы можете получить?

   (b) А теперь тот же вопрос, но ещё есть разные скидочные предложения!
   Скидка позволяет два инструмента i, j купить по специальной цене d: $\max(p_i, p_j) < d < p_i + p_j$. Каждый инструмент присутствует не более чем в одном скидочном предложении.

7. Пусть алгоритм Форда-Фалкерсона ищет дополняющие пути `DFS`-ом с фиксированным порядком просмотра рёбер в каждой вершине, и пропускает максимум по найденному дополняющему пути. Постройте вместе с порядком ребер (или докажите, что такого не существует) ориентированный граф:

(a) с целочисленными пропускными способностями, на котором алгоритм работает за экспоненту от V (пропускные способности могут быть большими, но должны быть полиномиальной суммарной длины).

(b) с вещественными пропускными способностями, на котором алгоритм не завершается за конечное количество шагов.

(c) с вещественными пропускными способностями, на котором алгоритм не сходится к корректному значению величины максимального потока.

8. Дана укладка неориентированного планарного графа: вершинам сопоставлены точки на плоскости, рёбрам — непересекающиеся отрезки между вершинами. У рёбер есть пропускные способности. Даны две вершины s и t, лежащие на одной грани. Требуется за $\mathcal{O}(\texttt{Dijkstra})$ найти величину максимального потока из s в t.

9. Какое максимальное количество уголков можно разместить на шахматной доске $n \times m$ с дырками? Уголком называется фигура, состоящая из трех клеток: центральная клетка черного цвета и две соседних с ней белых клетки со смежными сторонам.

10 Поиск подстроки в строке

10.1 Практика

1. Посчитайте z-функцию строки за $\mathcal{O}(n)$.
2. Найдите число различных подстрок строки. $\mathcal{O}(n^2)$.
3. Найдите все периоды строки. $\mathcal{O}(n)$.
4. Найдите позицию строки в ее суффиксном массиве. $\mathcal{O}(n)$.
5. Научиться искать за $\mathcal{O}(n+m)$ образец в строке, если между образцом и найденной подстрокой допустимо различие:
   (a) в один символ.
   (b) в два символа.
6. Дана строка длины n и образец длины m над алфавитом Σ. Найдите образец в строке за $\mathcal{O}(n+m+|\Sigma|)$, если допустимо в образце применять к алфавиту:
   (a) циклический сдвиг алфавита.
   (b) произвольную перестановку алфавита
7. Дан набор строк суммарной длины n над алфавитом Σ.
   (a) Отсортируйте за время $\mathcal{O}(n \log |\Sigma|)$.
   (b) Покажите, что за $\mathcal{O}(n)$ нельзя.
8. Дан массив a длины n из m-битных чисел. Найдите пару $a_i, a_j : a_i \oplus a_j = \max$. $\mathcal{O}(nm)$.
9. 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 за время $\mathcal{O}(L+l|T|)$. Слова можно использовать более одного раза.
10. Даны словарь и текст. Нужно уметь обновлять ответ `online` при добавлении символов в конец текста.
    (a) Пересчитайте суммарное число вхождений слов из словаря в текст за $\mathcal{O}(1)$.
    (b) Поддерживайте множество всех вхождений слов из словаря в текст. Пересчёт за время $\mathcal{O}(1+|\Delta A|)$, где ΔA – приращение ответа после добавления очередного символа.
11. Дан словарь слов суммарной длины L. За время $\mathcal{O}(L)$ определите, существует ли бесконечная строка, не содержащая ни одно словарное слово как подстроку.
12. За $\mathcal{O}(n)$ построить строку с данной:
    (a) Z-функцией.
    (b) префикс-функцией.
13. Придумайте, как про строку s детерминировано проверить за $\mathcal{O}(|s|)$, что:
    (a) она лексикографически меньше любого ее суффикса.
    (b) она лексикографически небольше любого ее циклического сдвига.

10.2 Домашнее задание

1. Для каждого префикса строки найти количество его префиксов, равных его суффиксу. $\mathcal{O}(n)$.
2. Преобразовать Z-функцию в префикс-функцию без промежуточного восстановления строки за $\mathcal{O}(n)$.
3. За $\mathcal{O}(n)$ построить строку с данной префикс функцией.
4. Найти все подпалиндромы заданной строки за $\mathcal{O}(n)$.
5. 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 за время $\mathcal{O}(L+l|T|)$. Слова можно использовать более одного раза.

Дополнительные задачи

6. За одну секунду в конец изначально пустого текста дописывается случайная буква (равномерное распределение). Чему равно матожидание времени T, когда первый раз s станет подстрокой выписанного текста?
7. В словаре могут добавляются и удаляются слова. Необходимо в `online` научится отвечать на запрос `get(t)` вида "входит ли в текст t хоть одно словарное слово". Амортизированное время работы `add(s)` и `del(s)`: $\mathcal{O}(|s|\log L)$, время работы `get(t)`: $\mathcal{O}(|t|\log L)$ (L — суммарная длина всего). Подсказка: заведите порядка $\log L$ боров.

11 Суффиксные структуры

11.1 Практика

1. Постройте строчку, у которой суффиксный массив совпадает с данным. $\mathcal{O}(n)$.
2. Нужно научится online лексикографически сравнивать подстроки строки за $\mathcal{O}(1)$, имея суффиксный массив строки и $\mathcal{O}(n)$ времени на предобработку.
3. Построить за линейное время:
   (a) суффмассив по суффдереву
   (b) суффдерево по суффмассиву + `lcp`
4. Найдите самую длинную общую подстроку k строк суммарной длины n за $\mathcal{O}(n)$.
   (a) Суффиксным массивом.
   (b) Суффиксным деревом.
5. Посчитать количество различных подстрок строки.
6. Найти самую длинную подстроку, входящую в заданную дважды:
   (a) вхождения могут пересекаться. $\mathcal{O}(n)$
   (b) вхождения не могут пересекаться. $\mathcal{O}(n)$
7. У вас были строка $s_0 s_1 \dots s_{n-1}$ длины n и ее суффиксный массив — массив позиций начал суффиксов, отсортированных в лексикографическом порядке. Символы строки — натуральные числа, не превосходящие n. Под покровом темноты подлые враги прокрались и стерли из массива все числа, делящиеся на 3, так, что даже дырок не осталось! Опишите алгоритм, восстанавливающий суффиксный массив за время $\mathcal{O}(n)$.
8. Дан словарь суммарной длины n, постройте за $\mathcal{O}(n \log n)$ структуру, чтобы быстро отвечать на запросы
   (a) `get(s)` – число строк в словаре, которые начинаются с s, заканчиваются на $rev(s)$. $\mathcal{O}(|s|)$.
   (b) `get(s,t)` – число строк в словаре, которые начинаются с s, заканчиваются на t, $|s| = |t|$. $\mathcal{O}(|s|)$.
   (c) `get(s,t)` – число строк в словаре, которые начинаются с s, заканчиваются на t. $\mathcal{O}(|s| + |t| + \log n)$.
9. Постройте строчку над минимальным алфавитом, у которой суффиксный массив совпадает с данным. $\mathcal{O}(n)$.
10. Дан набор строк s_i. Для каждой s_i найдите min по длине подстроку, которая не встречается в других. $\mathcal{O}(n)$.
11. Дан последний столбец отсортированного массива циклических сдвигов строки. Также дан номер исходной строки в этом порядке. Восстановить строку.
    (a) $\mathcal{O}(n^2)$
    (b) $\mathcal{O}(n)$

11.2 Домашнее задание

Дополнительные задачи

1. За одну секунду в конец изначально пустого текста дописывается случайная буква (равномерное распределение). Чему равно матожидание времени T, когда первый раз s станет подстрокой выписанного текста?

2. Найдите за $\mathcal{O}(n)$ максимальный рефрен — такую подстроку строки s, что количество ее вхождений (возможно, пересекающихся), помноженных на ее длину, максимально. Решите двумя способами: суффиксными деревом и массивом.

3. Дан набор строк s_i суммарной длины n. Для каждой s_i найдите min по длине подстроку, которая не встречается в других. $\mathcal{O}(n)$.

4. Дан словарь $s_1, s_2, \ldots, s_n$. Отвечать в offline на запросы `get(t, l, r)` – сколько подстрок из $\{s_l, \ldots, s_r\}$ входят в текст t как подстроки. Время работы – линия от размера входа на полилог.

5. Рассмотрим задачу `EXACT 4SAT`. Это `SAT` для формул, в которых каждая дизъюнкция состоит ровно из 4 литералов. Докажите, что эта задача `NP`-полна.

6. Рассмотрим следующую игру:
   Дан ориентированный граф, каждому ребру которого сопоставлено целое число. На одной из его вершин стоит фишка. Два игрока ходят по очереди, каждый в свой ход может передвинуть фишку по ребру и заплатить противнику противнику сумму, соответствующую этому ребру. Требуется проверить, существует ли у первого игрока стратегия, позволяющая ему бесконечно обогащаться. Докажите, что такая задача лежит в $\texttt{NP} \cap \texttt{coNP}$. Можно рассматривать только чистые стратегии.

7. Дана укладка планарного графа. Вершинам сопоставлены точки на плоскости, рёбра – отрезки между вершинами, рёбра не пересекаются. У рёбер есть пропускные способности. Граф неориентированный. Даны две вершины s и t, лежащие на одной грани. Задача: за $\mathcal{O}(\texttt{Dijkstra})$ найти величину максимального потока из s в t.

8. Разбить массив на минимальное число подпоследовательностей таких, что в каждой подпоследовательности разность соседних элементов по модулю не превышает X. $\mathcal{O}(n^3)$.

9. Есть k регионов памяти и n программ. У каждого есть размер s_i. У каждой программы есть необходимое ей количество памяти x_j и время выполнения t_j. Каждой программе нужно сопоставить номер региона памяти i_j, в котором она будет выполняться, и отрезок времени выполнения $[l_j, l_j+t_j)$. Для каждого региона памяти отрезки времени выполнения программ не должны пересекаться. Минимизировать $\sum_j l_j$.

10. Два игрока по очереди делают ходы на двудольном графе: передвигают фишку в одну из смежных вершин и удаляют ребро. Проигрывает тот, кто не может сделать ход. Определите, кто выиграет при оптимальной игре. Время $\mathcal{O}(poly(V, E))$.

11. Постройте строчку над минимальным алфавитом, у которой суффиксный массив (без попарных `lcp`) совпадает с данным. $\mathcal{O}(n)$.

12. Дан массив a длины n и число k. Элементы массива и k — m-битные числа. За время $\mathcal{O}(nm)$:

    (a) посчитайте количество таких пар индексов массива, что строка длины m, являющаяся побитовым `xor` элементов по этим индексам, $\geq k$.

    (b) посчитайте количество подмассивов a, побитовый `xor` всех чисел из которых $\geq k$.

    (c) найдите полмассив a, побитовый xor всех чисел из которого максимален.

13. Придумайте, как, используя суффиксный массив, реализовать оптимальное сжатие `LZSS` за $\mathcal{O}(n \log n)$.

14. Дан набор векторов с весами, оболочка векторов совпадает со всем пространством. Выбрать из данных векторов базис минимального суммарного веса.

15. Дана матрица A размера 2×2 над кольцом целых чисел. Придумайте алгоритм, представляющий A в виде $A = LDR$, где D — диагональная матрица, а L и R — обратимые.

16. Покажите, что `P` $\neq$ `NP`.