«Решения теоретических ДЗ»
ФПМИ МФТИ

Автор решения

Осень 2022

Содержание

Вступление для прочтения **4**

Вместо приветствия . . . 4

Немного стандартных примеров . . . 4

Немного математических примеров . . . 5

Решение первого задания **6**

№ 1 . . . 6

№ 2 . . . 7

№ 3 . . . 8

Решение второго задания **8**

№ 1 . . . 8

№ 2 . . . 9

№ 3 . . . 9

Решение третьего задания **10**

№ 1 . . . 10

№ 2 . . . 11

№ 3 . . . 12

№ 4 . . . 12

Решение четвёртого задания **13**

№ 1 . . . 13

№ 2а . . . 13

№ 2б . . . 13

Решение пятого задания **14**

№ 1 . . . 14

№ 2 . . . 15

№ 3 . . . 15

№ 4а . . . 16

№ 5 . . . 17
№ 7 . . . 17

Решение десятого задания **18**
№ 1 . . . 18
№ 2 . . . 19
№ 3 . . . 19
№ 4 . . . 19
№ 5 . . . 20

Решение одиннадцатого задания **20**
№ 1 . . . 20
№ 2 . . . 21
№ 4 . . . 21
№ 5 . . . 22

Решение двенадцатого задания **23**
№ 1 . . . 23
№ 2 . . . 23
№ 3 . . . 24
№ 4 . . . 25
№ 5 . . . 25

Решение тринадцатого задания **26**
№ 1 . . . 26
№ 2 . . . 27
№ 3 . . . 28

Additional information **29**

Вступление для прочтения

Вместо приветствия

Приветствую, читатель! Наверное, ты задаешься вопросом, а что это такое? Это аналог Google Docs, только тут еще есть LaTeX. Стоп, что такое LaTeX? Это язык для типографской математической верстки, которым пользуется подавляющее большинство людей, публикующих свои статьи и/или просто пишущих техническую литературу. Зачем мне это надо? Ну, отныне все теоетические домашки будут вами верстаться в этой среде, так что дерзайте.

Имеются папки `etc`, `pic` и `solutions`. В них, соответственно, добавочные материалы/ссылки, картинки, которые вы используете (решение в виде вставленной картинки приниматься не будет), это вспомогательные иллюстрации, и, собственно, папка с решениями. Еще в корне лежат файлы `main.tex` — там лежит шапка документа и нужные инпуты (какие файлы подключать при компиляции), `style.tex` — подключение различных пакетов и переопределение команд (для некоторых подписано комментарием, что оно делает — пользуйтесь с удовольствием).

Перейдем к папке с решениями. Там надо создать папку с названием «S_T», где S — номер ~~сезона~~ семестра, а T — номер домашнего задания. Изначально у вас первое. Далее в этой папке создать файлы вида `N.tex`, где N — номер задачи, которую вы решаете. Шаблон есть в `solutions/2_3/1.tex`, его желательно соблюдать (но можно украсить по своему желанию). Еще есть файлы `list.tex`, они нужны для того, чтобы подключать всю директорию как один файл. То есть аггрегируют информацию. А еще в них расставлено секционирование документа, тут рекомендуется не менять секционирование, так как оно вам поможет выжить в дальнейшем. Пользуйтесь и радуйтесь.

Немного стандартных примеров

Часто в ходе решения вам захочется использовать списки и перечисления. Для их использования необходимо использовать окружения `enumerate` (пронумерованный список) или `itemize` (непронумерованный вариант). Например,

1. Текст
2. Нижний текст

- Важное утверждение
- Еще одно важное утверждение

Также можно создавать вложенные перечисления:

1. А тут вложенное перечисление

   (a) Первый пункт второго уровня
   (b) А тут ненумерованное перечисление
      - Один пункт третьего уровня
      - Еще один пункт третьего уровня
   (c) Третий пункт второго уровня
2. Второй пункт первого уровня

Возможно вам понадобится *курсив* `textit` или **выделение** `textbf`.

Немного математических примеров

Время математических примеров. Во-первых, написание формул.

Однострочная формула из матстатистики

$$p_\theta(x) = \exp\left\{\sum_{i=1}^{k} b_i(\theta)\cdot T_i(x) + d(\theta) + S(x)\right\}\cdot I_{\mathcal{A}}(x), \quad \theta = (\theta_1,\ldots,\theta_k)$$

Или, быть может, вывод многострочной?

$$\widetilde{\mathbb{E}}\left[b(X,\theta)\right]^2 = \int\limits_{\mathbb{R}^n}\int\limits_{\mathbb{R}}\left(\frac{\partial}{\partial t}\ln\left(p_t(x)\cdot q(t)\right)\right)^2\cdot p_t(x)\cdot q(t)dxdt =$$

$$= \int\limits_{\mathbb{R}^n}\int\limits_{\mathbb{R}}\left(\frac{\partial \ln p_t(x)}{\partial t} + \frac{\partial \ln q(t)}{\partial t}\right)^2\cdot p_t(x)\cdot q(t)dxdt =$$

$$= \int\limits_{\mathbb{R}^n}\int\limits_{\mathbb{R}}\left(\frac{\partial \ln p_t(x)}{\partial t}\right)^2\cdot p_t(x)\cdot q(t)dxdt + \int\limits_{\mathbb{R}^n}\int\limits_{\mathbb{R}}\left(\frac{\partial \ln q(t)}{\partial t}\right)^2\cdot p_t(x)\cdot q(t)dxdt +$$

$$+ 2\int\limits_{\mathbb{R}^n}\int\limits_{\mathbb{R}}\frac{\partial q(t)}{\partial t}\cdot\frac{1}{q(t)}\cdot\frac{\partial p_t(x)}{\partial t}\cdot\frac{1}{p_t(x)}\cdot p_t(x)\cdot q(t)dxdt = I_p + I_q + 2\int\limits_{\mathbb{R}^n}\int\limits_{\mathbb{R}}\frac{\partial q(t)}{\partial t}\cdot\frac{\partial p_t(x)}{\partial t}dxdt =$$

$$= I_p + I_q + 2\int\limits_{\mathbb{R}}\frac{\partial}{\partial t}\left(\int\limits_{\mathbb{R}^n} p_t(x)dx\right)\cdot\frac{\partial q(t)}{\partial t}dt = I_p + I_q + 2\int\limits_{\mathbb{R}}\frac{\partial(1)}{\partial t}\cdot\frac{\partial q(t)}{\partial t}dt = I_p + I_q$$

На выбор есть вывод набора отцентрированных формул

$$A^T = \left(Z\left(Z^TZ\right)^{-1}Z^T\right)^T = A$$

$$A^2 = \left(Z\left(Z^TZ\right)^{-1}Z^T\right)^2 = Z\left(Z^TZ\right)^{-1}Z^TZ\left(Z^TZ\right)^{-1}Z^T = A$$

Или же, наоборот, выровненных по знаку?

$$I_p = \widetilde{\mathbb{E}}\left[\left(\frac{\partial}{\partial\theta}\ln p_\theta(X)\right)\cdot\left(\frac{\partial}{\partial\theta}\ln p_\theta(X)\right)^T\right]$$

$$I_q = \widetilde{\mathbb{E}}\left[\left(\frac{\partial}{\partial\theta}\ln q(\theta)\right)\cdot\left(\frac{\partial}{\partial\theta}\ln q(\theta)\right)^T\right]$$

А может вам угодно написать матрицу?

$$\begin{pmatrix} S_1 \\ \vdots \\ S_n \end{pmatrix} = \begin{pmatrix} 1 & 0 & & \dots & 0 \\ 1 & 1 & 0 & \dots & 0 \\ 1 & & \ddots & & \vdots \\ \vdots & & & \ddots & 0 \\ 1 & & \dots & & 1 \end{pmatrix} \begin{pmatrix} \xi_1 \\ \vdots \\ \xi_n \end{pmatrix}$$

Решение первого задания

№ 1

Условие первой задачи.

На прямой дощечке вбито n гвоздиков. Любые два гвоздика можно соединить ниточкой (но нельзя соединять гвоздик сам с собой). Требуется соединить некоторые пары гвоздиков ниточками так, чтобы к каждому гвоздику была привязана хотя бы одна ниточка, а суммарная длина всех ниточек была минимальна. Асимптотика: $O(n \log n)$.

Решение первой задачи.

Для начала отсортируем гвоздики по возрастанию координат, после чего заведём два массива длины n. В первом массиве на i-ом месте мы будем хранить суммарную длину ниточек, если мы привзяываем i-ый гвоздб к предыдущему, а на соответсвующем месте во втором массиве будем хранить длину, если мы обязуемся привязать i-ый к следующему. Кроме того, нулевой элемент первого массива будет равен бесконечности, ведь мы не можем привязать самый левый гвоздб к предыдущему, так как такого нет, а нулевой элемент второго массива будет равен нулю. Ответ же будет лежать в последнем элементе первого массива, так как мы не можем связать самый правый гвоздь с последующим. Первый массив будет называтсья $tied\_with\_previous$, а второй соответственно $tied\_with\_next$. Формула пересчёта:
$tied\_with\_previous[i] = \min(tied\_with\_previous[i-1], tied\_with\_next[i-1]) + coordinates[i] - coordinates[i-1]$
$tied\_with\_next[i] = tied\_with\_previous[i-1]$

№ 2

Условие второй задачи.

Нужно перевезти n объектов, стоящих в ряд, их веса равны $a_1, a_2, \ldots, a_n$. Корабль за одну переправу может перевезти лишь грузов суммарного веса не больше t. В каждый момент времени грузить на корабль разрешается только первый или последний объект, который ещё не был погружен. Иными словами, за одну переправу можно перевезти некий префикс и некий суффикс необработанных объектов. За $O(n^2)$ определите минимальное число переправ корабля для перевозки всех объектов.

Решение второй задачи.

Заведём таблицу $n \times n$, в каждой клетке будем содержаться пара элементов: 1-ый отвечает за количество переправ, которое нужно выполнить, чтобы перевезти предметы с 1-го по i-ый и с j-го по n-ый, а 2-ой отвечает за количество свободного места, которое есть в корабле, совершающем последнюю переправу. Кроме того, считаем, что у таблички есть фиктивная нижняя строка и фиктивный правый столбец. База динамики:

$$dp[0][n] = (0, 0)$$

Далее считаем справа-налево

$$dp[0][j] = \begin{cases} (dp[0][j+1].first, dp[0][j+1].second - weight[j]) & weight[j] \le d[[0][j+1].second \\ (dp[0][j+1].first + 1, t - weight[j]) & weight[j] > dp[0][j+1].second \end{cases}$$

Здесь считаем снизу-вверх

$$dp[i][n] = \begin{cases} (dp[i-1][n].first, dp[i-1][n].second - weight[i]) & weight[i] \le d[[i-1][n].second \\ (dp[i-1][n].first + 1, t - weight[i]) & weight[i] > dp[i-1][n].second \end{cases}$$

Далее считаем внешним циклом сверхку-вниз, а внутренним – справа-налево. Фактически мы просто выбираем, лучше ли этот элемент взять с правого конца или с левого, имея оптимальный ответ для меньших суффиксов и префиксов:

$$dp[i][j] = \begin{cases} (dp[i][j+1].first, dp[i][j+1].second - weight[j]) & weight[j] \le d[[i][j+1].second \\ (dp[i][j+1].first + 1, t - weight[j]) & weight[j] > dp[i][j+1].second \end{cases} \quad (1)$$

$$dp[i][j] = \begin{cases} (dp[i-1][j].first, dp[i-1][j].second - weight[i]) & weight[i] \le d[[i-1][j].second \\ (dp[i-1][j].first + 1, t - weight[i]) & weight[i] > dp[i-1][j].second \end{cases} \quad (2)$$

$dp[i][j] = \min((1), (2))$

Ответ будет минимумом среди диагональных элементов, так как именно в них мы в итоге перевезём все элементы с 1-го по n-ый.

Так как мы имеем табличку размером $n \times n$, то программа работает за $O(n^2)$

№ 3

Условие первой задачи.

Предложите метод решения задачи о рюкзаке с восстановлением ответа, использующий $O(W\sqrt{n})$ памяти и $O(nW)$ времени. Здесь n — число объектов, а W — вместимость рюкзака.

Решение первой задачи.

Будем считать задачу как обычный рюкзак, но с некоторыми допущениями: в процессе подсчёта сохраним 0-ой, $\sqrt{n}$-ый, $2\sqrt{n}$-йы, $\ldots$,n-ый. На это у нас уйдёт $O(nW)$ времени и $O(W\sqrt{n})$ памяти. После чего для восстановления ответа будем запускать "задачу о рюкзаке"между каждыми двумя блоками, начиная с последнего и предпоследнего. Мы знаем начальное и конечное положение динамики, а также используемые предметы, соответственно сможем в каждом блоке выяснить предметы, которые нужно взять. Это затребует $O(W\sqrt{n} * \sqrt{n}) = O(Wn)$ времени, так как у нас $\sqrt{n}$ блоков по $\sqrt{n}$ столбцов по W элементов. Памяти будет затребовано на каждый блок по $O(W\sqrt{n})$, а значит всего времени и памяти будет $O(Wn)$ и $O(W\sqrt{n})$ (игнорируя константу)

Решение второго задания

№ 1

Условие первой задачи.

Задан массив $a(0), \ldots, a(2n1)$. Определим $a'(mask) = \sum\limits_{submask \subseteq mask} a(submask)$. Докажите, что следующий код решает эту задачу на месте (результат сохраняется в исходном массиве).

```
void magic(vector<int>& a) {
    for (int i = 0; i < n; ++i)
        for (int mask = 0; mask < (1 << n); ++mask)
            if (!bit(mask, i))
                a[mask + (1 << i)] += a[mask];
}
```

Решение первой задачи.

Докажем факт аналогичный условию задачи, которую решает исследуемый алгоритм: из любой маски мы дойдём до всех её "надмасок причём едиснтвенным путём, это обеспечит, что в каждом элементе нового массива мы будем иметь сумму по всем подмаскам, а также ни одна подмаска не

будет подсчитана дважды.
Рассмотрим теперь 2 маски A и B, таких, что $A\&B = A$, т.е. A — подмаска B. Теперь рассмотрим все биты, которые отличаются у A и B, а также упорядочим их по возрастанию. Чтобы добраться до B нам надо сделать количество изменений одного бита равное количеству различающихся битов, что очевидно $\leq n$, значит добраться мы точно сможем, осталось разобраться, нет ли второго пути.
Рассмотрим самый быстрый путь: он делает из различающихся битов единички по очереди(так как мы идём по возрастанию номера битов, то быстрее нельзя). Пусть существует другой путь, который также начинается в A и заканчивается в B, раз он не совпадает с первым, то в какой-то итерации он не сделал из очередного нуля единицу, но тогда он не не может быть нужным путём, ведь пропущенный 0 больше не сможет стать 1, так как по каждому биту проходимся лишь. Значит путь единственен
Так как мы можем взять таким образом две любые вершины с отношением $submask - mask$, то в массиве после алгоритма будет действительно содеражаться ответ.

№ 2

Условие второй задачи.

Пусть в задаче о рюкзаке предметы не имеют стоимостей, то есть характеризуются только весами. Нужно найти максимальный суммарный вес предметов, который можно уместить в рюкзак вместимости W . Решите задачу за $O(nW/w)$, где w — длина машинного слова (обычно 32 или 64).

Решение второй задачи.

В обычном рюкзаке для каждого предмета мы проходимся по всем возможным весам и для каждого из них определяем, можем ли мы его набрать. Заведём вместо привычного массива битсет, размером с W, тогда на каждой итерации цикла по предмету нам нужно будет всего лишь сделать операцию $dp = dp|(dp << w_i)$, где dp - битсет, w_i — вес i-го предмета, а i-ый бит отвечате за возможность набрать вес i, тогда ответом будет максимум среди номеров единичных битов. Так как операция сразу над 32 или 64 возможными весами (так как в битсете мы храним будет занимать столько же времени, сколько операция с одним в обычной реализации, то асимптотика данного алгоритма будет $O\left(\frac{nW}{\omega}\right)$

№ 3

Условие третьей задачи.

На гранях шестигранного кубика могут располагаться числа от 1 до n, повторы не запрещены. Два кубика считаются различными, если на кубиках различны мультимножества расположенных чисел. Скажем, что один кубик превосходит другой, если с вероятностью, строго большей 2 1 ,

при случайном равномерном бросании обоих кубиков на первом выпадает большее число. Назовём тройку кубиков хорошей, если первый кубик превосходит второй, второй превосходит третий, а третий превосходит первый. Определите число хороших упорядоченных троек кубиков за

a (2 балла) $O(n)$;

b (1 балл) $O(\log n)$;

c (2 балла) $O(1)$

Решение третьей задачи.

Так как в задаче нет ограничений по памяти, то воспользуемся предподсчётом (в конечных пределах, не зависящих от входных данных). Очевидно, что у нас может быть максимум 18 различных значений на кубиках. Значит мы можем в офлайне посчитать всевозможные комбинации на кубиках,* а также то, подходят ли они нам (например, простым перебором всех возможных значений, которых ≤ 18 для грани, нам важно, чтобы был именно порядок, т.е. число на одной грани $<, >, =$ числу на другой). При запуске же алгоритма нам нужно будет лишь понять, сколько способов расставить значения по граням, чтобы они удовлетворяли подходящим наборам.
Тогда для каждого набора, коих константа, нужно посчитать количество сопсобов подставить числа из множества $\{1, \ldots, n\}$
В подходящем наборе не больше 18 чисел, так что фактически для каждого набора мы хоти посчитать количество способов поставить перегородки в множестве из n чисел, ведь нам важно, чтобы сохранялся порядок в наборе, а числа не так важны. Соответственно нам нужны сочетания: C_{n-1}^{k}, где k - количество различных чисел в наборе. Так как $k \leq 18$, то факт наличия в формуле сочетаний факториала не портит нам ассимптотику и фактически это считатеся за константу, ведь бы берём не более 18 чисел до n включительно и делим их на $k!$
В итоге мы имеем предподсчёт, после чего константное количество расчётов, производимых за константу, а значит в итоге асимптотика будет $O(1)$, хоть и с большой константой.

* Комбианции, которые сохраняют поярдок между элементами. А таких уже **конечное** число, не зависящее от n

Решение третьего задания

№ 1

Условие первой задачи

Для получения полного балла за задачу достаточно доказать эквивалентность любых четырёх условий, среди которых будут а) и б). Пусть G — связный граф хотя бы на трёх

вершинах. Здесь все циклы и пути подразумеваются **рёберно-простыми**. Докажите, что следующие условия эквивалентны:

a) в графе G нет мостов;

b) между любыми двумя вершинами есть два не пересекающихся по рёбрам пути;

c) любые две вершины принадлежат некоторому циклу;

d) любая вершина и любое ребро принадлежат некоторому циклу;

e) любые два ребра принадлежат некоторому циклу;

f) для любых двух вершин u, v и любого ребра e найдётся путь из u в v, проходящий через ребро e;

g) для любых двух вершин u, v и любого ребра e найдётся путь из u в v, не проходящий через ребро e;

h) для любых трёх вершин u, v, w найдётся путь из u в v, проходящий через w.

Решение первой задачи

$b \Leftrightarrow a$ Рассмотрим граф в котором для каждых двух вершин есть непересекающиесяся пути, и есть мосты. Рассмотри вершины по разные стороны ребра, являющегося мостом, из b следует, что существует два непересекающихся пути, но при этом любой путь обязан проходить через мост, так как вершины в разных компонентах связанности – противоречие
$b \Leftrightarrow c$ Если есть 2 не пересекающихся по рёбрам пути, то мы можем из вершины a в вершину b пойти по первому пути, а обратно по второму, так как они не пересекаются, то образуют тем самым цикл. Если же у нас есть цикл для любых 2 вершин, то мы можем разбить его этими 2-мя вершинами на 2 не пересекающихся по рёбрам пути
$c \Leftarrow d$ Докажем, что u, v образуют рёберно простой цикл – рассмотрим цикл, образуемый u и любым ребром, концом которого является v, очевидно, что раз ребро является частью цикла, то и его концы, отсюда и получаем искомый цикл. И так для всех вершин
$\Rightarrow d$ Рассмотрим ребро и вершину, построим пути от концов ребра в вершину: и дойдём до их первой точки пересечения (если таковой нет, то мы победили и нашли цикл), от этой точки строим цикл к изначальной, если он пересекается с началами путей, то изменяем их. В итоге получаем цикл, который не пересекается по рёбрам от концов ребра к точке между ней и начальной, а потом между точками. Если вершина совпадает с концом, то из существования цикла между точками следует d

№ 2

...

№ 3

Условие третьей задачи

Дан связный неориентированный граф вG Нужно ориентировать как можно больше его рёбер, так чтобы по-прежнему из каждой вершины был путь в каждую. Асимптотика: $O(n+m)$.

Решение третьей задачи

Найдём в графе все компоненты рёберной двусвязности. Получится, что граф разбился на таковые компоненты и мосты. Очевидно, что мы не можем ориентировать мосты, иначе из одной компоненты нельзя будет перейти в другую, а тогда граф перестанет быть связным. Тогда остаётся доказать, что любой двусвязный граф можно полностью ориентировать. Для этого во время прохода по нему dfs-ом будем ориентировать все рёбра, по которым продим. Оставшиеся же рёбра ориентируем в обратную сторону. Так как в рёберно-двусвязном графе нет мостов, то из любой вершины после их ориентировки можно добраться до корня (так как в графе без мостов есть 2 непересекающихся по рёбрам пути, а до вершины от корня есть конкретный путь, а это значит =, что второй будет ориентирован в обратную сторону), а из корня до любой другой вершины. В итоге будут ориентированы все рёбра кроме мостов, а так как поиск компонент будет занимать $O(n+m)$ времени, то это и будет итоговая асимптотика

№ 4

Условие четвёртой задачи

Пусть φ — формула в виде 2-КНФ с n переменными и m скобками. За $O(n \cdot (n+m))$ определите для каждой переменной верно ли, что её значение одинаково во всех выполняющих наборах φ. Иными словами, обязательно ли значение переменной фиксировано, если $\varphi = 1$?

Решение четвёртой задачи

Для каждой веришны просто будем дважды запускать $2SAT$ алгоритм, при этом считая, что в первый раз её значение – 1, а во второй – 0 . Если значение вершины единица, то мы считаем, что в неё не входит ни одно ребро, а из её отрицания не выходит не одно ребро (это следует из поведения импликации при подставлении 0 и 1 соответственно). Тогда мы запустим алгоритм $2n$ раз, а сама асимптотика $2SAT$ – $O(n+m)$, что и даёт нам в итоге $O(n \cdot (n+m))$

Решение четвёртого задания

№ 1

Условие первой задачи.

Во взвешенном неориентированном графе определите минимальный средний вес цикла. Средним весом цикла называем вес этого цикла, делённый на число рёбер в нём. Ответ определите с точностью до ε. Асимптотика: $O(nm \cdot \log\left(\frac{C}{\varepsilon}\right))$, где C — ограничение сверху на веса всех рёбер.

Решение первой задачи.

Очевидно, что средний вес цикла не может превышать вес ребра, кроме того, если мы уменьшим вес всех рёбер на какое-то число, то функция существования цикла отрицательного веса, а соответственно и цикла среднего отрицательного весса, будет монотонна относительно числа, на которое мы уменьшили вес всех рёбер. Тогда мы можем произвести бинпоиск по среднему весу цикла. То есть мы уменьшаем вес всех рёбер, а потом запускаем Форда-Беллмана, если цикл отрицательной длины нашёлся – есть цикл с меньшим средним весом, иначе нет. От этого и сдвигаем границу. В итоге Форд-Беллман работаем за $O(nm)$, бинпоиск за $O(\log\left(\frac{C}{\varepsilon}\right))$, так как мы пытаемся добиться определённой точности. Отсюда и асимптотика $O(nm \cdot \log\left(\frac{C}{\varepsilon}\right))$

№ 2а

Условие второй задачи, пункт **а**.

В неориентированном графе стоимостью пути назовём максимальный вес среди всех весов рёбер, входящих в него. По двум заданным вершинам найдите кратчайшее расстояние между ними за $O(m \log n)$

Решение второй задачи, пункт **а**.

Используем модифицированный алгоритм Дейкстры, формула пересчёта для вершины будет $dist[v] = min(dist[v], max(dist[u], weight[u][v]))$, где u — вершина, откуда мы просматриваем рёбра. Таким образом, асимптотика этого алгоритма будет совпадать с Дейкстрой, т.е. $O(m \log n)$

№ 2б

Условие второй задачи, пункт **б**.

В неориентированном графе стоимостью пути назовём максимальный вес среди всех весов рёбер, входящих в него. По двум заданным вершинам найдите кратчайшее расстояние между ними за $O(m+n)$

Решение второй задачи, пункт **б**.

Воспользуемся "бинпоиском": разделим все рёбра на две группы по весу, с помощью *quickselect*, что проделывается за $O(m)$, после чего проверим с помощью dfs, находятся ли заданные вершины в одной компоненте связности графа, сотсоящего из рёбер, нахоядщихся в меньшей весовой половине. Если это так, то будем работать с меньшей половиной, иначе сожмём меьшую половину так, что если две вершины были соединены ребром из меньшей половины, то они сжимаются в вершину, а из двух рёбер, ведущих из компоненты в вершину, мы выбираем меньшее. В обоих случаях количество рёбер уменьшается в два раза, по итогу мы ищем медиану за $O(m)$, сжимаем за $O(n+m)$, а по итогу из-за специфики бинпоиска (m уменьшается каждый раз в два раза), итоговая асимптотика и получится $O(n+m)$, т.к. $O(m)+O(\frac{m}{2})+\cdots=O(m)$

Решение пятого задания

№ 1

Условие первой задачи.

За $O(n+m)$ найдите произвольные $\frac{7}{8}n$ рёбер, которые можно дополнить до минимального остова.

Решение первой задачи.

Запустим итерацию алгоритма Борувки трижды (это займёт $O(n+m)$ времени из-за асимптотики самого алгоритма). Пусть в первый раз мы взяли в мин остов E_1 рёбер, а во второй и третий – E_2, E_3 соответственно. Заметим, что $E_1 > \lceil\frac{n}{2}\rceil$, при этом для второй итерации останется $n-E_1$ рёбер, т.е. $E_2 > \lceil\frac{n-E_1}{2}\rceil$, аналогично $E_3 > \lceil\frac{n-E_1-E_2}{2}$. Тогда заметим, что

$$E_1+E_2+E_3 \geq E_1+E_2+\lceil\frac{n-E_1-E_2}{2}\rceil \geq \lceil\frac{n+E_1+E_2}{2}\rceil \geq \lceil\frac{n+E_1+\lceil\frac{n-E_1}{2}\rceil}{2}\rceil \geq \lceil\frac{n+\lceil\frac{n+E_1}{2}\rceil}{2}\rceil \geq$$

$$\geq \lceil\frac{3n}{4}+\lceil\frac{E_1}{4}\rceil\rceil \geq \lceil\frac{3n}{4}+\lceil\frac{n}{8}\rceil\rceil \geq \frac{7n}{8}$$

№ 2

Условие второй задачи.

В двудольном графе вершины обеих долей пронумерованы последовательными целыми числами, начиная с единицы. Для каждой вершины i левой доли задан отрезок $[l_i, r_i]$. Это означает, что i соединена со всеми вершинами правой доли, номера которых попадают в отрезок $[l_i, r_i]$. Предложите алгоритм поиска максимального паросочетания в таком графе за $O(n \log n)$.

Решение второй задачи.

Заведём сканлайн с двумя событиями для каждой вершины из левой доли: открытием и закрытием (закрытие в $r_i + 1$) и расположим эти события в порядке возрастания координаты, что имеет асимптотику обычной сортировки.
Будем смотреть на все вершины из правой доли, для каждой из которых будем смотреть на события, где координата совпдает с номером вершины, также будет поддерживать сет из пар (r_i, i) для открытых вершин без пары из левой доли. Если отрезок открывается, то это потенциальная пара, которую мы доабвляем в сет, при закрытии и наличии – удаляем. После обработки всех событий берём минимальную пару из сета, если таковая имеется, а к нашей вершине ставим в пару вершину из этой пары.
Заметим, что мы всегда действуем оптимально и берём пару, максимищирующую паросочетание, так как ответ не ухудшится, возьми мы пару с границей дальше, а также не имеет разницы, возьми мы пару с такой же границей. Значит наш алгоритм даёт макс пар соч.
При этом событий $\approx n$, а значит, включая сортировку и учитывая сет, наш алгоритм работает за требуемые $O(n \log n)$

№ 3

Условие третьей задачи.

Докажите, что если G — ациклический транзитивный орграф, то наименьшее количество независимых множеств, на которые можно разбить все вершины G, равно размеру самого длинного пути в G.

Решение третьей задачи.

Пусть размер самого длинного пути – L

Для начала докажем, что нельзя разбить граф на меньшее число независимых множеств:
Рассмотрим самый длинный путь, очевидно, что каждая его вершина должна лежать в отдельном

независимом множестве, ведь между любыми двумя вершинами в этом пути есть ребро в одну из сторон в силу транзитивности графа (для этого мы просто из вершины, что раньше в пути берём ближайшую в пути, потом следующую, потом ищем "транзитивное"ребро, а потом повторяем процесс, в итоге найдём ребро от одной к другой)
Теперь научимся разбивать граф на нужное количество множеств:
Произведём топологическую сортировку графа (он ацикличен), после чего у нас рёбра будут только и вершин с большим номером в меньшие. Пойдём по вершинам по возрастанию. Для каждой вершины мы можем определить номер независимого множества($\leq L$) так, чтобы он не совпадал с номерами вершин, в которые можно из неё прийти (так как их мы уже обработали и их $< L$). Значит мы сможем сделать так для каждой веришны, т.е. мы можем разбить граф на L независимых множеств, ч.т.д.

№ 4a

Условие четвёртой задачи, пункт **a**.

Игра в города на неориентированном графе G определяется следующим образом. Изначально фишка расположена в одной из вершин (назовём её стартовой). Игроки ходят по очереди, на каждом ходу нужно сдвинуть фишку вдоль любого исходящего ребра в вершину, в которой фишка ещё ни разу не была. Проигрывает тот, кто не может сделать ход.

Докажите, что первый выигрывает, если и только если стартовая вершина лежит во всех максимальных паросочетаниях.

Решение четвёртой задачи, пункт **a**.

Для начала докажем, что если стартовая вершина лежит не во всех макс паросочетаниях, то первый выграть не может:
Для этого просто нужно, чтобы второй игрок ходил по рёбрам максимального паросочетания, в которое не входит стартовая вершина. Тогда первый игрок своим первым ходом обязательно придёт в вершину ребра из паросочетания из максимального увеличивающего пути не по ребру из паросочетания, а значит этот путь закончится на ребре из него, что будет ходом второго игрока, а первому будет некуда ходить. (Закончится на этом ребре, потому что иначе это не будет максимальным паросочетанием)

Теперь докажем в обратную сторону:
Тут всё в принципе аналогично: первый просто ходит по рёбрам из максимального паросочетания, а так как он начинает в вершине, нахоядщейся во всех макс пар соч, то второй игрок просто не может не ходить по рёбрам, не принадлежащим максимальному удлиняющему пути, (если же первый вдруг

в веришне, из которой нет рёбер с паросочетанием, то мы можем перекрасить рёбра (в макс пар соч/не в макс пар соч) для паросочетания, по которому идём и получить больше паросочетаний и противоречие условию (в плане, что стартовая вершина уже не во всех макс пар соч)

№ 5

Условие пятой задачи.

Дан ориентированный граф. Найдите в нём (или определите, что так сделать нельзя) некоторое множество циклов, которые попарно не пересекаются, но покрывают всё множество вершин. Цикл из одной вершины не считается циклом (а из двух — считается). Асимптотика: $O(nm)$.

Решение пятой задачи.

Раздвоим каждую вершину. Т.е. из вершины v получаем l_v и r_v. Создадим исток, из которого будут рёбра во все вершины l и сток, из которого будут рёбра во все вершины r. У всех этих рёбер будет пропускная способность 1. Кроме того создадим рёбра с пропускной способностью 1: $l_v \to r_u \Leftrightarrow (u, v) \in E$.
Если искомое множество циклов существует, то все рёбра из истока насыщены, так как в каждом искомом цикле в вершину вхоидт одно ребро и выходит, при этом задействованы все вершины.
Из-за того, что из каждой веришны должно выходить одно ребро и в каждую должно входить одно ребро, то мы можем воспользоватсья алгоритмом Куна для поиска максимального паросочетания, после чего, нужно проверить, является ли макс пар соч совершенным паросочетанием. В итоге из-за асимптотики алгоритма Куна асимптотика $O(nm)$

№ 7

Условие седьмой задачи.

В прямоугольной таблице $n \times m$ некоторые клетки заблокированы. Определите, можно ли разбить оставшиеся клетки на циклические маршруты длины хотя бы 3 (соседними клетками считаются разделяющие сторону)? Все неудалённые клетки должны участвовать ровно в одном маршруте ровно один раз. Начало каждого маршрута должно совпадать с его концом. Выберите оптимальный алгоритм и оцените его асимптотику как можно точнее.

Решение седьмой задачи.

Пусть таблица имеет шахматную раскраску, тогда чёрных и белых незаблокированных клеток поровну, иначе точно нельзя разбить, так как в каждом цикле будет поровну клеток ведь они чередуются.

Пусть все наши циклы начинаются из чёрных клеток, тогда разобьём граф на две доли, в одной все чёрные, а в другой все белые. После создадим исток с рёбрами во все вершины первой доли и сток с рёбрами из всех вершин второй доли, все эти рёбра пусть имеют пропускную способность 2. Также для каждых соседствующих клеток/вершин создадим ребро из чёрной в белую с пропускной способностью 1. Тогда условием успеха разбиения будет то, что из каждой чёрной вершины выходит два насыщенных ребра и аналогично в каждую белую два насыщенных ребра входят. Так мы обеспечим циклы длины хотя бы 4.

В итоге задача свелась к поиску максимального потока. Воспользуемся алгоритмом Диница, сумманый потенциал всех вершин $\geq mn$, т.е. $O(mn)$, так как у нас не более $\frac{mn}{2}$ вершин каждого цвета. Значит будет $O(\sqrt{mn})$ итераций алгоритма Диница по оценке Карзанова. Асимптотика одной итерации – $O(mn)$ (так как сумма по dfs-ам). Получается итоговая асимптотика $O(mn\sqrt{mn})$

Решение десятого задания

№ 1

Условие первой задачи

По данной префикс-функции найдите лексикографически минимальную строку над алфавитом $1, 2, \ldots$, имеющую в точности такую префикс-функцию.

Решение первой задачи

Будем строить нашу строку, помня о принципах работы алгоритма построения префикс-функции.

Если $p_i \neq 0$, то нам символ определён однозначно, ведь нам известен наибольший супрефикс, тогда мы просто ставим s_{p_i-1}. Чтобы суффикс совпадал с префиксом.

Если же $p_i = 0$, то при подсчёте префикс-функции мы перебрали все элементы $\{s_{p_i-1}, s_{p_{p_i-1}-1}, \ldots, s_0\}$ и среди них не нашли подходящий. То есть при построении строки нам нужно выбрать минимальное число, которое не будет входить в список значений.

Первым же символом строки будет 1.

В итоге наш алгоритм будет схож по структуре с алгоритмом построения префикс-функции, а значит будет иметь асимптотику $O(n)$

№ 2

Условие второй задачи

Докажите, что в строке длины n число различных подпалиндромов не превосходит n.

Решение второй задачи

Докажем по индукции:

1. **База:** $n = 1$: 1 подпалиндром — вся строка
2. **Предположение:** при добавлении нового символа появляется не более 1 нового палиндрома
3. **Переход:** Понятно, что все новые палиндромы должны заканчиваться в новом символе. Пусть палиндромы, включающие нвоый символ существуют, значит существует такой палиндром наибольшей длины, тогда рассмотрим все остальные: для каждого такого палиндрома будет существовать симметричный ему, начинающийся в начале длиннейшего палиндрома, так как и то, и то — палиндромы. Значит этот палиндром уже встречался в строке $\Rightarrow$ единственным палиндромом, не встречавшимся в строке, может быть только наидлиннейший палиндром, а он только один ■

№ 3

Условие третьей задачи

Дан набор слов $s_1, \ldots, s_n$. Поступает q запросов, в i-м запросе нужно сообщить, во сколько строк из набора $s_1, \ldots, s_n$ строка t_i входит в качестве супрефикса (является и префиксом, и суффиксом). Отвечать на запросы нужно онлайн. Асимптотика: $O\left(\sum |s_j| + \sum |t_i|\right)$. Решения, ошибающиеся с положительной вероятностью, получат 1 балл.

№ 4

Условие четвёртой задачи

С помощью какой-нибудь строковой суффиксной структуры (суффиксный массив, суффиксное дерево или суффиксный автомат) найдите для каждого i в строке s максимальный радиус чётного и нечётного палиндрома с центром в i (позиция i либо совпадает с одним из индексов s, либо находится между двумя соседними индексами). Асимптотика: $O(n)$, где $n = |s|$. Решения с асимптотикой $O(n \log n)$ получат 1 балл.

Решение четвёртой задачи

Заметим, что мы можем свести нашу задачу к работе над поиском только максимального радиуса нечётных палиндромов, добавив разделители между всеми буквами (нечётные палиндромы таковыми и останутся, а чётные станут нечётными с центром в разделителе).

Разделители же мы добавим не в нашу строку, а в строку, полученную приписывание к нашей строке обратной к ней. Длина такой строки не превосходит $O(n)$.

Теперь построим суффиксное дерево для полученной строки. Тогда заметим, что задача поиска любого палиндрома сводится к задаче поиска наибольшего общего префикса, у суффиксов начинающихся в i и в $len(S') - i - 1$, где S' — строка после приписывания обратной и добавления разделителей (так как суффикс второго — левая часть первого, соответственно палиндрома). Тогда наша задача просто сводится к нахождению LCA в суффиксном дереве, что делается за $O(1)$ с $O(n)$ предподсчётом, построение суффиксного дерева также занимает $O(n)$

Итоговая асимптотика алгоритма — $O(n)$

№ 5

Условие пятой задачи

Дан словарь из n слов: $s_1, \ldots, s_n$. Для каждого i сообщите, входит ли s_i как подстрока в одно из других словарных слов. Асимптотика: $O\left(\sum\limits_{i=1}^{n} |s_i|\right)$.

Решение пятой задачи

Воспользуемся алгоритмом Ахо-Карасик на тексте $s_1 \# s_2 \# \ldots \# s_n$. Так как Ахо-Карасик находит все вхождения, то нам надо лишь проверить, что их хотя бы 2 (тогда строка входит сама в себя и в какую-то другую). Так как мы разделили всё решётками, то не найдётся таких вхождений, где строка входит в несколько по частям. При этом Ахо-Карасик верен и работает за суммарную длину строк, т.е. удовлетворяет требованиям асимптотики.

Решение одиннадцатого задания

№ 1

Условие первой задачи

За $O(n)$ найдите количество делителей у всех чисел среди $1, 2, \ldots, n$.

Решение первой задачи

Насчитаем решетом Эратосфена, работающим за $O(n)$ минимальный простой делитель. Для каждого числа будем хранить ответ, т.е. количество делителей, а также степень вхождения минимального простого делителя. При этом для единицы всё очевидно: `num_of_divs[1] = 1; deg[1] = 0`. Осталось научиться пересчитывать значения этих двух массивов для следующих чисел (минимальный простой делитель обозначим за `mpd[i]`). Если мпд входит хотя бы дважды в рассматрвиаемое число, то формула будет иметь вид: `num_of_divs[i]=num_of_divs[i/mpd[i]]*(deg[i/mpd[i]] + 2)/(deg[i/mpd[i]] + 1)`, `deg[i]=\deg[i/mpd[i]] + 1` — этот случай мы будем рассматривать если минимальный делитель частного совпадает с минимальным делителем исходного, в ином же случае формулы примут вид: `num_of_divs[i]=num_of_divs[i/mpd[i]]*2; deg[i] = 1`. В итоге просто двух проходов мы будем иметь требуемую асимптотику.

№ 2

Условие второй задачи

В этой задаче можно пользоваться фактом, что $\sum\limits_{\substack{p \le n \\ p \text{ простое}}} \frac{1}{p} = O(\log\log n)$. За $O(\log\log n)$ найдите все простые числа, лежащие в отрезке $[n^2, n^2 + n]$.

Решение второй задачи

Если число непростое, т.е. составное, то у него точно есть делитель $< 2n$, если это не так, то число имеет 2 делителя, не меньших $2n$, но тогда само число $\ge 4n^2$, но при любых натуральных n такое число не лежит в нужном нам отрезке. То есть у любого непростого числа есть делитель, меньший $2n$.

Тогда за $O(n)$ мы можем найти все простые на отрезке $[2; 2n]$. Далее найдём все числа, делящиеся на найденные простые в требуемом отрезке и пометим их непростыми, соответственно мы нашли все простые. Осталось оценить асимптотику.

Для каждого простого в отрезке будет $\le \frac{n}{p}$ чисел, поэтмоу мы сделаем $\sum\limits_{\substack{p \le n \\ p \text{ простое}}} \frac{n}{p} = O(n \log\log n)$■

№ 4

Условие четвёртой задачи

Найдите число решений уравнения $x^n + y^n = z^n$ в кольце вычетов $\mathbb{Z}_m$. Асимптотика: $O(N \log N)$, где $N = \max\{n, m\}$.

Решение четвёртой задачи

Переберём все вычеты, возведём их всё в том же колце в n-ую степень. Теперь составим многчлен, где степень x будет равна одному из получившихся вычетов, а коэффициент будет количеству раз, сколько встречается этот вычет, т.е. если a_i — i-ый вычет и он встречается k_i раз, то мы получим многочлен: $F(x) = \sum_i k_i * x^{a_i}$. Далее возведём его в квадрат и рассмотрим коэффициент при какой-нибудь степени.

$coef(x^y) = \sum_{i+j=y} k_i * k_j$, т.е. число способов получить вычет y из двух вычетов. Тогда мы можем за $O(m \log m)$ насчитать коэффициенты нашего многочлена в квадрате, тогда будет перебирать z^n и количество решений будет $p_{z^n} + p_{m+z^n}$, при этом p — коэффициенты квадрата многочлена, тогда всего решений: $\sum_{i=1}^{k} a_k \cdot (p_{z^n} + p_{m+z^n})$. Решение будет работать за $O(m \log m + n \log m)$, что очевидно является требуемой асимптотикой.

№ 5

Условие пятой задачи

Пусть n — чётно. Номер трамвайного билета — это строка из n допустимых цифр (допустимыми являются некоторые десятичные цифры $d_1, \ldots, d_k$, то есть не обязательно все цифры от 0 до 9). За $O(n \log n)$ найдите число счастливых билетов, то есть таких билетов, в которых сумма первых $\frac{n}{2}$ цифр равна сумме остальных.

Решение пятой задачи

Так как для суммы мы опять будем использовать сумму степеней, то рассмотрим многочлен, где степенями будут являться допустимые цифры, тогда после возведения в степень $\frac{n}{2}$ мы получим при i-ой степени количество чисел с суммой цифр i, состоящие только из допустимых цифр. Осталось научиться возводить наш многочлен в нужную степень ($\frac{n}{2}$).

Воспользуемся комбинацией fft и $fastpow$, степень многочлена в k-ой степени будет меньше $9k$, т.к. цифр не более 10, а значит, у нас получится рекурсивная формула, которая очень легко раскрывается: $T(n) = T(\frac{n}{2}) + O(9n \log 9n) = T(\frac{n}{2}) + O(n \log n)$. Теперь просто подставим рекурсивно $T(n)$:

$T(n) = O(n \log n) + O(\frac{n}{2} \log \frac{n}{2}) + \cdots = O((1 + \cdots + \frac{n}{2} + n) * \log n) = O(n \log n)$. То есть многочлен мы насчитываем за требуемую асимптотику, остаётся лишь сложить квадраты коэффициентов, потому что мы можем именно столькими способами выбрать левую и правые части билета.

В итоге получили ответ с требуемой асимптотикой■

Решение двенадцатого задания

№ 1

Условие первой задачи

На плоскости даны три непересекающихся круга. Найдите точку, из которой все три круга видны под одинаковыми углами. Считайте, что круги не загораживают друг друга.

Решение первой задачи

Круг виден под углом $2\arcsin\frac{r}{l}$, где r — радиус окружности, l — расстояние до центра, рассмотрим задачу, но для 2-х кругов, т.е.

$$2\arcsin\frac{r_1}{l_1} = 2\arcsin\frac{r_2}{l_2}$$

$$\frac{l_2}{l_1} = \frac{r_1}{r_2} = const$$

Точки, удовлетворяющие уравнению будут образовывать окружность Аполлония для центров окружностей (соответственно подойдёт любая точка на ней).

Для решения оригинальной задачи нам нужно просто посчитать эту окружность для других двух кругов, тогда решением будет одна точек пересечения, если такой точки нет, то и ответа на задачу нет, так как всякая точка, из которой хотя бы 2 круга видны под одинаковым углом, находится на окружности Аполлония■

№ 2

Условие второй задачи

Река заключена между двумя прямыми-берегами. В реке есть n круглых островков, движение по которым запрещено. Определите максимальный радиус корабля, который может проплыть по этой реке, избежав столкновения с островами, за $O(n^2\log(1/\varepsilon))$, где ε — необходимая точность ответа.

Решение второй задачи

Воспользуемся двоичным поиском по радиусу корабля: минимальный радиус — 0, максимальный же — расстояние между берегами. Отсюда получим $\log(1/\varepsilon)$, так как этого расстояния между границами бинпоиска мы будем требовать для ответа. Проверять же то, подходит ли наш радиус будем следующим образом: представим систему островов с берегами в качестве графа, вершинами будут являться все острова и оба берега, а рёбра будем проводить, если расстояние между вершинами

меньше проверяемого радиуса, ребро от вершины к берегу выбираем наименьшее. Такой граф строится перебором всех вершин, т.е. за $O(n^2)$. После этого остаётся лишь проверить, что кораблю негде пройти, для этого надо проверить, находятся ли оба берега в одной компоненте связанности, для этого достаточно запустить DFS из одного из берегов, что тоже будет удовлетворять $O(n^2)$.

Таким образом найдём ответ. Итоговая ассимптотика $O(n^2 \log(1/\varepsilon)$

№ 3

Условие третьей задачи

На плоскости расположено n точек. За $O(n \log n)$ найдите треугольник минимального периметра с вершинами трёх различных точках этого множества.

Решение третьей задачи

Отсортируем точки сначала по абциссам, а потом по ординатам, после чего запустим наш алгоритм:

1. Разделим область, над которой мы работаем напополам, после чего рекурсивно запустимся в обеих (считаем, что если треугольников нет, то минимальный периметр в области $+\infty$)

2. После нахождения треугольника минимального периметра в обеих частях рассмотрим точки, входящие по абциссам в область $[mid - P/2; mid + P/2]$, где P — минимальный периметр (в случае с бесконечностью берём до границы). Берём столько, потому что иначе точки, будучи и из левой, и из правой части, образуют сторону длиной $P/2$, а значит по правилу треугольника периметр этого самого треугольника будет больше P

3. Теперь осталось доказать, что если мы будем смотреть на точки последовательно в зависимости от их ординаты, то, смотря на точки, отстающие от рассматриваемой не более чем на P, мы не выбьемся из нужной асимптотики. (То есть хотим доказать, что это константа)

4. Рассмотрим рассмотрение какой-то точки, для неё мы рассмотрим квадрат $P \times P$, после чего разделим каждую сторону мысленно напополам, а потом ещё раз напополам каждую половну. В итоге получим 16 частей. Теперь докажем, что в этом квадрате не может находиться более 32 точки.

5. Пусть это не так, тогда в каком-то из 16 маленьких квадратиков есть 3 точки, но тогда с одной из сторон нашей прямой из основного алогритма есть треугольник с периметром $\leq \frac{P}{4} * 3 < P$ — противоречит предположению, что периметр минимален. Значит точек не более 32, т.е. константа, а значит по мастер теореме это будет выполняться за $O(n \log n)$

6. В итоге мы рассматриваем точку и перебираем константу точек для проверки треугольника и так далее по всей прямой. После чего корректируем или нет наш минимальный периметр.

Итоговая асимптотика соответствует требуемой.

№ 4

Условие четвёртой задачи

На плоскости расположено n точек. За $O(n \log n)$ найдите прямоугольник минимальной площади (его стороны не обязаны быть параллельны осям координат), который содержит в себе все n точек

Решение четвёртой задачи

Докажем, что, во-первых, каждая сторона прямоугольника доожна пересекать одну из сторон выпуклой оболочки, которую мы заблаговременно построили за асимптотику, удовлетворяющую данной. Если это не так, то просто будем сдвигать сторону прямоугольника в сторону выпуклой оболочки, чем уменьшим площадь прямоугольника. Теперь докажем, что прямоугольник содержит одну из сторон выпуклой оболочки. Если это не так, то на каждой стороне есть вершина согласно предыдущему пункту. Тогда образуем с помощью них диагонали, с помощью которых посчитаем площадь прямоугольника. Если длины диагоналей n и m, α — угол между "диагональю"и стороной, β — угол между "диагоналями то $S = n \cdot m \cdot \cos\alpha \cdot \sin(\alpha + \beta)$. Взяв производную получим, что она не может быть равна 0, поэтому одна из сторон прямоугольника содержит сторону выпуклой оболочки.

После чего надо перебрать стороны и для каждой найи такие точки, через которые провести оставшиеся прямые, если, например, мы имеем вертикальную прямую, то для горизонатльных сторон нам нужны самая верхняя и самая нижняя точки. Такие точки можно найти бинпоиском по углу в выпуклой оболке. Зная количество сторон и асимптотику бинпоиска и получаем нужную асимптотику.

№ 5

Условие пятой задачи

На плоскости даны n точек. Далее поступает q запросов, каждый из которых — очередная прямая. Для каждого запроса определите, является ли прямая запроса разделяющей, то есть найдутся ли две точки исходного множества, лежащие по разные стороны от прямой. Асимптотика: $O((n + q) \log n)$.

Решение пятой задачи

Воспользуемся эдаким бинпоиском по выпуклой оболочке, котору сначала построим за $O(n \log n)$, кроме того каждую сторону мы будем считать за направленный по часовой стрелке отрезок. Алгоритм обработки запроса будет следующий:

1. Выбираем одну из сторон оболочки (она будет считаться и левой и правой границей, но двигать будем в разные стороны, работать будем над стороной, являющей серединой между двумя границами)

2. Найдём точку пересечения прямой и стороны (если таковой не имеется, т.е. сторона параллельна прямой, то присваиваем любой границе номер этой стороны)

3. Если точка пересечения лежит на нашей стороне, то мы сразу выдаём положительный ответ на запрос и переходим к следующему

4. В ином же случае мы смотрим на коллинеарность/неколлинеарность вектора от "начала"направленного отрезка до точки пересечения и вектора, направляющим отрезком которого является сторона. В случае коллинеарности мы смещаем правую границу в нашу сторону, а инчае — левую.

5. таким образом мы осуществим бинпоиск и найдём либо нужную сторону, либо то, что точек из условия для данной прямой не найдётся

Итоговая асимптотика будет $O(n \log n)$ за построение оболчки и $O(q \log n)$ за все запросы, а значит: $O(n+q) \log n$

Решение тринадцатого задания

№ 1

Условие первой задачи

Из точки $(0, 0)$ можно бросить дротик под углом к горизонту, траектория его движения описывается параболой. В воздухе висит n мишеней, i-я из которых представляет собой отрезок $(x_i, l_i) \div (x_i, r_i)$. Найдите максимальное k такое, что брошенный дротик может поразить первые k целей. Асимптотика: $O(n \log n)$.

Решение первой задачи

Заметим, что так как мы кидаем дротик из точки $(0; 0)$, то парабола, описывающая его движение будет иметь вид: $y = ax^2 + bx$, так как $y(0) = 0$.

Тогда, чтобы дротик поразил i-ую цель, должно выполнятсья условие: $l_i < ax_i^2 + bx_i < r_i$. Тогда рассмотрим координатную плоскость с осями a и b, там каждое условие превращается в две полуплоскости (по одной на каждое из неравенств).

Теперь наша задача свелась к тому, чтобы найти максимальное k, такое, что пересечение полуплоскостей с 1-ой по k-ую будет непусто. Нам будет досаточно выбрать любую точку из этого пересечения, чтобы получить нужную параболу (из точки получим коэффициенты).

Воспользуемся бинпоиском по k, ведь очевидно, что функция "пусто ли пересечение первых $2i$ полуплоскостей?"будет монотонна (добавляя полуплоскости мы не увеличиваем площадь пересечения; $2i$, потому что каждое условие даёт 2 полуплоскости).

Мы можем проверять непустоту за $O(n)$ в среднем, тогда бинпоиск даст нам $T(n)$, такое, что $T(n) = T(\frac{n}{2}) + O(n) \Leftrightarrow T(n) = O(n \log n)$■

№ 2

Условие второй задачи

На плоскости даны два выпуклых многоугольника. Найдите их пересечение за $O(n + m)$, где n и m — число вершин в многоугольниках.

Решение второй задачи

Воспользуемся следующим алгоритмом, а потом докажем корректность:

Выберем по ребру на каждом многоугольнкие (ребро на одном и ближайшее к нему на другом, это делается за линию) и запомним их, алгоритм остановится, когда оба ребра "дойдут"или "пройдут"свои начала. Теперь будет двигать одно из рёбер, в зависимости от случая, тогда по прошествии круга мы и будем иметь требуемую асимптотику. Случаи будут в зависимости от того, как стороны смотрят друг на друга: интерпретируем сторону как направленный отрезок, при этом мы обходим многоугольники против часовой стрелки, поэтому будем брать направление из обхода. Тогда стороны смотрят друг на друга, если их продолжения туда, куда они направлены пересекаются (или они параллельны), не смотрят друг на друга, если их продолжения в обратную сторону пересекаются и одна смотрим на другую, если продолжение одной по направлению пересекает другую, либо её продолжение против направления. Рассмотрим действия во всех этих случаях:

- Если стороны смотрят или не смотрят друг на друга, то мы двигаем (под движением мы понимаем переход к следующей по обходу стороне) ту, что находится в правой полуплоскости от другой.
- Если же только одна сторона смотрит на другую, то мы двигаем именно смотрящую

При этом на каждом шаге мы проверяем стороны на пересечение и если такое происходит, то добавляем точку пересечения в оба прямоугольника и смещаем рассматриваемые стороны во вторые части поделенных точкой сторон, после чего продолжаем алгоритм (это позволит избежать пропуска пересечений).

Асимптотика работы очевидна и соответствует требуемой, так как проверка на пересечение сторон делается за константу, как и нахождение стороны, которую нужно двинуть.

Докажем корректность алгоритма. Очевидно, что мы не добавим в наш многоугольник пересечения ничего лишнего (кроме вершин пересечения мы добавляем туда ещё и левые рёбра пока идём). Осталось доказать, что мы точно найдём все пересечения. Рассмотрим любую сторону, из алгоритма следует, что если на ней есть точка пересечения, то она не будет двигаться, просто потому что не будет правым при обходе против часовой стрелки, т.е. будет двигаться другое ребро, пока не достигнет пересечения. Тогда мы действительно найдём все пересечения, а также добавим все нужные рёбра и получим искомый многоугольник за искомое время■

№ 3

Условие третьей задачи

На плоскости задан многоугольник (необязательно выпуклый) с n вершинами. В его вершинах ставят излучатели добра, каждый покрывает круг одного и того же радиуса r. Определите минимальное r, при котором весь многоугольник (со внутренностью) будет добрым. Асимптотика: $O(n^2 \log n)$.

Решение третьей задачи

Сначала опишем алгоритм, а потом докажем его корректность и асимптотику:

Построим диаграмму Вороного для точек нашего многоугольника за $O(n \log n)$, после чего для каждой ячейки в диаграмме рассмотрим её вершины, лежащие внутри исходного многоугольника, а также точки, в которых стороны исходного многоугольника пересекают стороны ячейки. Найдём максимум расстояний от сайта нашей ячейки до всех этих точек, тогда искомым радиусом будет являться максимум по всем ячейкам.

Оценим время работы нашего алгоритма: всего мы имеем n ячеек, нахождение максимума по расстоянием до вершин ячейки занимает $O(n)$ времени, так как колиечтсво вершин в диаграмме линейно, а исключить вершины вне многоугольника мы можем предподсчётом после построения диаграммы за $O(n^2)$. Проверка пересечения ячейки стороной исходного многоугольника происходит за $O(\log n)$, так как мы просто пользуемся бинпоиском по сторонам ячейки (сторона ячейки либо пересекает, либо выше, либо ниже, при этом имеется монотонность). Точка же пересечения находится за $O(1)$, тогда для всех сторон многоугольника мы найдём точки пересечения за $O(n \log n)$. Максимум ищется за $O(n)$ для ячейки, так как количество вершин ячейки линейно, как и кол-во точек пересечения, так как каждая сторона исходного многоугольника даёт не больше двух. Тогда время для одной ячейки — $O(n \log n)$, а для всех ячеек — $O(n^2 \log n)$.

Докажем, что мы нашли нужный радиус. Для начала докажем, что для покрытия ячейки (точнее её пересечения с исходным многоугольником) достаточно круга с центром в сайте этой ячейки: если это

не так и нужен ещё круг, то рассмотрим место, которое не покрывается, тогда для него расстояние от него до другого сайта будет меньше, чем расстояние до первого сйата, но тогда эта часть должна принадлежать другой ячейке□. А минимальный радиус для ячейки достигается либо в вершине, если пересечение ячейки с исх. многоугольником выпукло, либо может быть на пересечении со стороной исх. многоугольника, так как расстояние до любой стороны будет меньше максимума из расстояний до этих точек. В итоге найденный нашим алгоритмом радиус будет и правильным, и минимальным (в силу того, что иначе найдётся ячейка с непокрытой частью в силу предыдущих двух утверждений)■

Additional information

Плейлист лекций

И немного еще про тех.

Сборник всякого про тех, там в конце есть полезные ссылки

Тех статов, оттуда взяты примеры этого документа, а еще там много всего есть