# «Решения теоретических ДЗ»
## ФПМИ МФТИ

Автор решения

Весна 2022

# Содержание

# 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 $\leq 18$ 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 в$G$ 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 $\varphi$ — 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 $\varphi$. 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 $\varepsilon$. 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, суммарный 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 $O(mn\sqrt{mn})$

# 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