2 курс ПМИ, осень 2025/26
Практика по алгоритмам #1

Паросочетания, IS, VC

9 сентября

Собрано 1 сентября 2025 г. в 16:41

Содержание

Паросочетания, IS, VC

1. **Покрытие доминошками**

   Покрыть клетчатое поле $n \times m$ с дырками доминошками. Каждая клетка должна быть покрыта ровно один раз. $\mathcal{O}(\text{poly}(n, m))$.

2. **Двудольная клика**

   Найти максимальную двудольную клику (полный двудольный подграф) в двудольном графе за (a) $\mathcal{O}(V^3)$, (b) `(*)` $\mathcal{O}(VE)$.

3. **Максимизация** $|A| - |N(A)|$

   Выбрать в двудольном графе множество A из левой доли, что $|A| - |N(A)| \to \max$.
   $N(A)$ – множество соседей A в правой доле. $\mathcal{O}(VE)$.

4. **Чётность числа паросочетаний**

   Верно ли, что в любом двудольном графе чётное число совершенных паросочетаний?
   Если верно, то докажите, иначе приведите контрпример.

5. **Лемма Холла и Кун**

   Лемма Холла: $\exists$ или совершенное паросочетание, или множество A: $|A| > |N(A)|$.
   Придумайте алгоритм, который находит или совершенное паросочетание, или такое A.

6. **Удаление графа**

   Дан орграф. За один ход можно удалить все входящие или все исходящие ребра одной вершины (но не одновременно). Удалить все рёбра графа за min число ходов.

7. **Покраска матрицы отрезками**

   Дана изначально чёрная матрица. Дан чёрно-белый вид, к которому нужно привести эту матрицу за *минимальное число ходов*. За один ход можно «сделать мазок кисточкой» — в матрице взять вертикальный или горизонтальный отрезок $1 \times k$ произвольной длины k и целиком покрасить в белый цвет. Мазки могут перекрываться.

8. **Можно снимать часть пометок**

   Рассмотрим алгоритм Куна, который при успешном нахождении дополняющего пути снимает пометки со всех вершин. Докажите, что можно снимать пометки только с вершин, первый раз посещённых в последнем запуске `dfs`.

9. **Жадная инициализация не нужна**

   Докажите, что первый проход «Куна» с оптимизацией «вообще не обнуляем пометки» найдет паросочетание не меньше $|M|/2 \Rightarrow$ жадная инициализация не нужна.

10. **Единственность максимального паросочетания**

    По данному максимальному паросочетанию в **двудольном** графе проверить, является ли оно единственным. $\mathcal{O}(VE)$? а $\mathcal{O}(E)$?
    `(*)` А если граф недвудольный, за любой полином?

11. **Разбиение DAG-а**

    Дан ацикличный орграф (DAG). Разбить все его вершины на минимальное число путей. Каждая вершина должна лежать ровно в одном пути.

12. **Ездят такси, но нам нечем платить** (*Виктор Цой*)

    По городу ездят такси. Город задаётся взвешенным графом, где вес ребра – время проезда по нему. Заказ такси задаётся начальной вершиной, конечной вершиной и временем отправления из начальной вершины. Выполнить все заказы, используя минимальное число машин.

13. **Разбиение на плавные подпоследовательности**

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

14. **Покрытие множества паросочетанием**

    Придумайте алгоритм, который строит паросочетание, покрывающее множество A вершин первой доли за время $\mathcal{O}(VE)$, докажите его корректность.
    Как найти максимальное среди всех паросочетаний, покрывающих A?

15. **Рёберное покрытие**

    Реберное покрытие – множество рёбер, среди концов которых есть все вершины.
    Найти минимальное рёберное покрытие двудольного графа. А для недвудольного?

16. **Эйлеровость и паросочетания**

    Научитесь красить рёбра d-регулярного двудольного графа в d цветов за $\mathcal{O}(\texttt{Matching} \cdot \log d)$.
    В этой задаче предполагается, что `Matching` уже написан и дан вам, как чёрный ящик.
    (a) $time(\texttt{Matching}) = \mathcal{O}(VE)$ или $\mathcal{O}(E\sqrt{V})$, (*) (b) для $time(\texttt{Matching}) = \mathcal{O}(V \log V)$.

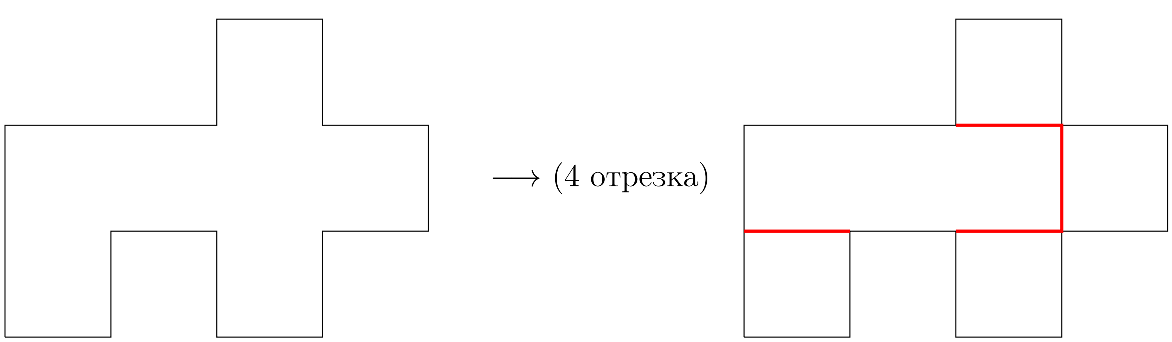
17. **(*) Birkhoff–von Neumann decomposition**

    Дана матрица из неотрицательных вещественных чисел. Сумма элементов в каждой строке и каждом столбце равна 1. Разложить заданную матрицу на сумму перестановочных матриц с коэффициентами. Перестановочная матрица – перестановка строк единичной матрицы (т.е. в каждой строке и каждом столбце ровно одна единица).

    (a) $\mathcal{O}(n^5)$, (b) $\mathcal{O}(n^4)$.

18. **(*) Прямоугольные многоугольники**

    Дан простой многоугольник со сторонами параллельными осям координат. Все углы прямые. Разрезать его на минимальное число прямоугольников. $N \leqslant 100$.



Разбор задач практики

1. **Покрытие доминошками**

   Невырезанная клетка – вершина графа, граница между соседними клетками – ребро.
   Покрытие доминошками – в точности паросочетание.
   Осталось заметить, что граф двудольный (рассмотрим шахматную раскраску).

2. **Двудольная клика**

   Решение: инвертируем ребра и находим в новом графе IS.

   То есть, проведём между долями все ребра, которых не было, и удалим все, которые были.
   Любая клика в исходном графе G – независимое множество в новом G'.
   Время работы выйдет $\mathcal{O}(V(V^2 - E)) = \mathcal{O}(V^3)$.

   Решение за $\mathcal{O}(VE)$. Вкратце: можно перебирать только «отсутствующие рёбра».

   Пусть в орграфе присутствуют все рёбра кроме k. Научимся делать bfs за $\mathcal{O}(V + k)$.
   Поддерживаем `vector<int> notUsed` непосещённых вершин. Пусть `notG[`v`]` – отсутствующие у v рёбра. Шаг bfs: добавить все вершины, которые в `notUsed`, но не в `notG[`v`]` в очередь.

```
    cc++, newSize = 0; // cc++ = обнуление skip
    for (int x : notG[v]) skip[x] = cc;
    for (int x : notUsed)
        if (skip[x] == cc) addQueue(x)
        else notUsed[newSize++] = x;
    notUsed.resize(newSize); // вершины, которые остались непосещёнными
```

   Время работы $\mathcal{O}(q + |$`notG[`v`]`$|)$, где q – число вершин, добавленных в очередь.
   Каждая вершина добавится 1 раз $\Rightarrow$ суммарное время $\mathcal{O}(V + k)$.

3. **Максимизация** $|A| - |N(A)|$

   Пусть R – правая доля. Вместо $|A| - |N(A)| = |A| + |R \setminus N(A)| - |R|$. Константа $-|R|$ не влияет на максимизацию. $A \cup \{$не соседи A$\}$ – независимое множество.
   Итого нас просят найти независимое множество и вернуть его вершины из левой доли.

4. **Чётность числа паросочетаний**

   Контрпример: граф из двух вершин и одного ребра.

5. **Лемма Холла**

   Запускаем Куна. Пусть очередной dfs не нашел дополняющий путь. A – посещённые вершины левой доли. Обход неудачный $\Rightarrow$ в $N(A)$ нет свободных вершин $\Rightarrow$ у каждой в $N(A)$ есть пара в A (ребро паросочетания, по которому прошёл dfs). А ещё в A есть ровно одна вершина без пары – та, от которой запускались. Итого $|A| = |N(A)| + 1$.

6. **Удаление графа**

   Здесь мы первый раз встретим приём «раздвоение графа».

   Строим новый двудольный граф: вершине v сопоставим пару вершин v_{out} и v_{in}, ребру (v, u) — новое ребро (v_{out}, u_{in}). v_{in} покрывает ребра, входящие в v, а v_{out} – исходящие $\Rightarrow$
   Ответ на задачу – ровно минимальное вершинное покрытие в новом графе.

7. **Покраска матрицы отрезками**

   Заметим, отрезок выгодно максимально продлить в обе стороны ⇒ каждую клетку можно покрасить ровно двумя способами – или горизонтальным, или вертикальным отрезком. Рассмотрим множества H и V всех максимальных по включению горизонтальных и вертикальных белых отрезков. Каждая клетка – ребро между горизонтальным и вертикальным отрезком, которые её покрывают. Получили двудольный граф, ответ – min VC этого графа.

8. **Можно снимать часть пометок**

   Доказывая корректность Куна, мы доказали «если из вершины нет дополняющего пути, то и никогда больше не будет» ⇒ если мы посетили какие-то вершины в неуспешном запуске dfs, то из них никогда не будет дополняющего пути ⇒ их пометки можно никогда не очищать.

9. **Жадная инициализация не нужна**

   Как в коде «`for v do dfs(v)`» работает `dfs` на первой фазе?
   Если сосед v без пары, ребро добавится в парсоч, `dfs` закончится, пометка останется.
   Если сосед v с парой, у пары уже стоит пометка ⇒ в неё не пойдём ⇒ `dfs` лишь перебрал соседей вершины ⇒ мы сделали ровно то же самое, что при жадной инициализации.

10. **Единственность максимального паросочетания**

    Ответ: не единственно iff есть чередующийся чётный путь или цикл.

    Док-во. Пусть паросочетаний два – M_1, M_2.
    Рассмотрим их симметрическую разность $M_1 \triangle M_2$, она состоит из путей и циклов.
    Нечётных путей быть не может (M_1 и M_2 максимальны) ⇒ нашли чётный путь или цикл.
    В другую сторону: если есть M и дополняющий путь или цикл, просто применим его к M.

    Упрощение: есть чётный путь ⇒ достаточно оставить первое ребро из свободной вершины.

    Как искать? Ребро из свободной вершины: просто переберём. Как найти чётный цикл? dfs от Куна ищет путь в орграфе вершин 1-й доли (из 1-й во 2-ю и сразу обратно в 1-ую), ровно в этом орграфе нужно запустить стандартный алгоритм поиска цикла.

    Недвудольном граф. Если цикл пытаться искать также, есть контрпример: рёбра 1-2-3-4-1 и 1-3, в паросочетании 2-3 и 4-1. Если пойти `dfs` по пути 1-3-2, то мы уже никогда не найдём чередующийся цикл 1-2-3-4-1.

    Недвудольном граф. Решение. Попробуем удалить каждое ребро, и поискать дополняющий путь без него. Если умеем за $T(V, E)$ искать дополняющий путь, то время $|M| \cdot T(V, E)$.

11. **Разбиение DAG-а**

    Коротко: раздвоим граф, найдём паросочетание, рёбра паросочетания образуют пути.

    Возьмём двудольный граф по n вершин в каждой доли. Проведём ребро $i \to j$ исходного графа из i-й вершины первой доли в j-ую вершину второй доли. Такая операция называется «раздвоение орграфа». В раздвоенном графе возьмём рёбра максимального паросочетания, нарисуем их в исходном, получили разбиение на минимальное число путей.

    Почему разбиение минимально? Во-первых, есть биекция между разбиениями на пути и паросочетаниями. Во-вторых, число путей равно $n - |M|$ (изначально n путей, каждое ребро паросочетания склеивает какие-то два).

12. **Разбиение на плавные подпоследовательности**

Сопоставим каждому элементу массива вершины l_i и r_i. Проведем ребро $l_i \to r_j$ iff j может быть в подпоследовательности после $i \Leftrightarrow i < j$ и $|a_j - a_i| \leqslant X$.
Хотим для каждого элемента выбрать не более одного предыдущего и не более одного следующего, это как раз соответствует паросочетанию в таком графе. У скольких элементов нет следующего, столько и подпоследовательностей.
Min число подпоследовательностей $\Rightarrow$ min число непокрытых паросочетанием правых вершин (равное числу непокрытых левых) $\Rightarrow$ max паросочетание.

13. **Ездят такси, но нам нечем платить**

Сделаем заказы ($\texttt{from}, \texttt{to}, \texttt{time}$) вершинами графа. Ребро (a, b) проведем, если такси после запроса a может успеть обслужить запрос b, то есть $\texttt{time}_a + \texttt{dist}(\texttt{from}_a, \texttt{to}_a) + \texttt{dist}(\texttt{to}_a, \texttt{from}_b) \leqslant \texttt{time}_b$ (расстояния можно насчитать заранее). Граф ациклический, так как ребра ведут вперед по времени.
Теперь то же, что в предыдущей задаче, для каждого заказа хотим найти предыдущий и следующий, обслуживаемый той же машиной.

14. **Покрытие множества паросочетанием**

Пусть доли L, R. Запустим Куна из всех вершин множества A в любом порядке. Если покрылось, хорошо, иначе покрыть нельзя.
Почему? Запуск Куна из вершин только A равносилен запуску Куна на графе из долей (A, R), так как Кун никогда не смотрит на вершины левой доли, из которых его не запускали, ведь назад в левую долю мы ходим уже по ребрам из паросочетания.
Раз Кун корректен, он верно найдет паросочетание на графе из долей (A, R).

Если хотим max паросочетание, покрывающее все вершины из A, то же самое: просто сначала запустить обход Куна из всех вершин A, а потом уже из остальных.
Если обломаемся уже на A, то ответа нет, как мы поняли выше. Иначе всё получится, так как алгоритм Куна никогда не лишает пары тех вершин, которым ее уже дал.

15. **Рёберное покрытие**

Строим ответ. Начнём с пустого множества. Добавляем в него рёбра. Каждое новое покроет или 1, или 2 новые вершины. Выгодно почаще покрывать 2, сколько раз можем так сделать? max паросочетание раз $\Rightarrow$ ищем max паросочетание M, берём его жадно, каждую не покрытую им вершину покроем любым соседним с ней ребром. Размер полученного множества $|M| + (n - 2|M|) = n - |M|$.

16. **Эйлеровость и паросочетания**

Фактически, мы хотим найти d непересекающихся паросочетаний.
Если d нечётное, найдем паросочетание, выкинем из графа.
Если d чётное, найдем эйлеров цикл (в каждой компоненте), разделим ребра по чередованию в цикле. Получим два графа степени $\frac{d}{2}$ на вершинах исходного. Рекурсивно покрасим оба графа.
На каждом уровне рекурсии мы ищем паросочетание на графах из V вершин, а ребер во всех графах суммарно E. Если время поиска паросочетания таково, что $T(V, E_1) + T(V, E_2) \leqslant T(V, E_1 + E_2)$, то потратили $\mathcal{O}(\texttt{Matching} \cdot \log d)$. Для Куна ($\mathcal{O}(VE)$) и Хопкрофта-Карпа ($\mathcal{O}(E\sqrt{V})$) это верно.

(*). Если $d = 2k+1$, за $\mathcal{O}(\texttt{Matching}+E)$, разделим граф на 3 части: $d = k+k+1$. Рекурсивно вызовемся от k, получили $k+1$ парсочей. *Оптимизация:* во второй ветке вызовемся не от k, а докинем парсочей до степени двойки, степень двойки умеем решать за $\mathcal{O}(E \log d)$.
Итого: $T(V, E) = \texttt{Matching} + E + T(V, \frac{E}{2}) + E \log d = \mathcal{O}(\texttt{Matching} \cdot \log d + E \log d)$.

17. **(*) Birkhoff–von Neumann decomposition**

a) $\mathcal{O}(n^5)$. Построим полное паросочетание в графе, где доли – множество строк и множество столбцов, ребра – ненулевые клетки матрицы. $\mathcal{O}(n^3)$.
Выберем минимальное по весу ребро x и вычтем из матрицы перестановочную подматрицу, заданную паросочетанием, с коэффициентом x.
Из каждой строки и столбца вычлось ровно x, сумма осталась постоянной (не нужно, чтобы она была ровно 1). При этом занулилась хотя бы одно клетка, за $\leqslant n^2$ таких операций полностью разложим матрицу.

Почему паросочетание найдется? По лемме Холла. У любых k строк сумма k, если у них $t < k$ столбцов-соседей, то у этих столбцов сумма хотя бы $k > t$.

b) $\mathcal{O}(n^4)$. То же самое, но не ищем паросочетание с нуля, а достраиваем для строк с обнулившимися ребрами. То есть проход Куна за $\mathcal{O}(n^2)$ вызовется $\leqslant n^2$ раз.

18. **(*) Прямоугольные многоугольники**

Нам нужно соединить с чем-то только вогнутые внутрь углы. Для каждого есть ровно два отрезка, которые можно провести. У некоторых углов какие-то отрезки могут совпадать.
Двудольный граф: вертикальные и горизонтальные интересные отрезки – вершины, углы – ребра. Ищем минимальное вершинное покрытие.
$\mathcal{O}(n)$ углов, $\mathcal{O}(n)$ отрезков, итого $\mathcal{O}(n^2)$.

Домашнее задание

3.1. Обязательная часть

1. **(1.5) Петя и гиперкуб**

   У маленького Пети любимая игрушка – d-мерный гиперкубик ($d \leqslant 10$). Некоторые вершины у куба белые, некоторые чёрные. Вы знаете цвета всех вершин, помогите Пете перекрасить как можно меньше белых вершин в красный цвет так, чтобы не было двух смежных по ребру белых вершин.

2. **(2+1) Разбиение на циклы**

   a) **(2)** Разбейте вершины **ор**графа на циклы. Каждая вершина должна быть покрыта ровно одним циклом. Либо скажите, что это невозможно.
   b) **(1)** Разбейте вершины **ор**графа с весами на ребрах на циклы так, чтобы вес максимального ребра был минимальным.

3. **(3) Степень не более двух**

   Дан произвольный **не**орграф. Найдите max по размеру мультимножество рёбер, такое, что степень каждой вершины $\leqslant 2$. Любое ребро можно брать два раза.

4. **(3) Множество прямых**

   Дано N различных прямых. Выбрать max по размеру подмножество прямых, такое, что никакие две прямые не параллельны, и никакие прямые не пересекаются в точке с $x = 0$.

3.2. Дополнительная часть

1. **(1.5+1.5) Лексмин вершинное покрытие**

   Дан двудольный граф. Среди всех вершинных покрытий минимального размера найти лексикографически минимальное.

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

   Сравнение вершинных покрытий: отсортируем вектора номеров вершин и сравним вектора на больше-меньше.

2. **(2) Вершинно-взвешенное паросочетание**

   В двудольном графе у каждой вершины левой доли есть вес a_i. Вес ребра равен весу его конца из левой доли. Найдите паросочетание max веса за $\mathcal{O}(VE)$.
   Правильное решение без доказательства оценивается в 1 балл.

3. **(3) Вершинно-взвешенное паросочетание-2**

   В двудольном графе у каждой вершины левой доли есть вес a_i, у каждой вершины правой вес b_i. Вес ребра равен сумме весов его концов. Найдите паросочетание max веса за $\mathcal{O}(VE)$.
   Правильное решение без доказательства оценивается в 2 балла.

4. **(3) Оптимизация H**

   Рассмотрим такую оптимизацию Куна.

```
bool kuhn(int v) {
    for (int u : graph[v])
        if (!used[u]):
            used[u] = true;
            if (right_pair[u] == 0 || kuhn(right_pair[u])):
                right_pair[u] = v;
                used[u] = false;
                return true;
    return false;
}
...
fill(used.begin(), used.end(), false);
for (int v = 1; v <= n; ++v) kuhn(v);
```

   Какой физический смысл происходящего?
   Оцените как-нибудь сверху и снизу асимптотику (в зависимости от $E, V, |M|$).
   Круто, если оценки сойдутся.
   Приведите тест (небольшой), на котором реализация работает некорректно.

   (*) А можно ли ее завалить, если в начале сделать `random_shuffle` всех ребер?
   Последняя часть задачи исследовательская. Число баллов заранее не определено.