SPb HSE, 1 курс, осень 2024/25
Практика по алгоритмам #13

Наконец-то не динамика

11 декабря

Собрано 10 декабря 2024 г. в 18:16

Содержание

Наконец-то не динамика

1. **Компактификация**

   Придумать способ хранения графа, чтобы памяти он занимал $\mathcal{O}(m)$,
   а dfs с ним работал за $\mathcal{O}(m + n \log m)$.

2. **Как хранить?**

   Как хранить граф, чтобы уметь за $\mathcal{O}(1)$ отвечать на запросы четырёх типов: добавить ребро, удалить ребро, связны ли две данные вершины, перебрать всех соседей вершины v за $\mathcal{O}(deg_v)$.

3. **Треугольник**

   Дан неорграф, посчитать количество треугольников (K_3). Это не про dfs.
   (a) $\mathcal{O}(V^3)$, (b) $\mathcal{O}(VE)$, (c) оптимизации специально не рассказываем.

4. **Достижимость множеств**

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

5. **Цикл через вершину**

   Дан орграф и вершина v. Найти простой цикл, проходящий через v за $\mathcal{O}(E)$.

6. **Дерево**

   Дан граф. Проверить, является ли деревом. (a) $\mathcal{O}(E)$, (b) $\mathcal{O}(V)$.

7. **А вы точно граф?**

   Даны две операции: $x \to (2x{+}1) \bmod N$ и $x \to (x^2 + 7) \bmod N$. x целое, $N = 10^6 + 3$.
   Можно ли из числа 13 получить число 17? Если да, то как?

8. **Ямки**

   В некоторых вершинах орграфа есть ямки.
   Нужно дойти из a в b не более двух раз наступив в ямку. Как?

9. **Ацикличная ориентация**

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

10. **Транзитивное замыкание**

    Дан орграф, построить матрицу достижимости. $V \leqslant 20\,000$, $E \leqslant 200\,000$.

    (a) DAG; (b) произвольный граф.

11. **В мире перестановок**

    Даны перестановки p и q из 10 элементов. Можно ли в каком-то порядке применяя к тождественной перестановке перестановки p и q, получить перестановку z?

12. **Строкапуты**

    Дан орграф, на рёбрах написаны буквы. Найдите путь в орграфе, на котором написана ровно данная строка s. За полином.

13. **Разбиение на дружелюбные доли**

    У каждой вершины не более 3 врагов.
    Разбить на 2 доли так, чтобы с вершиной в долю попало не более 1 врага. $\mathcal{O}(V+E)$.

14. **Снятие пометок и циклы**

    Вася хочет перебрать все-все-все циклы в неорграфе. Для этого он придумал «запустить обычный dfs для поиска цикла и снимать при выходе из dfs пометки». За сколько в худшем работает такой алгоритм? Какие у него проблемы? Как всё-таки посчитать число различных циклов?

15. **(*) Здоровенная зебра**

    Дана матрица из чисел. Выберите наибольшее связное по стороне множество клеток, чтобы было использовано не более двух различных чисел. $\mathcal{O}(wh)$.

16. **(*) Максимальное число различных циклов**

    Есть неорграф, нужно выбрать максимальное число простых циклов так, чтобы каждый следующий содержал хотя бы одно новое ребро (ни разу не использованное в предыдущих циклах). $\mathcal{O}(V+E)$.

Разбор задач практики

1. **Компактификация**

   Сортированный список рёбер (a, b).
   Перебор рёбер из a: бинпоиском найти первое a и за степень a перебрать рёбра.
   Время `dfs` – $\mathcal{O}(n \log m + m)$. Памяти нужно $8m$ для орграфа $16m$ для неорграфа.

2. **Как хранить?**

   Хранить для каждой вершине `unordered_set` соседей.

3. **Треугольник**

   $\mathcal{O}(V^3)$ — перебираем тройку вершин и матрицей смежности проверяем.

   $\mathcal{O}(VE)$ — перебираем вершину и ребро, и матрицей смежности проверяем.

4. **Достижимость множеств**

   dfs из всех вершин A, проверить, что посещена хотя бы одна из B.

5. **Цикл через вершину**

   `dfs` из вершины v «в себя». Можно модифицировать `dfs`, чтобы он умел в v зайти второй раз. Можно создать v' копию v и `dfs`: $v \to v'$.

6. **Дерево**

   Проверить $m = n - 1$, проверить связность. Проверка связности будет работать за $\mathcal{O}(n)$, т.к. мы уже проверили, что рёбер $n - 1$.

7. **А вы точно граф?**

   В условии описан граф из N вершин и $2N$ рёбер. dfs по нему.

8. **Ямки**

   Вершина нового графа $\langle v, k\rangle$ — вершина старого и сколько раз уже были в ямке.

9. **Ацикличная ориентация**

   Ориентируем ребра от меньшей вершины к большей.

   Способ #2. Запустим dfs, ориентируем все ребра вниз, то есть ребра дерева и обратные из предка в потомка. В обоих случаях цикла нет, ибо на любом пути либо строго растёт номер вершины, либо глубина в дереве dfs.

10. **Транзитивное замыкание**

    Сконденсируем граф. Из каждой вершины достижима её ксс и ещё какие-то. Какие?

    Динамика по конденсации `bitset dp[v]` – множество вершин, достижимых из v.
    `dp[v]` – это `OR` bitset'ов детей v. Динамика ленивая (dfs-о-динамика).

11. **В мире перестановок**

    Видим граф из $10! < 4\,000\,000$ вершин, на нём dfs.

12. **Строкапуты**

    Динамика `dp[v,i]` – можно ли оказаться в вершине v, выписав первые i символов строки s.
    Начальное состояние $\forall v$ `dp[`v`,0] = true`. Конец пути – любая v: `dp[`v`,`$|s|$`] = true`.

13. **Разбиение на дружелюбные доли**

Разобьем как-нибудь. Метод локальных оптимизаций: если у какой-то вершины в её доле 2 врага, перекинем её в другую долю.

Каждый раз уменьшается суммарное число рёбер-врагов внутри долей, поэтому надо $\mathcal{O}(E)$ шагов. Чтобы быстро находить плохие вершины, сделаем это dfs-ом.

```
void dfs(int v):
    if (bad(v)):
        color[v] ^= 1;
        for (int x : g[v]) dfs(x);
for (int v = 0; v < n; v++)
    dfs(v);
```

Заметим, что тут нет пометок посещенных вершин, но рекурсивные вызовы возникают только после перекидывания вершины, поэтому время $\mathcal{O}(V+E)$.

14. **Снятие пометок и циклы**

Работает за $n!$ на полном графе. Важно запомнить, что dfs быстро ищёт «один какой-то цикл», а вот «найти все циклы» – это уже экспонента.

Проблема: каждый цикл найдётся несколько раз (разный предпериод, 2 направления прохождения, вход в разные вершины).

Возможное решение проблемы = после нахождения цикла переберём циклический сдвиг и направление, выберем цикл с минимальным `vector<int>`, сложим всё это добро в `set`.

15. **(*) Здоровенная зебра**

Сожмём компоненты одного цвета. В новом графе у каждой вершины есть вес (размер компоненты) и цвет (число из матрицы). Тип ребра в сжатом графе = типы его концов.

Для каждого типа возьмём все рёбра такого типа и запустим `dfs` по ним.
Чтобы суммарное время работы было линейным `dfs` по должен работать ровно за $\mathcal{O}(E)$ – обнулять пометки за $\mathcal{O}(1)$ и ходить только по вершинам, из которых $\exists$ рёбра.

16. **(*) Максимально число различных циклов**

Рассмотрим любое остовное дерева `dfs`. Его рёбра нужны, чтобы хотя бы посетить вершины. А все не древесные рёбра образуют циклы, всего $m-(n-1)$ циклов.

Кстати, полученные циклы образуют базис в «пространстве циклов».

Домашнее задание

3.1. Обязательная часть

1. **(2) Путь через ад**

   Дан орграф. В некоторых вершинах живут монстры. Монстры бывают трёх типов. Проходя через некоторые вершины, можно получить иммунитет к некоторым типам монстров (три разных иммунитета). В каждый момент времени каждый из трёх иммунитетов или есть, или нет, копить их нельзя. Иммунитет при встрече с монстром спасает от гибели ровно один раз и пропадает. Можно ли из вершины a дойти до вершины b и не умереть? ($V, E \leqslant 10^5$).

   Подсказка: важно увидеть правильный граф.

2. **(2) Минимизация максимального ребра на цикле**

   Дан неорграф с целыми положительными весами на рёбрах. Найти в неорграфе такой цикл, что максимальный вес ребра этого цикла минимален. $\mathcal{O}((V+E)\log E)$.

   Решить, используя только известное нам. Подсказка: откуда берутся логи?

3. **(2) Ещё один вариант разбиения на две доли**

   Разбить множество вершин неорграфа на две доли так, чтобы у каждой вершины был сосед в другой доле. $\mathcal{O}(V+E)$.

4. **(3) Ромбики**

   Найти в неорграфе «ромбик с диагональю»: 4 вершины a, b, c, d и 5 рёбер ab, bc, cd, da, ac. Оценка: (1) $\mathcal{O}(V^3)$, (1) $\mathcal{O}(VE)$, (1) $\mathcal{O}(\frac{VE}{w})$, (+1) $\mathcal{O}(E^{3/2})$.

5. **(3) Покраска в 3 цвета**

   Дан изначально пустой неорграф. В него добавляются рёбра и вершины. При этом поддерживается инвариант, что степени всех вершин не более 5. Поддерживать покраску вершин в 3 цвета так, чтобы у каждой было не более одного соседа того же цвета. При добавлении новых ребра/вершины обновлять покраску за амортизированное $\mathcal{O}(1)$.

3.2. Дополнительная часть

1. **(3) Минимизация максимальной исходящей степени**

   Дан неорграф. Ориентировать его так, что максимальная исходящая степень была минимальна. $\mathcal{O}(E^2)$. Можно еще на полилог.

2. **(3) Существование пути**

   Напишите детерминированное решение, которое имеет доступ к оракулу $g(i, j)$ – есть ли ребро между i и j, и должно, используя полилогарифм от n памяти проверить наличие пути из a в b в графе из n вершин.