Первый курс, осенний семестр 2020/21
Практика по алгоритмам #7

DSU, MST

26 февраля

Собрано 3 марта 2021 г. в 00:48

Содержание

DSU, MST

1. **Online двудольность**

   Дан неорграф. В него в online добавляются рёбра.
   После каждого добавления говорить, двудолен ли граф?

   a) $\mathcal{O}(m \log n)$ через перекрашивание компонент.
   b) Быстрее через DSU.

2. **Чётность**

   В каждой клетке прямой записано число 0 или 1. Поступает информация: чётность количества единиц на отрезке $[L_i, R_i]$, найти первый запрос, после которого данные противоречивы.

3. **Единственность MST**

   a) Доказать, что в графе с различными весами рёбер MST единственно.
   b) Проверить, что минимальное по весу остовное дерево единственно.

4. **Перестроение MST**

   Дан взвешенный граф G. Дано минимальное остовное дерево на нем.
   У ребра e поменяли вес. Найти новое минимальное остовное дерево за $\mathcal{O}(V + E)$.

   Разберите все 4 *случая!*

5. **Второе по весу ST**

   Найти второе по весу остовное дерево: (a) за полином, (b) за $\mathcal{O}(V^2 + E)$.

6. **Тест для Борувки**

   Постройте пример для алгоритма Борувки, на котором он делает

   a) $\Theta(\log n)$ фаз.
   b) $\Theta(1)$ фаз.

7. **Казалось бы, причём здесь СНМ?**

   У нас есть массив длины n, мы хотим выполнить m запросов вида «покрасить отрезок $[l_i, r_i]$ массива в цвет c_i» и вывести, что получилось в конце. Решите за $\mathcal{O}(n \log m)$.

   Подсказка: попытайтесь выполнять запросы, начиная с последнего.

8. **Краскал наоборот**

   Пусть дан связный взвешенный неорграф, будем рассматривать его ребра в порядке невозрастания веса и удалять текущее ребро, если связность графа при этом не нарушается. Докажите, что этот алгоритм находит минимальный остов, или придумайте контрпример.

9. **(*) Ориентированное MST**

   Дан орграф, постройте остовное дерево с корнем в вершине 1 минимального веса. Все ребра в нем должны быть направлены от предков к потомкам.

10. **(*) MST через k ближайших**

    Рассмотрим алгоритм построения остовного дерева на плоскости: найдем к каждой точке k ближайших, на полученном графе за $\mathcal{O}(nk \log n)$ найдем минимальное остовное дерево. Постройте контрпример.

11. (*) **Random MST через k ближайших**

    А для точек равномерно распределённых внутри $[0..10^9] \times [0..10^9]$ этот алгоритм отлично работает. Почему? Для какого k его запускать?

12. (*) **Все деревья связаны**

    Пусть дан взвешенный связный неорграф $G = \langle V, E \rangle$ с выделенной вершиной s. Все веса положительны и различны. Могут ли какое-либо минимальное покрывающее дерево в G и какое-либо дерево кратчайших путей из s не иметь ни одного общего ребра? Если да, приведите пример. Если нет, докажите, что такого не может быть.

13. (*) **Второй минимальный простой путь**

    В орграфе с положительными весами рассмотрим множество всех **простых** путей из s в t. Упорядочим это множество по весу пути. Найдите второй элемент.

14. (*) **k-е по весу ST**

    Среди всех остовных деревьев неорграфа верните k-е по весу.

Разбор задач практики

1. **Online двудольность**

   a) $\mathcal{O}(m + n \log n)$ на все запросы. Поддерживаем цвета вершин (изначально все 1) и «DSU на списках» для компонент связности. Пришло ребро. Рассмотрим случаи:

   • Ребро соединяет вершины одной компоненты? Концы ребра одного цвета ⇒ граф отныне и во веки веков не двудолен, есть нечётный цикл. Концы разных цветов ⇒ всё хорошо, ничего не делаем.

   • Ребро соединяет вершины разных компонент? Концы ребра разных цветов ⇒ всё хорошо, ничего не делаем. Концы ребра одного цвета ⇒ перекрашиваем все вершины меньшей компоненты.

   Каждая вершина поменяет свой цвет не более $\mathcal{O}(\log n)$ раз ⇒ $\mathcal{O}(m + n \log n)$.

   b) $\mathcal{O}((m + n)\alpha(n))$.

   Способ 1 (основной). У каждой компоненты связности есть одна или две доли. Храним доли в DSU. Если у компоненты две доли, их корни a_i, b_i, то будем хранить «ссылку на вторую долю» `link[`a_i`] = `b_i`, link[`b_i`] = `a_i. При добавлении ребра (x, y), три случая:

   1. `root(a) == root(b)` ⇒ нечётный цикл.
   2. `link[root(a)] == root(b)` ⇒ ребро между долями, ничего не делаем.
   3. Ребро между компонентами.
      У нас есть 4 доли – корни и их `link`-и, вызовем два `join`-а, обновим `link`-и.

   Способ 2. Храним цвета вершин и компоненты связности в «DSU на деревьях», перекрашиваем лениво. На рёбрах DSU указаны цвета 0 и 1.
   Цвет вершины v – `XOR` значений на пути от v до корня DSU.
   При `join` ставим на новое ребро 1, если нужно перекрашивание.
   При сжатии путей на новое ребро ставим цвет, равный цвету вершины.

2. **Чётность**

   Число единиц на отрезке $[L, R]$ чётно, то число единиц на префиксах $[1, L-1]$ и $[1, R]$ имеет одинаковую чётность, иначе разную
   ⇒ видим граф, где вершины – префиксы, рёбра – отрезки
   ⇒ задача аналогична предыдущей, только здесь нам сообщают, либо что концы ребра должны иметь одинаковый цвет (лежать в одной доли), либо разный (лежать в разных долях). Каждая компонента связности двудольна, можем, как и в решении выше хранить её, как «пара двух долей».

   Решение без двудольности. Будем проводить рёбра с весами 01. СНМ. Каждая компонента — множество в СНМ. Когда приходит ребро $a \xrightarrow{w} b$, если a и b уже в одной компоненте, проверяем, что $w_e =$ `xor` на пути $a \rightsquigarrow b$, а это ровно $up(a) \oplus up(b)$, где $up(a)$ – `xor` на пути от a до корня. $up(a)$ считается СНМ-ом также, как $get(a)$. Когда делаем `join`, проводим ребро не между a и b, а между $get(a)$ и $get(b)$ веса $up(a) \oplus up(b) \oplus w$.

   P.S. При больших координатах в offline можно сжать координаты, в online использовать хеш-таблицу (если `p[x]` еще не был определен, то `p[x] = x`).

3. **Единственность MST**

   a) Как в лемме о разрезе: рассмотрим Краскала, пусть в какой-то момент он не берёт минимальное ребро, увидим цикл, сделаем замену, благодаря тому, что веса различны, вес остова строго уменьшится.

   b) **Простой способ.** Строим как-нибудь MST, обозначим его T.
   Строим второе дерево: запустим Краскала, который пытается при равенстве весов сперва брать рёбра не из T. Если новое дерево равно T, то MST единственно.

   Второй способ.
   Запускаем алгоритм Краскала. Внутри обрабатываем ребра группами одинакового веса. Посмотрим на все рёбра веса w. Выберем из них те, которые сейчас соединяют разные компоненты (образованные меньшими ребрами), но в дерево пока не добавим.
   По выбранным рёбрам делаем второй проход, но уже объединяя компоненты и добавляя рёбра в остовное дерево.
   Если какое-то помеченное ребро e_i не добавилось в дерево, значит, его добавление создаёт цикл по рёбрам веса $w \Rightarrow$ MST не единственно. Если ни разу не возникла такая ситуация, MST единственно. Доказательство аналогично (a) «пусть не взяли, сделаем замену...».

4. **Перестроение MST**

   Обозначим наше минимальное дерево T, а меняющееся ребро e.

   • Если **уменьшилось** $e \in T$, или **увеличилось ребро** $e \notin T$, MST не изменилось.

   • Пусть **увеличилось** $e \in T$. Удалим e, получим разрез.
   Добавим минимальное ребро через разрез (им может оказаться снова e).
   Время $\mathcal{O}(E)$ (раскрасить компоненты, перебрать рёбра).
   Корректность. Выбранное нами ребро точно нужно брать по лемме о разрезе, возьмём. Далее запустим Краскала, он возьмёт ровно те же рёбра, что и раньше.

   • Пусть **уменьшилось** $e \notin T$. Рассмотрим $T+e$, e образует цикл. Если в цикле максимальное ребро f больше e, заменим f на e. Время работы = времени поиска максимума на пути.
   Корректность. Если $w_f \leqslant w_e$, шаги Краскала не изменятся. Если $w_f > w_e$, удалим f, получим разрез, e – минимальное в разрезе (если не e и не f, T было не минимальным) $\Rightarrow$ по лемме о разрезе e выгодно взять. Далее Краскал возьмёт ровно те же рёбра, что и раньше.

5. **Второе по весу ST**

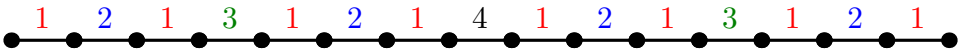
   Обозначим первое по весу T_1, второе по весу T_2.
   Переберём (угадаем) $e \in T_2 \setminus T_1$, рассмотрим $e + T_1$, $e(a, b)$ образует цикл.
   Найдём f максимальное ребро на пути $a \rightsquigarrow b$ в T_1.
   При замене $f \to e$, вес остова меняется на $w_e - w_{f(e)}$, $T_2 = T_1 + e - f$, где e: $w_e - w_{f(e)} = \min$.

   Корректность. Докажем, что $\forall e$ среди всех остовов, содержащих e, мы нашли минимальный.
   Мысленно запустим Краскала, он взял ровно те же рёбра, что и при построении T_1. Кроме последнего рассмотренного ребра на цикле – f.

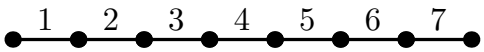
   Время работы.
   Искать максимальный вес на пути можно за $\mathcal{O}(n)$, тогда получим время $\mathcal{O}(mn)$.
   Можно за $\mathcal{O}(n^2)$ предподсчитать все n^2 максимумов, тогда время $\mathcal{O}(n^2 + m)$.
   Можно искать максимум на пути ещё быстрее. Даже за $\mathcal{O}(1)$.

6. **Тест для Борувки**

   a) $\Theta(\log n)$ фаз. Бьем вершины на пары, проводим в парах ребра веса 1. Полученные компоненты соединяем ребром веса 2, и так далее. То есть граф размера 2^k с k фазами строится рекурсивно: соединим ребром веса k два графа размера 2^{k-1}.



   b) $\Theta(1)$ фаз.



   Еще можно сделать бамбук с одинаковыми весами, в котором вершины пронумерованы по порядку. Тогда за счет сравнения пар тоже сработает за одну фазу.

   Ещё можно расставить веса в произвольном графе: возьмем любое остовное дерево и поставим вес i ребрам между вершинами глубины $(i-1)$ и глубины i. На ребрах не из дерева вес ∞.

7. **Казалось бы, причём здесь СНМ?**

   Обрабатывая запросы с конца, идём слева направо и красим ещё непокрашенные клетки.
   Как быстро пропускать непокрашенные?
   Сжатие путей: изначально `p[i]=i+1`, после покраски слева направо путь сжимается.
   Сжатие путей, как и СНМ с одной эвристикой даст суммарное время $\mathcal{O}((m+n)\log n)$.

   Полноценный СНМ. В СНМ храним области «что-то покрашенное и, возможно, крайняя правая не покрашенная». Для корня множества помним крайнюю правую клетку. Тогда

```
for (int l = right[get(l)]; l <= r; l = right[get(l)]) {
  if (color[l] == -1) color[l] = c;
  join(l, l+1);
}
```

8. **Краскал наоборот**

   Работает. По индукции. Потому что Краскал не возьмёт последнее ребро. Если все веса различны – очевидно. Если бывают одинаковые – сделаем их различными, заменив на $\langle w_e, e\rangle$.

9. **(*) Ориентированное MST**

   Отличный текст от Олега Давыдова.

   Пусть все вершины достижимы из корня, иначе сразу ясно, что решения нет.
   Наблюдение: если взять $\forall v \neq$ `root` и вычесть из всех входящих в неё рёбер число x, то вес любого дерева изменится на x, так как в v входит только одно ребро дерева. Значит, MST останется тем же.

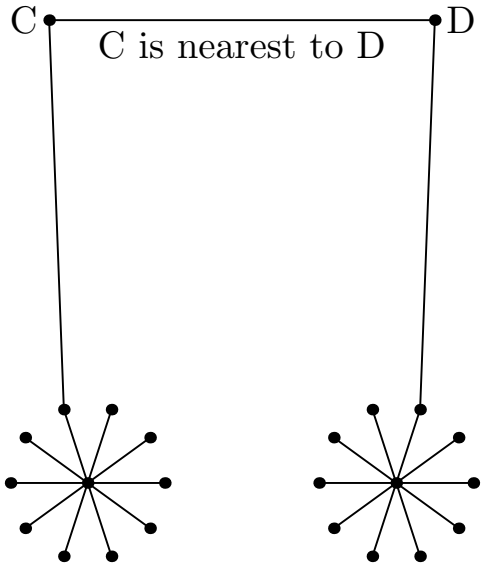
   $\forall v \neq$ `root` находим $x_v = \min w(u,v)$ и вычитаем x_v из всех входящих в v ребер.
   Все ребра стали $\geqslant 0$, и в каждую вершину v входит хотя бы одно нулевое e_v.
   Если теперь есть дерево из нулей, оно – ответ. Иначе есть цикл из нулей: из какой-то v не смогли дойти до корня по нулевым ребрам, значит, зациклились.

   Сожмем цикл и рекурсивно запустим алгоритм на новом графе.
   Получили ответ для графа, где некоторые вершины – сжатые циклы. Эти циклы нужно расжать. При расжатие цикла в остовное дерево добавляются все ребра цикла, кроме входящего в вершину, у которой уже есть входящее ребро.

Каждая фаза за $\mathcal{O}(E)$, за фазу хотя бы на 1 вершину стало меньше $\Rightarrow \leqslant V$ фаз $\Rightarrow \mathcal{O}(VE)$.

10. **(*) MST через k ближайших**

Пример для $k < \frac{n}{2}$: $\frac{n}{2}$ точек вокруг $(0, 0)$ и $\frac{n}{2}$ точек вокруг $(10^9, 0)$. Остов выйдет несвязным.
Можно ещё построить пример, в котором мы возьмём лишнее ребро при $k < \frac{n}{2} - 1$:



11. **(*) Random MST через k ближайших**

Чем k больше, тем меньше вероятности ошибки.
Авторское решение задачи из контеста использует $k = 13$.

Обозначим сторону квадрата за M. Квадрат со стороной $m = \frac{M}{\sqrt{n}}$ назовём единичным.
Весь квадрат $M \times M$ разбивается на n единичных квадратов.
Единичный квадрат пуст с вероятностью $(1 - \frac{1}{n})^n \underset{n\to\infty}{\to} e^{-1}$.
$\forall t$ квадрат со стороной tm **не** пуст с вероятностью $\approx 1 - e^{-t^2}$.
Все $\frac{n}{t^2}$ квадратов со стороной tm **не** пусты с вероятностью $(1 - e^{-t^2})^{\frac{n}{t^2}}$.
При $t = 2\sqrt{\ln n}$ получаем $(1 - \frac{1}{n})^{\frac{n}{\ln^2 n}} \underset{n\to\infty}{\to} 1 \Rightarrow$ почти наверняка все квадраты такого рамера содержат хотя бы одну точку.
Пользуемся леммой о разрезе. Рассмотрим произвольный разрез.
Если в каком-то из квадратов $tm \times tm$ есть точки из разных частей разреза, расстояние между ними $\leqslant \sqrt{2}tm$.
Иначе есть два соседних квадрата, которые лежат в разных частях разреза, расстояние между точками $\leqslant \sqrt{5}tm$.
То есть, длина ребра, которое нужно добавить в MST не более $r = \sqrt{5}tm$.
Посмотрим, сколько точек может лежать в круге радиуса r. Площадь круга равна $5\pi t^2 m^2 = 20\pi(\ln n)m^2$. Матожидание числа точек в этом круге $E = 20\pi(\ln n)$.
Вероятность того, что туда попадёт более $40\pi \ln n$ точек, незначительна.
$\Rightarrow$ нам точно хватит $k = 40\pi \ln n$.

12. **(*) k-е по весу ST**

Йен, будем класть в кучу кандидатов и то, к каким «классам» они относятся.
Сначала другой способ получения второго: перебираем, какое ребро запретить, т.е. какое

есть в T, но нет в T_2.
При запрете ребра получаем разрез дерева, находим min ребро через него.
Итого $\mathcal{O}(nm)$: для каждого из $(n-1)$ ребра остова ищем ему замену за $\mathcal{O}(m)$.
Что задает «класс» решений? Множество запрещенных ребер. Итого $\mathcal{O}(knm)$.

Домашнее задание

3.1. Обязательная часть

1. **(1) Ремонт дорог**

   Дана страна, состоящая из городов и двусторонних дорог. Каждая дорога или в рабочем состоянии, или требует ремонта стоимости w_i. Ездить можно только по дорогам в рабочем состоянии. За минимальную стоимость отремонтировать некоторые дороги так, чтобы из любого города страны можно было добраться в любой другой.

2. **(2) Проверка на минимальность**

   Пусть мы умеем искать **минимум** на пути в дереве за время $M(n)$. Дано остовное дерево. За сколько можно проверить то, что оно является минимальным по весу?
   Доказать максимально строго.

3. **(2) Насколько сжатие путей быстрое?**

   Доказать, что СНМ с одной эвристикой сжатия пути, к которому поступают следующие запросы «сперва только `join`-ы вида `p[root] = x` (где `root` – именно корень, а `x` – любая вершина другого множества), затем только `get`-ы», работает за $\mathcal{O}(n + m)$, где n – число элементов, m – суммарное число запросов.

4. **(3) Минимум на пути**

   Дано корневое дерево из n вершин. Все ребра ориентированы к корню, на них есть веса. Путь называется вертикальным, если его вершина-конец является предком вершины-начала. Даны m вертикальных путей, за $\mathcal{O}((n+m)\log n)$ времени и $\mathcal{O}(n+m)$ дополнительной памяти найти минимум на каждом из путей. **(2)** Балла за решения для «бамбука».

   При решении этой задачи вам могут помочь следующие идеи: (1) сжатие путей, (2) объединение множеств за $\mathcal{O}$(размера меньшего).

   Дерево отрезков и прочие неизвестные нам структуры использовать нельзя.

5. **(2) Кратчайший путь**

   За $\mathcal{O}(E \log V)$ сделать предподсчёт, а затем для любой пары вершин за $\mathcal{O}$(размера ответа) возвращать путь между a_i и b_i, в котором минимальный вес ребра максимален.

6. **(2) Random и СНМ**

   Рассмотрим реализацию СНМ:

```
1 int get(int v) { return v == p[v] ? v : get(p[v]); } // без сжатия путей!
2 void join(int a, int b) {
3   a = get(a), b = get(b);
4   if (rand() & 1) swap(a, b);
5   p[a] = b;
6 }
```

   За сколько в худшем по всем тестам случае работает такая реализация?
   Докажите оценку сверху, приведите тест, на котором она достигается.

   (+1) За оценку того же кода с `get` со сжатием путей. Здесь можно предположить, что уже решена задача доп.3 из текущего дз.

3.2. Дополнительная часть

1. **(3) Dynamic connectivity in directed graph**

   Дан ацикличный орграф. Нужно за $\mathcal{O}(n^2)$ обрабатывать запросы «добавить ребор» и «удалить ребро». Гарантируется, что граф всегда остается ацикличным. Также нужно за $\mathcal{O}(1)$ отвечать на запрос «есть ли путь из a в b»?

   Подсказка: наш алгоритм будет вероятностным.

2. **(3) Dynamic 2-connectivity**

   В неорграф добавляются ребра.
   Нужно после каждого запроса добавления говорить, сколько в графе мостов.

   Подсказка: число мостов – число рёбер в дереве рёберной двухсвязности.

3. **(3) Одного сжатия путей мало...**

   Построить тест, на котором m запросов к СНМ-у с одной эвристикой сжатия путей работают за $\Omega(m \log n)$. Можете делать любые запросы. $m = \Theta(n)$.