Algorithm

KhuzinTka@yandex.ru

December 2021

1. Найти в графе количество треугольников за O(E^1.5)
2. Ориентировать неор граф так, чтобы он стал сильносвязным за O(V+E) или сказать, что это невозможно
3. Разбить все ребра неорграфа на минимальное число путей за $\mathcal{O}(V + E)$.
4. Для каждой пары вершин в графе найти $w[a, b]$ – такой минимальный вес, что из a в b есть путь по рёбрам веса <= $w[a, b]$. O(V^2)
5. Дана система из m неравенств на n переменных вида $x_i - x_j \le \delta_{ij}$. Найти решение системы или сказать, что его не существует, за $\mathcal{O}(nm)$.
6. Найти в графе цикл минимального среднего веса V, E <= 2000, |w_i| <= 10**9
7. Дан массив чисел. За $\mathcal{O}(\log n)$ в online обрабатывать запросы:
   • посчитать сумму кубов чисел на отрезке $[L, R]$;
   • прибавить x ко всем числам на отрезке $[L, R]$;
   • получить значение i-го числа.
8. Есть массив из нулей и единиц. В online за $\mathcal{O}(\log n)$ отвечать на запросы: поменять элемент; найти ближайший слева/справа ноль к позиции i.
9. Запросы: сумма на отрезке, прибавить $ai + b$ к i-у элементу отрезка $[L, R]$ для всех i. То есть, сделать array[L + i] += a*i + b. O(log n)

1) Найти в графе количество треугольников за $O(E^{1.5})$

Решение. Разделим все вершины графа на белые и черные. Пусть белые вершины — те, у которых количество соседей (смежных вершин) меньше, чем $\sqrt{E}$. Черные — у которых количество соседей больше (либо равно) $\sqrt{E}$. Треугольников, образованных черными вершинами, всего $C^3_{\sqrt{E}} = O(E^{1.5})$. Треугольников, образованных белыми вершинами, тоже всего $O(E^{1.5})$. Докажем это. Зафиксируем белую вершину, входящую в какой-то треугольник, тогда для нее будет $O(E)$ ребер таких, что они являются инцидентными этой белой вершине. Также для каждого такого ребра будет $O(\sqrt{E})$ парных ребер, в силу определения белой вершины. Тогда всего таких треугольников $O(E^{1.5})$. Что и требовалось доказать.

Теперь как треугольники найти? Переориентируем ребра так, чтобы ребро вело от вершины с меньшей степенью к вершине с большей. В таком случае утверждается, что из каждой вершины графа выходит не более $O(\sqrt{E})$

ребер, так как степень любой белой вершины — $O(\sqrt{E})$, а из любой черной вершины ребра идут только в черные вершины, который $O(\sqrt{E})$.

Теперь для каждой вершины графа пометим как-то ее соседей и запустим из нее поиск путей длины два. Каждый раз, когда попали в помеченную вершину, мы нашли новый треугольник. Для каждого первого ребра пути рассмотрим $O(\sqrt{E})$ ребер, значит асимптотика поиска — $O(E^{1.5})$.

2) Ориентировать неориентированный граф так, чтобы он стал сильно связным за $O(V+E)$ или сказать, что это невозможно.

Решение. Очевидно, что несвязный граф никак нельзя сделать сильносвязным, как бы мы не старались. Поэтому сначала с помощью поиска в глубину за $O(V+E)$ проверим, что граф связен. Если граф оказался связен, то по теореме Роббинсона мы знаем, что связный граф сильносвязен $\Leftrightarrow$ в этом графе нет мостов. Проверим за $O(V+E)$, что в графе нет мостов. Если это так, то граф можно ориентировать, чтобы он стал сильносвязный. Делать это будем так: зафисксируем произвольную вершину $root$ в графе и запустим из нее обход в глубину, чтобы получить MST. Все ребра графа, входящие в MST ориентируем по направлению от $root$, а остальные ребра ориентируем к $root$. Если в какую-то вершину можно войти, но нельзя выйти, то ребро, по которому мы в нее вошли — это мост, а мы проверили, что в графе нет мостов. Таким образом из любой вершины графа можно попасть в любую другую, то есть граф стал сильносвязный. Так как мы везде использовали только обход в глубину, то итоговая асимптотика $O(V+E)$.

3) Разбить все ребра неориентированный графа на минимальное число путей за $O(V+E)$.

Решение. Допустим, что граф связен, если нет, то будем работать с каждой компонентой связности как с отдельным графом. Дополним граф до Эйлерова с помощью обхода за $O(V+E)$, как-то помечая при этом ребра, которые добавили. Запомним Эйлеров цикл в графе и удалим ребра, которые добавили, чтобы дополнить граф. Полученные «куски», которые останутся от Эйлерова цикла после удаления ребер, и будут искомыми путями. Принимая, что количество количество компонент связности — константа, получаем асимптотику — $O(V+E)$.

4) Для каждой пары вершин в графе найти $w[a,b]$ – такой минимальный вес, что из a в b есть путь по рёбрам веса $\leq w[a,b] O(V^2)$.

Решение. С помощью алгоритма Прима на массиве за $O(V^2)$ найдем MST. Далее запустим по полученном дереву обход в глубину, чтобы найти $w[a,b]$, обновляя результат во время обхода. Так как обход запускаем по дереву, то время обхода — $O(V+E) = O(V) < O(V^2)$. Итоговая асимптотика — $O(V^2)$.

6) Найти в графе цикл минимального среднего веса $V, E \leq 2000, |w_i| \leq 10^9$.

Решение. Для нахождения цикла будет использовать алгоритм Форда-Белмана. С помощью него найдем матрицу $dist[vertex][n]$, в которой будем хранить вес минимального пути длиной n ребер из новой (фиктивной) вершины u в вершину $vertex$. Тогда цикл минимального среднего веса, начинающийся в вершине $vertex$ — это

$$sup_n \frac{dist[vertex][E] - dist[vertex][n]}{E - n},$$

где E — максимальное количество ребер графа. Остальось взять inf по всем $vertex$ и вес искомого цикла будет найден.

Как восстановить сам цикл? Пусть $vertex_0$ и n_0 — те, значения $vertex$ и n, при которых достигается inf и sup соответственно. Тогда если при подсчете матрицы $dist[vertex][n]$ запоминать из какой вершины мы перешли в следующую, то восстановление цикла становится тривиальной задачей.

7) Дан массив чисел. За $O(logn)$ в online обрабатывать запросы: посчитать сумму кубов чисел на отрезке $[L, R]$, прибавить x ко всем числам на отрезке $[L, R]$, получить значение i-го числа.

Решение. Построим дерево отрезков, тогда каждая операция будет занимать $O(logn)$. Полученное дерево будет отличаться от обычного только тем, что в узле будем дополнительно хранить сумму квадратов и сумму кубов на подотрезке. Это требуется для обновления. Допустим мы хотим увеличить значение на подотрезке на величину x, тогда

$$\sum_{i=1}^{n}(x_i + x)^2 = \sum_{i=1}^{n} x_i^2 + x^2 n + 2x \sum_{i=1}^{n} x_i.$$

$$\sum_{i=1}^{n}(x_i + x)^3 = \sum_{i=1}^{n} x_i^3 + x^3 n + 3x \sum_{i=1}^{n} x_i^2 + 3x^2 \sum_{i=1}^{n} x_i.$$

Все остальные операции обрабатываются как для обычного дерева отрезка.

8) Есть массив из нулей и единиц. В online за $O(logn)$ отвечать на запросы: поменять элемент; найти ближайший слева/справа ноль к позиции i.

Решение. Построим дерево отрезков, тогда каждая операция будет занимать $O(logn)$. В узле будем хранить два числа. Первое — левая крайняя позиция нуля на подотрезке. Второе — правая крайняя позиция нуля на подотрезке. Если нуля нет вообще, то оба числа положим ∞.

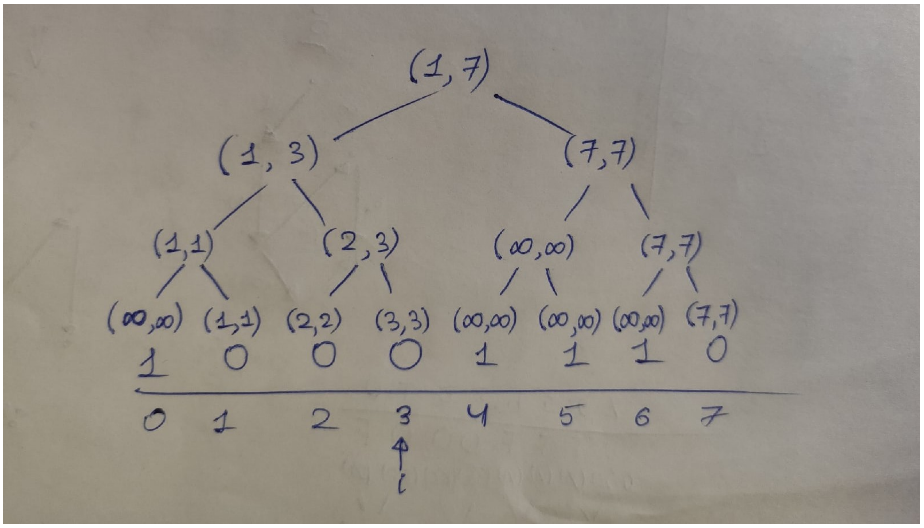

Рис. 1: Пример построенного дерева

Как получить запрос? Допустим хотим найти крайний левый от нас ноль. Поднимаемся в родителя, затем спускаемся в его другого ребенка. Смотрим позицию правого нуля на данном подотрезке. Если позиция $! = \infty$, то нашли крайний левый от нас ноль. Если $== \infty$, то повторяем процесс еще раз, пока не дойдем до корня. Если дошли до корня, а позиция все еще равна ∞, то позиции не существует.

Аналогично ищем крайний правый от нас ноль.

Обновляем элемент аналогично, поднимаясь по дереву.