Практика по алгоритмам #14
Графы, MST, DSU

---

1 Разбор задач из ДЗ #13

2 Новые задачи

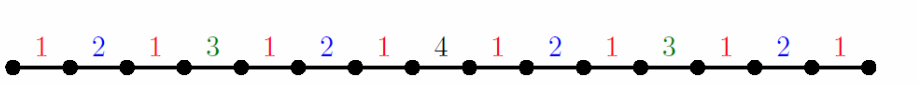
1. Постройте пример для алгоритма Борувки, на котором он делает $\Theta(\log n)$ фаз
2. Постройте пример для алгоритма Борувки, на котором он делает $\Theta(1)$ фаз
3. Рассмотрим алгоритм построения остовного дерева на плоскости: найдем к каждой точке k ближайших, на полученном графе за $\mathcal{O}(nk \log n)$ найдем минимальное остовное дерево. Постройте контрпример.
4. Рассмотрим реализацию СНМ:

```
int get( int v ) { return v == p[v] ? v : (p[v] = get(p[v])); }
void join( int a, int b ) {
  if (rand() & 1)
    swap(a, b);
  p[a] = b;
}
```

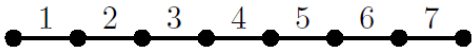
   За сколько такая реализация работает?

3 Разбор задач

1. **Борувка и $\Theta(\log n)$ фаз.**
   Строим граф-путь из 2^k вершин с весами ребер от 1 до k. База: $k = 0$, 1 вершина, пустой граф. Переход: берем два графа-пути из 2^{k-1} вершины и соединяем крайние вершины ребром веса k.

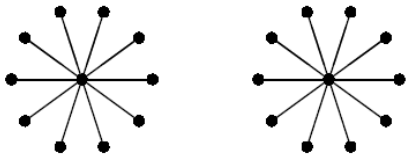


2. **Борувка и $\Theta(1)$ фаз.**
   Строим граф-путь из n вершин. i-я вершины соединена со следующей ребром веса k.

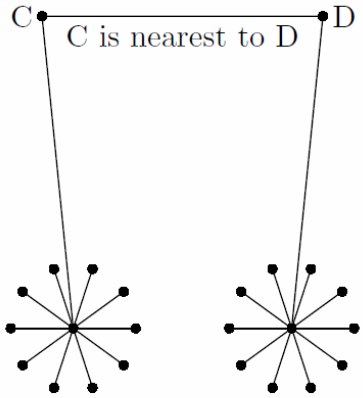


4. **Реализация СНМ с рандомом.**
   Данная реализация в худшем случае работает за $\Theta(n \log n)$, где n – число вершин, а число запросов равно $\Theta(n)$. Длина путей благодаря `if`-у в два раза меньше. Доказательство того, что в худшем случае $\Theta(n \log n)$: рассмотрим тест, на котором сжатие путей работает за $\Theta(n \log n)$, в нём есть момент, когда текущее дерево мы подвешиваем за новый корень. В худшем случае достаточно одной операции подвешивания, в случае, если мы подвешиваем случайное дерево к случайному, матожидание количества подвешиваний равно двум.

3. **Алгоритм для поиска MST "возьмем у каждой вершины k ближайших".**
   Первый контрпример: выбранные ребра могут не образовывать связный граф, для такого эффекта достаточно рассмотреть две "звездочки" на большом расстоянии. Второй контрпример: мы можем добавить лишнее ребро.



Несвязный результат



Лишнее ребро

4 Домашнее задание

4.1 Обязательная часть

1. (5) Напишите код, который получает множество неориентированных ребер и удаляет петли и кратные ребра. Время работы должно быть $\mathcal{O}(n+m)$.

4.2 Дополнительная часть

1. (5) Даны $n \leq 50\,000$ точек на плоскости. Точки порождены равномерным двумерным распределением в $[0,1] \times [0,1]$. Рассмотрим алгоритм построения MST: для каждой точки взять $C_1\sqrt{n}$ ближайших по координате x. Из этих $C_1\sqrt{n}$ выбрать 20 ближайших и на полученном графе из $C_2 n$ ребер найти MST. Оцените вероятность ошибки. Какие константы C_1 и C_2 стоит выбрать?