Minimum Spanning Tree

原案：宮村
解答：播磨、宮村、森
スライド：宮村

問題概要

- 重み付き無向グラフが与えられる。
- 各辺について、その辺が使用できないときの最小全域木(MST)のコストを求めよ。

- 頂点数≦10^5
- 辺数≦2*10^5

駄目な解法

- 愚直に、各辺が使えない場合のMSTを毎回KruskalやPrimで求める。
- MSTを1回計算するのにO(M * log N)とか。
- これをM回繰り返すのでO(M^2 * log N)でTLE。
- 前処理でソートしても焼け石に水。

観察

- とりあえず全ての辺が使えるとしてMSTを構成してみる。
- 構成できなかったら全部-1を出力すればよい。
- このMSTに含まれない辺に関しては、使えなくなってもMSTのコストに全く影響は無い。
- なので、実際に考える必要があるのはMSTに含まれるN-1本の辺だけ。

観察

- MSTに含まれるある辺が使われなくなったらどうすればよい？

  →木が2つに分断されるので、それらをつなぐような辺のうちコストが最小なもので代用すればよい。つなぐような辺が無ければ当然MSTを構成することはできない。

観察

- 根付き木にして考える。
- 各子について部分木から外へ向かう辺集合を再帰的に併合すれば上手くいきそう。

- つまり、集合の併合と最小値の取得を効率的に行えるデータ構造が必要。
- すなわち**meldable heap**の出番!

Meldable Heap

- Meldable Heapって何？
- 値の追加、最小値の取得や削除、二つのHeapの併合(meld)を効率よく処理することができるデータ構造。

- 実装方法はいろいろある。

  Randomized Meldable Heap, Skew Heap, Liftist Heap, Pairing Heap、Fibonacci Heapなどなど。

解法

- 全ての辺が使えるとしてMSTを構成する。
- MSTを適当に根付き木にする。
- 部分木の外へ向かっている辺を集めたminヒープを再帰的にmeldして効率的に最小値を取得し、親への辺が使えないときのMSTのコストを更新していく。
- 途中、あるノードが自分の子孫かどうかを効率的に判定する必要が出てくるが、LCAやオイラーツアー等を使えばよい。

計算量

- 最初にMSTを計算するのはO(M*log N)等。
- ヒープを再帰的にmeldする部分については、meldの操作がO(N)回で、各辺についてpushやpopが1回行われるのでO(M*log M)。
- なので全体としてはO(M * log M)。

注意点

- スタックオーバーフローに注意する。
- 木を走査する部分を再帰で実装した場合は危険。
- Pairing HeapやSkew Heapの主要な操作を再帰で実装している場合にもスタックオーバーフローが起こりうる。再帰を展開したり、適当なタイミングでほぐしたり、木の深さが対数程度におさまる別のヒープを選択して回避しましょう。

補足

- 「meldable heap書けないから解けないや」と諦めてしまうのはもったいない。
- std::priority_queueのような普通のheapを併合するとき、サイズが小さい方のheapの各要素をひとつずつ大きい方のheapに突っ込むようにすると O(m * log^2 m)で実装できる（汎用性高い手法）。
- m = 10^5に対してO(m * log^2 m)を投げるのはTLEが怖いので勇気がいるけど、試してみる価値はあるかも。