

Lemma (Safe edges in Kruskal's algorithm.)

If G = (V, E) is a graph, A is a subset of a minimum spanning tree for G, (u, v) is the lightest edge connecting any distinct connected components of A, then (u, v) is a safe edge for A, that is, $A \cup \{(u, v)\}$ is a subset of a minimum spanning tree.

▲ロト ▲圖ト ▲画ト ▲画ト 三回 - のへで

Proof. Suppose everything in the hypothesis, in particular that A is a subset of some minimum spanning tree T and that u and v are in distinct connected components of A, call them A_u and A_v . Let w_T be the total weight of T, that is, the sum of the weights of all the edges of T. We want to prove that adding (u, v) to A makes something that is still a subset of some minimum spanning tree.

If $(u, v) \in T$, then we're done. Suppose, then, that T does not contain (u, v). Since T is a spanning tree, it means that u and v are connected in T. Pick the lightest edge on the path from u to v that is not in A, call it (x, y). Essentially (x, y) is an edge that was picked instead of (u, v) that contributed to connecting A_u and A_v .

▲□▶ ▲圖▶ ▲目▶ ▲目▶ 目 のへで

Snip out (x, y). This would disconnect T, that is, the graph $T - \{(x, y)\}$ is not a tree, but rather contains two connected components, one with u in it and the other with v in it. Now splice in (u, v). That will reconnect u and v and make it into a tree again. Formally we've made a new spanning tree $(T - \{(x, y)\}) \cup \{(u, v)\}.$

The hypothesis says that (u, v) was the lightest edge connecting distinct components of A. That means $w(u, v) \le w(x, y)$. That in turn means that the total weight of the new spanning tree is also just as good, if not better, than the old one: $w(T - \{(x, y)\}) \cup \{(u, v)\} \le w_T$. Since it ties or beats a (supposed) minimum spanning tree, $(T - \{(x, y)\}) \cup \{(u, v)\}$ must be a minimum spanning tree. Therefore (u, v) is safe.

◆□▶ ◆□▶ ◆三▶ ◆三▶ ○○ ○○○

```
initialize A to \emptyset
make a disjoint-set data structure with each vertex its own set
sort the edges by weight
for each edge (u, v)
if findSet(u) \neq findSet(v)
add (u, v) to A
union(u, v)
```

◆□▶ ◆□▶ ◆臣▶ ◆臣▶ 臣 ○

```
initialize A to \emptyset
initialize all vertices with distance \infty and parent -1
initialize pq with all vertices
while pq is not empty
     u = pq.extractMax()
     if u.p \neq -1
          add (u.p, u) to A
     for each v \in u.adj
          if v \in pq and (u, v).w < v.distBound
               v.p = u
               v.distBound = (u, v).w
               pq.increaseKey(v)
```

◆□▶ ◆□▶ ◆臣▶ ◆臣▶ 臣 の�?