CS 344: Kruskal's Minimum Spanning Tree Algorithm

MST-KRUSKAL ($G = (V, E), w$);

$A := \emptyset$;

for all $v \in V$ **do** Make-Set(v);

Sort edges in E in nondecreasing weight;

for all $(u, v) \in E$, in order of nondecreasing weight **do**

  if Find-Set(u) $\neq$ Find-Set(v) **then** $A := A \cup \{(u, v))\}$, Union (u, v);

Remarks. The procedure Make-Set(v) creates a set out of the vertex v. We represent sets as linked-list. The head of the list will be the name of that set. The procedure Find-Set(u) returns a pointer to the name of the set. The procedure Union(u, v) creates a new set from the two disjoint sets containing u and v.

We will use the *weighted union* rule: always append the smaller list onto the longer.

Theorem. The time complexity of Kruskal's algorithm is $O(|E| \log |E|)$.

Proof. Clearly sorting takes $O(|E| \log |E|)$. Each Make-Set and Find-Set operations take constant time. We claim that the total complexity of Union(u, v) operations is $O(|V| \log |V|)$. To prove this consider a fixed vertex v. Let us determine how many times the representative pointer of v is updated. Let k be this number. Since we use the weighted union rule, the representative pointer of v changes only when the size of current set that it belongs to is smaller than the size of the set to which it is appended. So the size of the new set must at least double. This implies $2^k \leq |V|$, i.e. $k = \lg |V|$. This completes the proof of theorem.