Lecture 11: Kruskal's MST Algorithm

CLRS Chapter 23

Main Topics of This Lecture

- Kruskal's algorithm
  Another, but different, greedy MST algorithm
- Introduction to UNION-FIND data structure.
  Used in Kruskal's algorithm
  Will see implementation in next lecture.

Outline by Example

Idea of Kruskal's Algorithm

Build a forest.

Initially, trees of the forest are the vertices (no edges).

In each step add the cheapest edge that does not create a cycle.

Continue until the forest is a single tree.
(Why is a single tree created?)

This is a *minimum* spanning tree
(we must prove this).

Outline of Kruskal’s Algorithm

Step 0: Set $A = \emptyset$ and $F = E$, the set of all edges.

Step 1: Choose an edge e in F of minimum weight, and check whether adding e to A creates a cycle.

- If “yes”, remove e from F.
- If “no”, move e from F to A.

Step 2: If $F = \emptyset$, stop and output the minimal spanning tree (V, A). Otherwise go to Step 1.

Remark: Will see later, after each step, (V, A) is a subgraph of a MST.

Outline of Kruskal’s Algorithm

Implementation Questions:

- How does algorithm choose edge $e \in F$ with minimum weight?
- How does algorithm check whether adding e to A creates a cycle?

How to Choose the Edge of Least Weight

Question:
How does algorithm choose edge $e \in F$ with minimum weight?

Answer: Start by sorting edges in E in order of increasing weight.
Walk through the edges in this order.
(Once edge e causes a cycle it will always cause a cycle so it can be thrown away.)

How to Check for Cycles

Observation: At each step of the outlined algorithm, (V, A) is acyclic so it is a forest.

If u and v are in the same tree, then adding edge $\{u, v\}$ to A creates a cycle.

If u and v are not in the same tree, then adding edge $\{u, v\}$ to A does not create a cycle.

Question: How to test whether u and v are in the same tree?

High-Level Answer: Use a disjoint-set data structure
Vertices in a tree are considered to be in same set.
Test if Find-Set(u) = Find-Set(v)?

Low -Level Answer:
The UNION-FIND data structure implements this:

The UNION-FIND Data Structure

UNION-FIND supports three operations on collections of **disjoint sets**: Let n be the size of the universe.

Create-Set(u): $O(1)$
Create a set containing the single element u.

Find-Set(u): $O(\log n)$
Find the set containing the element u.

Union(u, v): $O(\log n)$
Merge the sets respectively containing u and v into a common set.

For now we treat UNION-FIND as a black box.
Will see implementation in next lecture.

Kruskal's Algorithm: the Details

```
Sort $E$ in increasing order by weight $w$;        $O(|E| \log |E|)$
/* After sorting $E = \langle \{u_1, v_1\}, \{u_2, v_2\}, \ldots, \{u_{|E|}, v_{|E|}\} \rangle$ */

$A = \{\,\}$;
for (each $u$ in $V$) CREATE-SET($u$);             $O(|V|)$

for $i$ from 1 to $|E|$ do                          $O(|E| \log |E|)$
    if (FIND-SET($u_i$) != FIND-SET($v_i$) )
    {   add $\{u_i, v_i\}$ to $A$;
        UNION($u_i, v_i$);
    }
return(A);
```

Remark: With a proper implementation of UNION-FIND, Kruskal's algorithm has running time $O(|E| \log |E|)$.

Correctness of Kruskal's Algorithm

Lemma:
Let (V, A) be a subgraph (part) of a MST of $G = (V, E)$, and let $e = \{u, v\} \in E \setminus A$ be an edge such that

(1) $(V, A \cup \{e\})$ has no cycle;
(2) e has minimum weight among all edges in $E \setminus A$ such that (1) is satisfied.

Then $(V, A \cup \{e\})$ is a subgraph of some *MST* containing (V, A).

Corollary: Kruskal's algorithm produces a MST.

Proof of the Lemma: The Idea

Idea of Proof: Let T be any MST with (V, A) as a subgraph. Then we prove that

- either $e \in T$ so $(V, A \cup \{e\})$ is a subgraph of MST T and lemma is correct or
- if $e \notin T$ there is edge $e' \in T - A$ such that $W(e) = W(e')$ and $T' = T \cup \{e\} - \{e'\}$ is a tree.
  Since $W(T') = W(T)$ this implies T' is a MST so $(V, A \cup \{e\})$ is a subgraph of MST T' so lemma is correct.

Proof of the Lemma: The idea

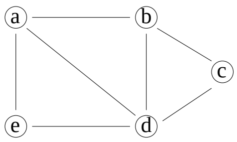
Correctness of Kruskal's Algorithm – Continued

Proof: Let T be any MST with (V, A) as a subgraph.
If $e \in T$, we are done.

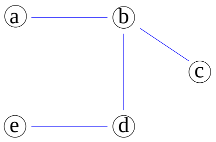
Suppose that $e = \{u, v\} \notin T$.
There is a unique path from u to v in the MST T,
which contains at least one edge $e' \in E \setminus A$.
$e' \neq e$ (because $e \notin T$ but $e' \in T$).
$(V, A \cup \{e'\})$ has no cycles
(because $(V, A \cup \{e'\})$ is included in T.)
$W(e) \leq W(e')$ (because both edges in $E \setminus A$
and assumption (2) in previous page).
Consider the new tree $T' = (T \cup \{e\}) \setminus \{e'\}$.
If $W(e) = W(e')$, then T' is another minimum
spanning tree containing $(V, A \cup \{e\})$.
If $W(e) < W(e')$, then
$W(T') - W(T) = W(e) - W(e') < 0$.
Contradiction.

Understanding the Proof of Lemma

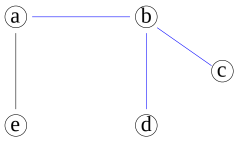
Understanding the Proof of Lemma



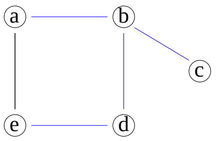
Original graph G



Spanning tree T



New spanning tree T’
after deleting {e, d}



T U {a, e}