Kruskal's MST Algorithm

CLRS Chapter 23, DPV Chapter 5

Version of November 5, 2014

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.

Idea of Kruskal's Algorithm

Build a forest.

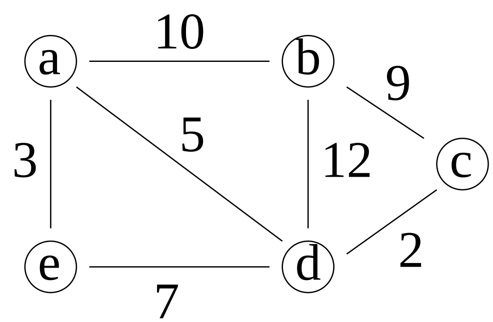
Initially, trees of the forest are the vertices (no edges).

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

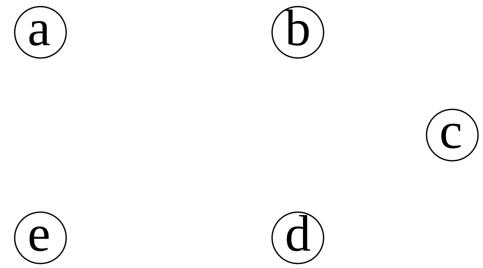
Continue until the forest is a single tree.
(Why is a single tree created?)

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

Outline by Example



original graph



forest → MST

E

edge	weight
{d, c}	2
{a, e}	3
{a, d}	5
{e, d}	7
{b, c}	9
{a, b}	10
{b, d}	12

Forest (V, A)

A={ }

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; $\quad O(|E| \log |E|)$

/* 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); $\quad O(|V|)$

for i from 1 to $|E|$ do $\quad O(|E| \log |E|)$

    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

Sort the graph edges in nondecreasing order so that

$$w(e_1) \le w(e_2) \le \cdots \le w(e_m)$$

Let A_i be A in Kruskal's algorithm after processing e_i.

Set $A_0 = \emptyset$. Then

If e_{i+1} forms a cycle with A_i, $\Rightarrow$ $A_{i+1} = A_i$
If e_{i+1} doesn't form a cycle with A_i, $\Rightarrow$ $A_{i+1} = A_i \cup \{e_{i+1}\}$

We will prove that, $\forall i$, $\exists$ MST T_i such that $A_i \subseteq T_i$.

In particular, this means that

$$A_0 \subseteq A_1 \cdots \subseteq A_m \subseteq T_m$$

which implies (why?) Kruskal's algorithm produces MST T_m.

Correctness of Kruskal's Algorithm

Need to prove that $\forall i$, $\exists$ MST T_i such that $A_i \subseteq T_i$.
Proof will be by induction on i

Obviously true for base $i = 0$. If $i \geq 0$,
(a) If e_{i+1} forms a cycle with A_i, $\Rightarrow A_{i+1} = A_i$
(b) If e_{i+1} doesn't form a cycle with A_i, $\Rightarrow A_{i+1} = A_i \cup \{e_{i+1}\}$

Claim is true for case (a).
To prove for case (b)
note that T_i is forest on n nodes.
Let C_1, C_2, C_K, be connected components (trees) of forest.
Let $V_1, V_2, \ldots, V_k$, be their vertices.

Without loss of generality,
let V_1 contain one of the endpoints of e_{i+1}.
Note that the other endpoint is *not* in V_1.

Correctness of Kruskal's Algorithm

Recall Lemma proved previously

- Let $G = (V, E)$ be a connected, undirected graph with a real-valued weight function w defined on E
- A be a subset of E that is included in some MST for G.

Let

- $(S, V - S)$ be any cut of G that respects A
- e be a light edge crossing the cut $(S, V - S)$

Then, $A \cup \{e\}$ is included in some MST for G.

Now plug in the information from previous slide.

Let $S = V_1$, $A = A_i$ and $e = e_{i+1}$
Induction hypothesis is that A_i is in some MST.

Since V_1 is CC of A_i, $(V_1, V - V_1)$ respects A_i.

Easy to see (how?) that e_{i+1} is a light edge crossing the cut.

So, $A_{i+1} = A_i \cup \{e_{i+1}\}$ is included in some MST for G,
and laim is proven.

Odds and Ends

On previous slide we stated that it's easy to see that e_{i+1} is a light edge crossing the cut.

Suppose that this was not true
Then $\exists$ some e_j with $w(e_j) < w(e_{i+1})$ that crosses the cut.
By definition, if edge crosses the cut, its endpoints are in different connected components of T_i (and therefore A_i) so it can't form a cycle with A_i.

$w(e_j) < w(e_{i+1})$ so $j < i + 1$ and e_j is processed *before* e_{i+1}. Since $A_{j-1} \subseteq A_i$ and e_j doesn't form a cycle with A_i, e_j also doesn't form a cycle with A_{j-1}.

Thus, e_j would have been added to A_j by Kruskal's algorithm! But this contradicts fact that e_j *can not* be in A_i since it connects two items that are not connected in A_i.