CSC 373 H 1 Y — Summer 2006

University of Toronto — St. George Campus

Lecture Summary for Week 2

Graph theory terminologies

Graphs, paths, cycles, acyclic graphs, tree, connectivity, spanning trees, weighted graphs, minimum spanning trees.

Fact For a graph G, any two conditions of the following imply the remaining one:

1. G is connected.
2. G is acyclic.
3. G has $(n-1)$ edges.

In particular, any tree with n nodes has exactly $(n-1)$ edges.

Algorithms for finding a spanning tree

Note that examining each of the $(n-1)$-element subsets of the edges takes exponential time, because there are potentially exponentially many such subsets.

Approach 1

- Start with no edge: $E' \leftarrow \varnothing$.
- Keep adding edges if they do not creat cycle.

Approach 2

- Start with all edges in G: $E' \leftarrow E$.
- Find a cycle, remove one edge from it. Repeat.

Kruskal's algorithm and Prim's algorithm more or less follow the first approach, and the so-called Reverse-Delete Algorithm follows the second, with the additional concern for the weights.

The Minimum Spanning Tree Problem

"Given a weighted connected graph. Find a spanning tree of minimum weight."

Kruskal's Algorithm

- Sort the edges in increasing order of weight:

$$w(e_1) \leq w(e_2) \leq \ldots \leq w(e_m)$$

- $E' \leftarrow \varnothing$
- For $i = 1$ to n do
- If $E' \cup \{e_i\}$ does not contain a cycle then $E' \leftarrow E' \cup \{e_i\}$

Runtime? $\Theta(m \log m)$ for sorting. The main loop leaves room for different implementations. With careful chosen data structures, it can be done in time $\Theta(m \log n)$. (Note that since $m = \mathcal{O}(n^2)$, $\log m = \mathcal{O}(\log n)$. See Section 4.6 in the text for a discussion.

Correctness of Kruskal's Algorithm

It turns out that Kruskal's algorithm is an instance of the generic algorithm for MST given below. So we will actually prove the correctness of this generic algorithm, and then show that Kruskal's algorithm is an instance of it.

Generic Algorithm for MST

In the following algorithm, B and R are sets containing the "blue" and "red" edges, respectively.

1. $B \leftarrow \varnothing, R \leftarrow \varnothing$: Initially all edges are uncoloured.
2. Repeat until all edges are coloured:
3. Apply either the *Blue Rule* or the *Red Rule.*
4. Return B.

The Blue Rule

- Select a cut (a partition of vertices into X and $V \setminus X$) so that no blue edge crosses.
- Select a minimum-weight uncoloured edge e across that cut. Colour it blue:

$$B \leftarrow B \cup \{e\}$$

The Red Rule

- Select a simple cycle C containing no red edge.
- Select a maximum-weight uncoloured edge e in that cycle. Colour it red:

$$R \leftarrow R \cup \{e\}$$

Termination and Correctness

Does the Generic algorithm always terminate ? Does it produce correct answer ?

Theorem: The Generic algorithm always terminates.

Proof Sketch To prove the Theorem, we need to show that as long as there remains an uncoloured edge, we can apply either the Blue Rule or the Red Rule.

So consider an uncoloured edge $e = (u, v)$. Consider the set X of all vertices that are connected to u by a path of blue edges. There are two cases, depending on whether v is in this set or not:

Case 1: u and v are connected by a blue path (i.e., $v \in X$). This path together with the edge e form a cycle with no red edge in it. Thus we can apply the Red Rule.

Case 2: u and v are not connected by a blue path (i.e., $v \notin X$). Then $v \in V \setminus X$ and thus $V \setminus X$ is non-empty. By the definition of X, there is no blue edge crossing X and $V \setminus X$. We can apply the Blue Rule with this cut. □

Theorem The Generic algorithm always return a MST.

Proof We say that a pair (B, R) is "promising" if there is a MST T so that

$$B \subseteq T \qquad \text{and} \qquad T \cap R = \varnothing$$

We show that at any time during the Generic algorithm, the pair (B, R) is always promising. Then when the algorithm terminates, B must form a MST.

When the algorithm starts, $B = R = \varnothing$, so they are promising. (Take any MST T, then both $\varnothing \subseteq T$ and $T \cap \varnothing = \varnothing$.)

Now suppose that (B', R') is obtained from (B, R) after one application of either the Blue Rule or the Red Rule. Suppose that (B, R) is promising, we show that (B', R') is also promising.

Consider two cases:

Case 1: (B', R') is obtained from (B, R) by the Blue Rule. Then there is a cut X with no edge in B crossing $X, V \setminus X$, so that $B' = B \cup \{e\}$ where e is a minimum-weight edge crossing the cut. Also, $R' = R$. Let T be a MST such that

$$B \subseteq T, \qquad T \cap R = \varnothing$$

We show that there is a MST T' so that

$$B' \subseteq T', \qquad T' \cap R' = \varnothing$$

Consider two subcases:

Subcase 1a: $e \in T$. Then we can take $T' = T$. Since $B \subseteq T$ and $e \in T$, we have $B \cup \{e\} \subseteq T$, i.e., $B' \subseteq T'$. Also, $R' = R$, so $T' \cap R' = T \cap R$, so $T' \cap R' = \varnothing$.

Subcase 1b: $e \notin T$. We show how to obtain T' from T by exchanging an edge e' in $T \setminus B$ for e without increasing the total weight.

Adding e to the tree T creates a cycle C. This cycle has an edge e that crosses the cut $X, V \setminus X$; therefore it must have another edge e' crossing the cut. Since $X, V - X$ is a cut used in the Blue Rule, at this moment e' is not coloured blue (i.e. $e' \notin B$) and also $w(e') \geq w(e)$. Let $T' = (T - \{e'\}) \cup \{e\}$. Now T' has $(n-1)$ edges (the same number of edges as T), and is connected (we only break a cycle of $T \cup \{e\}$). So by the Fact at the beginning of class, T' is a spanning tree of G. Also, $w(T') \leq w(T)$, so T' is a MST of G.

Since $e' \notin B$, it follows that $B' \subseteq T'$. Also $T' \cap R' = \varnothing$, because $e \notin R$.

Case 2: (B', R') is obtained from (B, R) by the Red Rule: **Exercise**. □

Exercise Complete the proof above by carrying out the proof for Case 2.

Theorem The Kruskal's algorithm is an instance of the Generic algorithm.

Proof Define the blue colour as "accepted", and red colour as "rejected". In other words, at the i-th iteration, define B_i to be the set of edges among $e_1, \ldots, e_i$ that are accepted by Kruskal's algorithm, and $R_i = \{e_1, \ldots, e_i\} - B_i$. (The "uncoloured" edges are $e_{i+1}, \ldots, e_m$.) The following two Exercises are the remaining steps of the proof. □

Exercise Suppose that during the i-th iteration, Kruskal's algorithm accepts e_i. Find a cut $X, V - X$ such that no edges from B_{i-1} cross the cut. Show also that e_i is a minimum weight edge that crosses the cut.

Exercise Now suppose that during the i-th iteration, Kruskal's algorithm rejects e_i. Find a cycle C that contains e_i but no edges in R_{i-1}, such that e_i is a maximum weight edge in the cycle.