Recitation 10 — More Graph Contraction

Parallel and Sequential Data Structures and Algorithms, 15-210 (Spring 2013)

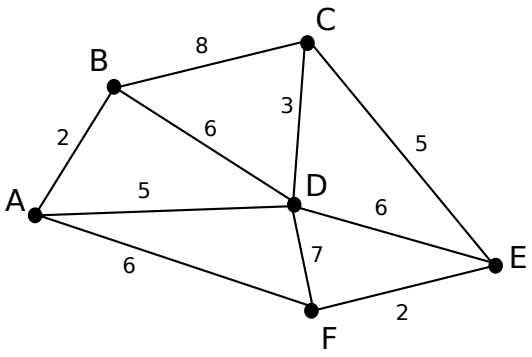
March 27, 2013

Today's Agenda:

- Prim's algorithm
- Boruvka's algorithm

1 Prim's Algorithm

Prim's algorithm is an algorithm for computing a minimum spanning tree (MST) in a connected graph.

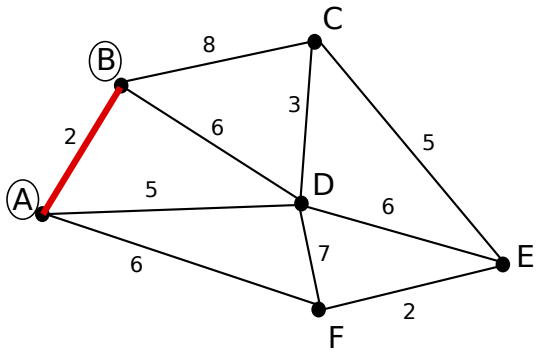


Algorithm:

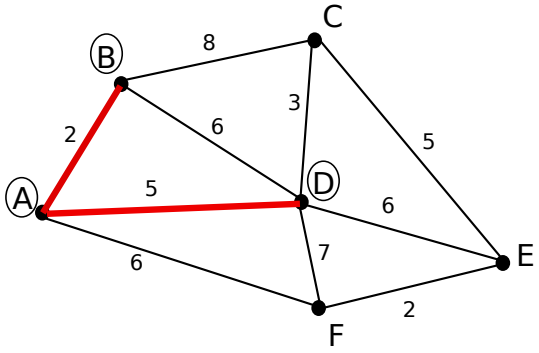
1. Choose any starting vertex. Look at all edges connecting to the vertex and choose one of the ones with the lowest weight and add this to the tree.
2. Look at all edges connected to the tree that do not have both vertices in the tree. Choose the one with the lowest weight and add it to the tree.
3. Repeat step 2 until all vertices are in the tree.

Steps:

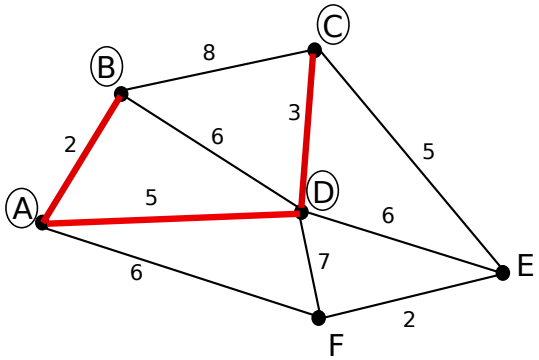
- Choose vertex A. Choose edge with lowest weight: (A,B).



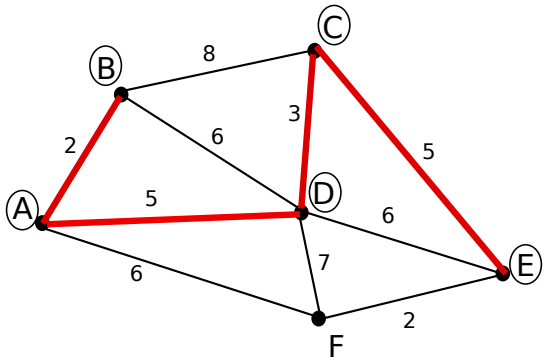
- Look at all edges connected to A and B: (B,C), (B,D), (A,D), (A,F). Choose the one with minimum weight and add it to the tree: (A,D).



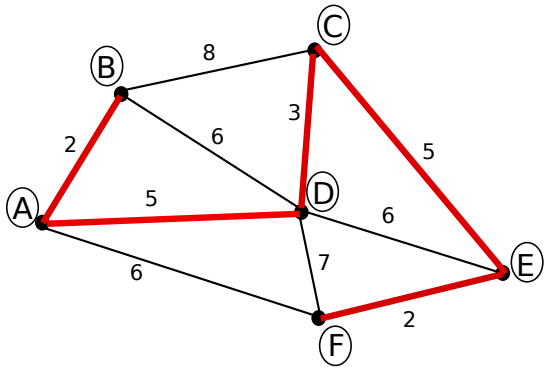
- Look at all edges connected to the tree: (B,C), (D,C), (D,E), (D,F), (A,F). We don't need to consider edge (B,D) because both B and D are in the tree already. We choose edge (D,C).



- Look at all edges connected to A, B, C and D. We still have to connect E and F to the tree. So we look at the edges connected to those and choose the one with the lowest weight: (C,E).



- There is only one vertex F to add before we have a connected minimum spanning tree. We choose edge (E,F) and add that one to the tree.



The tree is now connected and spans all vertices in the graph.

Q: What is the rule about MSTs and cut edges that applies to Prim's algorithm?

A: The light edge rule:

The following theorem states that the lightest edge across a cut is in the MST of G:

Theorem 1.1. *Let $G = (V, E, w)$ be a connected undirected weighted graph with distinct edge weights. For any nonempty $U \subsetneq V$, the minimum weight edge e between U and $V \setminus U$ is in the minimum spanning tree of G.*

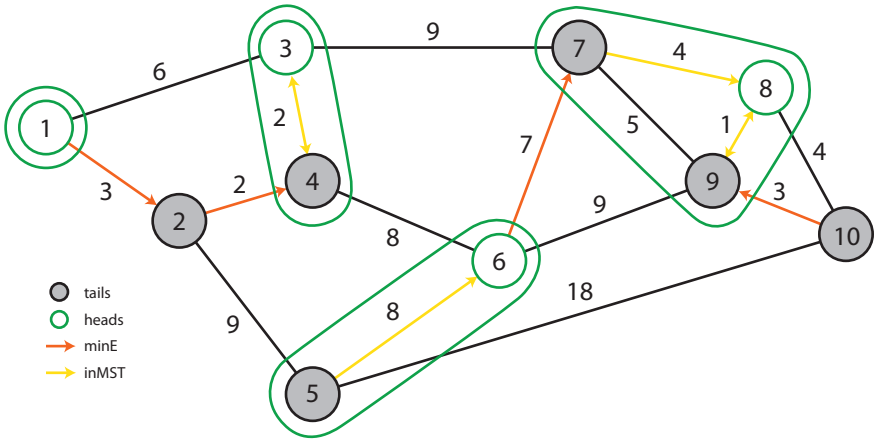
Q: What is the work and span of Prim's algorithm?

A: $O(m \log n)$ for both work and span (same as Dijkstra's).

2 MST

In lecture 17 we presented Boruvka's algorithm, a parallel algorithm for finding a MST. The idea is similar to star contraction, but instead of contracting any of the edges, we only contract edges which are minimum weight from each vertex. Why does this work? Recall the Light Edge Rule / Cut Property from lecture. Refer to the notes for lecture 17 for the pseudocode of this algorithm.

Let's go over an example:



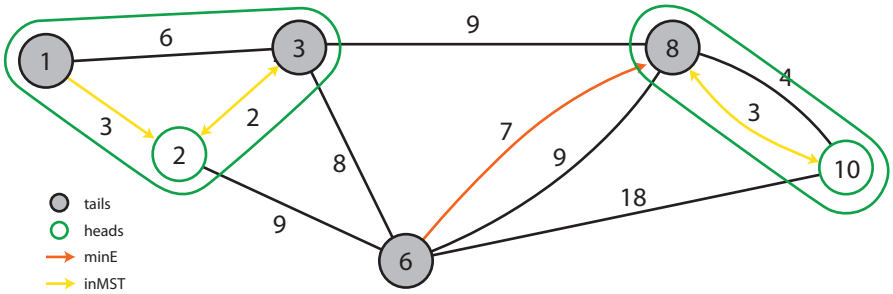
In the first round of the algorithm, we have the following flips:

1	2	3	4	5	6	7	8	9	10
H	T	H	T	T	H	T	H	T	T

Notice that vertices 3 and 4 are contracted, but 1 and 2 are not. Why?

A key point to note here is that in our version of the algorithm, we only consider the minimum *out*-edges from every vertex. That is to say, even though the input graph is undirected, we only pick an edge to be in our MST if it goes from tails to heads. This works because we represent undirectedness by having an edge in both directions.

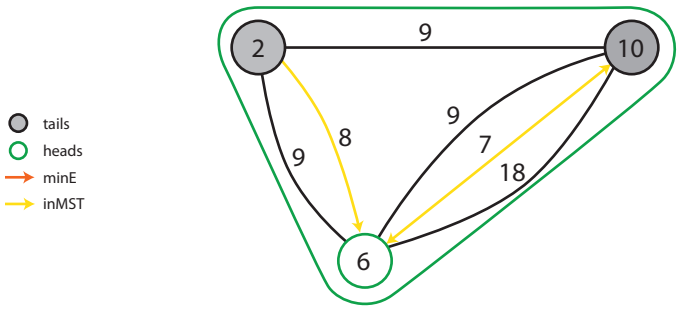
We get the following contracted graph in the next round:



Here is the sequence of flips generated for the second round:

1	2	3	4	5	6	7	8	9	10
T	H	T			T		T		H

We will generate another sequence of 10 flips here, but we only look at the ones generated for the vertices which remain in our contracted graph. This gives us:



with the following sequence of flips (ignoring vertices not in our graph):

1	2	3	4	5	6	7	8	9	10
	T				H				T

Of course, the flips seem a little fortuitous, allowing us to contract this graph in 3 rounds. That's because they were made up for this example. In general, it's not unreasonable to have a round of flips which results in no contractions at all. This is where expectation comes in.

Recall from lecture that each vertex has a minimum edge out which contracts with probability $1/4$. By linearity of expectations, $n/4$ vertices in expectation will be removed in each round.