MATH 409 LECTURE 7
PRIM'S ALGORITHM FOR MINIMUM SPANNING TREES

REKHA THOMAS

Prim's Algorithm (Jarnik (1930), Dijkstra (1959), Prim (1957))

Input: A connected graph G with edge costs $c_e \in \mathbb{R}$, $\forall\ e \in E(G)$.
Output: A MST T of G.

(1) Choose $v \in V(G)$. Set $T = (\{v\}, \emptyset)$.
(2) While $V(T) \neq V(G)$ do
Choose an edge $e \in \delta(V(T))$ of minimum cost.
Set $T = T + e$.

figure to be added

Theorem 1. *The graph T found by Prim's algorithm is an MST of G.*

Proof. We need to check the following facts about T:
1. T is a spanning subgraph of G,
2. T is a tree,
3. T is a minimum spanning tree of G.

1. T is a spanning subgraph of G since the "while" loop in the algorithm does not quit until $V(T) = V(G)$.
2. T is a tree at all intermediate steps of the algorithm. In Step 1, it is clearly a tree. During each step of the while loop, we add an edge $e \in \delta(V(T))$ to the current tree T which does not create any cycles since one end point of e is always outside $V(T)$.

Date: April 16, 2010.

3. T is a minimum spanning tree of G by Theorem 7 (3) of Lecture 6 which says that if $\forall\ e \in E(T)$, e is a minimum cost edge of $\delta(V(C))$, where C is a connected component of $T\backslash e$, then T is a MST. Let us make sure that this property is satisfied by the T of Prim's algorithm. Pick $e \in T$ and consider $T\backslash e$ and its two connected components C and C'. The edge e was added to T at some stage of the while loop. At this stage, we had a partial tree which is one entire connected component of $T\backslash e$ — say C. The algorithm chooses e to be the min cost edge in $\delta(V(C))$. Since e was an arbitrary edge of T, Theorem 3 (3) is true for all edges $e \in T$. □

We now analyze the complexity of Prim's algorithm. We will assume that $G = (V, E)$ has $O(m)$ nodes and $O(n^2)$ vertices when $|V| = n$ and $|E| = m$. Why is this reasonable ? Remember we are doing worst-case complexity analysis. If $|V| = n$ then the maximum number of edges possible in G is $\binom{n}{2} = \frac{n!}{(2!)(n-2)!} = \frac{n(n-1)}{2} = O(n^2)$. If $|E| = m$ then G has at most $2m = O(m)$ vertices as each edge has two vertices as endpoints. Note that if G had parallel edges or loops we could delete them at the start and so we can assume that G is a simple graph. When deleting parallel edges, we would retain one of minimum cost.

Theorem 2. *Prim's algorithm has a running time of $O(n^2)$ where $n = |V(G)|$.*

Proof. We use the following data structure: At each stage of the algorithm, maintain for all $v \notin V(T)$, the cheapest edge e_v from v to a vertex in T. These will be the candidates for the new e in each loop of the algorithm. We analyze the algorithm in steps.

(i) At the start of the algorithm we need to initialize this data structure. How do we do this? We have picked a vertex $v = v_0$ to be in the initial tree T. Therefore for each vertex $v \neq v_0$, either there is an edge in G between v and v_0 or not. Since G is simple, for each $v \neq v_0$, such that $\{v_0, v\} \in E(G)$ there is a unique such edge. We maintain these edges. This takes $O(n)$ work since there are at most n neighbors of v_0.

(ii) At each step of the while loop we need to choose an edge e to add to the current tree. This takes $O(n)$ work since we are maintaining only the cheapest edge to the current tree from a vertex outside the tree which means that we are maintaining at most $O(|V| = n)$ edges. It takes $O(n)$ work to look through these edges and pick an e.

(iii) After we have picked an e in step k of the while loop we need to update our special data structure for step $k + 1$. How much work is that?

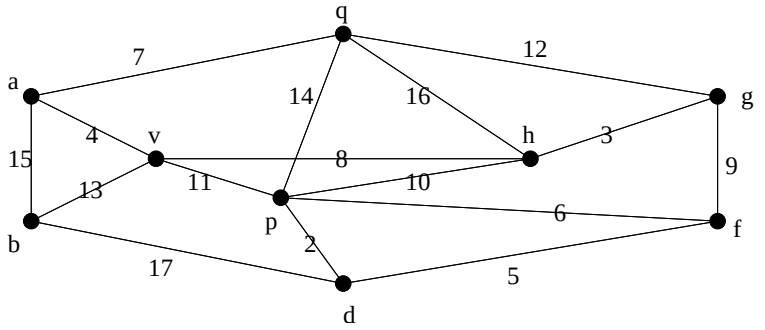

FIGURE 1. Figure for Exercise 3

Suppose we just added w to $V(T)$. The cheapest edge e_v from a vertex $v \notin V(T)$ can change only if $vw \in E$. Therefore, we scan through all edges $wv \in E(G)$ such that $v \notin V(T)$ and update e_v if c_{wv} is strictly smaller than the old e_v that was maintained in the data structure. This takes $O(n)$ time as w has at most $O(n)$ neighbors and we need to do at most $O(n)$ comparisons.

Final tally. There are $n-1$ iterations in Step (2) of Prim's algorithm. Therefore the total work taken is $O(n) + (n-1)(O(n) + O(n)) = O(n^2)$. □

The following exercises are taken from Section 2.1 of [1].

Exercise 3. Find a MST in the graph in Figure by using Prim's algorithm. The starting vertex v is indicated.

Exercise 4. Show that the following algorithm finds an MST of a connected graph G. Begin with $H = G$. At each step, find (if one exists) a maximum cost edge e such that $H \backslash e$ is connected, and delete e from H. (Try this algorithm on the graphs we have worked with so far to check if it works.)

REFERENCES

[1] W. Cook, W. Cunningham, W. Pulleyblank, and A. Schrijver. *Combinatorial Optimization*. Wiley-Interscience Series in Discrete Mathematics, 1998.