Algorithms Lab 10

In lab

Prim's algorithm. In class we talked about the high-level idea of Prim's algorithm:

- Start from an arbitrary vertex v.
- Consider all edges with exactly one endpoint in the current tree and pick the one of minimum weight; add it to T. Repeat.

To implement this, the main question is how to store the edges with one endpoint in T so that we can select the minimum weight edge fast. Do you remember a data structure that can give us the smallest/largest element fast? A priority queue! Main ideas:

- The priority queue will store all the vertices that are not in T yet;
- A vertex v in the PQ has priority equal to the weight of the minimum edge that connects v with a vertex already in T. The other endpoint of this edge is stored in $visit(v)$.
- Essentially the priority queue stores all edges that *cross* the cut between the vertices in T and the vertices in $V - T$ (If a vertex not in T is connected by several edges to vertices in T, the pq will store only one of these edges, the smallest).
- Initially T is empty and PQ contains all vertices in G with priority ∞, except an arbitrary vertex v which has priority 0.

The textbook implementation is the following:

PRIM(r)
For each vertex $u \in V, u \neq v$,: $\text{INSERT}(PQ, u, \infty)$
$\text{INSERT}(PQ, v, 0)$, set $visit(v) = NULL$
WHILE PQ not empty DO

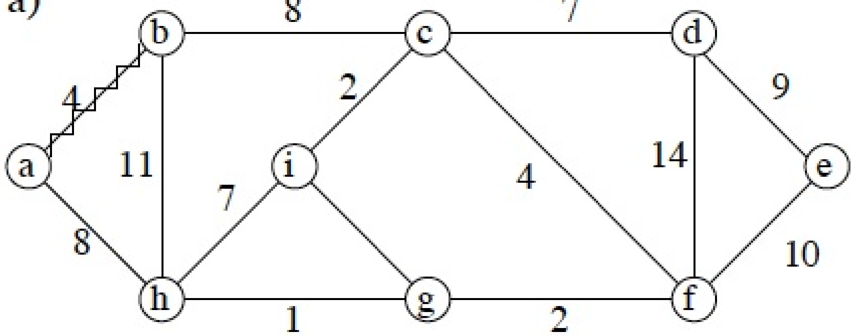
    $u = \text{DELETE-MIN}(PQ)$

    For each $(u, v) \in E$ DO

        IF $v \in PQ$ and $w(u, v) < \text{key}(v)$:
            visit$[v] = u$
            $\text{DECREASE-KEY}(PQ, v, w(u, v))$

Output edges $(u, visit(u))$ as the MST

a)



1. Show how the algorithm runs on an example graph.

2. What is the role of checking whether $v \in PQ$?

3. How many INSERT operations are performed by the algorithm?

4. How many DELETE-MIN operations are performed by the algorithm?

5. How many DECREASE-KEY operations are performed by the algorithm?

6. Assuming the priority is implemented as a heap, what is the complexity of the algorithm?

Homework

1. (CLRS 23.1-1) Show that a minimum-weight edge in G belongs to some MST of G.

2. (CLRS 24.2-4) Suppose that all edge weights in a graph are integers in the range from 1 to $|V|$. How can you take advantage of this in Kruskal's algorithm, and how fast can you make it run? What if the edge weights are integers from 1 to W for some constant W?