CSC263 Fall 2017 - Week 11

University of Toronto

CLRS 23.1-1

Let *(u, v)* be a minimum-weight edge in a connected graph *G*. Show that *(u, v)* belongs to some minimum spanning tree of *G*.

Solution:

Suppose that A is an empty set of edges. Then, make any cut that has *(u, v)* crossing it. Then, since that edge is of minimal weight, we have that *(u, v)* is a light edge of that cut, and so it is safe to add. Since we add it, then, once we finish constructing the tree, we have that *(u, v)* is contained in a minimum spanning tree.

CLRS 23.1-3

Show that if an edge *(u, v)* is contained in some minimum spanning tree, then it is a light edge crossing some cut of the graph.

Solution:

Let T_0 and T_1 be the two trees that are obtained by removing edge *(u, v)* from a MST. Suppose that V_0 and V_1 are the vertices of T_0 and T_1 respectively. Consider the cut which separates V_0 from V_1. Suppose to a contradiction that there is some edge that has weight less than that of *(u, v)* in this cut. Then, we could construct a minimum spanning tree of the whole graph by adding that edge to T_1 U T_0. This would result in a minimum spanning tree that has weight less than the original minimum spanning tree that contained *(u, v)*.

CLRS 23.1-5

Let e be a maximum-weight edge on some cycle of connected graph $G = (V, E)$. Prove that there is a minimum spanning tree of $G' = (V, E - \{e\})$ that is also a minimum spanning tree of G. That is, there is a minimum spanning tree of G that does not include e.

Solution:

Let A be any cut that causes some vertices in the cycle on one side of the cut, and some vertices in the cycle on the other. For any of these cuts, we know that the edge e is not a light edge for this cut. Since all the other cuts wont have the edge e crossing it, we won't have that the edge is light for any of those cuts either. This means that we have that e is not safe.

CLRS 23.2-2

Suppose that we represent the graph $G = (V, E)$ as an adjacency matrix. Give a simple implementation of Prim's algorithm for this case that runs in $O(V^2)$ time.

Solution:

At each step of the algorithm we will add an edge from a vertex in the tree created so far to a vertex not in the tree, such that this edge has minimum weight. Thus, it will be useful to know, for each vertex not in the tree, the edge from that vertex to some vertex in the tree of minimal weight. We will store this information in an array A, where $A[u] = (v, w)$ if w is the weight of (u, v) and is minimal among the weights of edges from u to some vertex v in the tree built so far. We'll use $A[u].1$ to access v and $A[u].2$ to access w.

PRIM-ADJ(G, w, r):
Initialize A so that every entry is (NIL, ∞)
$T = \{r\}$
for i = 1 to V do
    if $Adj[r, i] \neq 0$ then
        $A[i] = (r, w(r, i))$
    end if
end for
for each $u \in V - T$ do
    $k = \min_i A[i].2$
    $T = T \cup \{k\}$
    $k.\pi = A[k].1$
    for i = 1 to V do
        if $Adj[k, i] \neq 0$ and $Adj[k, i] < A[i].2$ then
            $A[i] = (k, Adj[k, i])$
        end if
    end for
end for