Wyatt Kormick

Assignment 5 CSCI4041
DUE: **Monday**, Nov 21 10:00pm (+1hr55m grace period)   Fall 2016

1. The transpose of a directed graph G = (V, E) is the graph G^T = (V, E^T), where E^T = { (v,u) in VxV : (u,v) in E } Thus, G^T is G with all its edges reversed. Write efficient algorithms for computing G^T from G, for both the adjacency-list and adjacency-matrix representations of G. Analyze the running times of your algorithms.

```
//A is the original adjacency list
ALIST-TRANSPOSE(A)
    n = |V|
    B = new AdjList of length n
    for i = 1 to n
        B[i] = nil
    for i = 1 to n
        while A[i] != nil
            Append(i, B[A[i]])
            A[i] = next A[i] node
    return B

//Where A is a square n x m matrix (m=n)
//A is formatted as an array stored in row-major order
AMAT-TRANSPOSE(A)
    N=length(A)
    for n = 1 to N-1
        for m = n+1 to N
            swap(A[n][m], A[m][n])
    return A
```

ALIST-TRANSPOSE has a while loop nested inside a for loop. The for loop runs |V| times and the while loop runs the length of the linked list at the index of the list. O(V+x)

AMAT-TRANSPOSE has a for loop inside of a for loop. The outer runs N-1 times, the inner runs N-(n+1) times. O(VE)

2. Let G = (V ,E) be a directed graph in which each vertex u in V is labeled with a unique integer L(u) from the set { 1, 2, …, |V| }. For each vertex u in V, let R(u) = { v in V : u has path to v } be the set of vertices that are reachable from u. Define min(u) to be the vertex in R(u) whose label is minimum, i.e., min(u) is the vertex such that L(v) = min { L(w) : w in R(u)}. Give an O(V+E)-time algorithm that computes min(u) for all vertices u in V.

3. 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^0 = (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.
   M is a minimum spanning tree of G^0
   Since G^0 and G contain the same vertices, M is also a spanning tree of G
   Suppose M isn't an MST of G
   Then there must be a M' that is a MST containing edge e with weight less than M
   e is a maximum-weight edge on some cycle
   M'' can be made by removing edge e and adding some edge (a,b)
   M'' has to be a tree with a weight less than M' because it contains some edge other than e
   But then M' isn't a minimum spanning tree, so there is a contradiction and M is a minimum spanning tree of G

4. Prove that a graph has a unique minimum spanning tree if, for every cut of the graph, there is a unique light edge crossing the cut. Show that the converse is not true by giving a counterexample.
   Suppose there is not a unique MST, then there are two (at least)
      Let us call these two M and M'
      M and M' have at least one edge that is different (a,b)
   (a,b) crosses a cut in G. Since M' does not contain (a,b), it must contain another edge (c,d) that crosses the same cut.
   Suppose (a,b) is the light edge crossing the cut (since it must be either (a,b) or (c,d))
   M' does not contain (a,b), there must be some path in M' from a to b
      This path must cross the cut somewhere
      Suppose it is at edge (c,d)
   Since (a,b) is a unique minimum edge, the weight of (c,d) is more than (a,b)
   Replace (c,d) in T' with (a,b) to get a smaller weight tree
   This is a contradiction, as T' was not a MST

   Converse Counterexample
      V={w,x,y,z}
      E={(w,x,1), (x,y,1), (y,z,1)}
      S={a,d}
      (w,x) and (y,z) cross the cut (S,V-S) and are light edges
      There is only one possible spanning tree

5.

a. Suppose that all edge weights in a graph are integers in the range from 1 to |V|. How fast can you make Kruskal's algorithm run? What if the edge weights are integers in the range from 1 to W for some constant W ?

   For 1 to |V|: Counting sort can be used to sort the edges to reduce the time to

$O(E+V) = O(E)$ time. Reduces the entire algorithm to $O(E\alpha(E,V))$ time

For 1 to W: Counting sort will sort the edge weights in time $O(W+E) = O(E)$. Reduces the entire algorithm to $O(E\alpha(E,V))$ time

b. Suppose that all edge weights in a graph are integers in the range from 1 to |V|. How fast can you make Prim's algorithm run? What if the edge weights are integers in the range from 1 to W for some constant W ?

For 1 to |V|: The run time cannot be improved beyond $O(E+ V \log V)$

For 1 to W: Use a array implementation containing doubly linked lists to reduce the run time to $O(E)$