Ken Ford
07/14/17
CSCI 3104

Homework 6

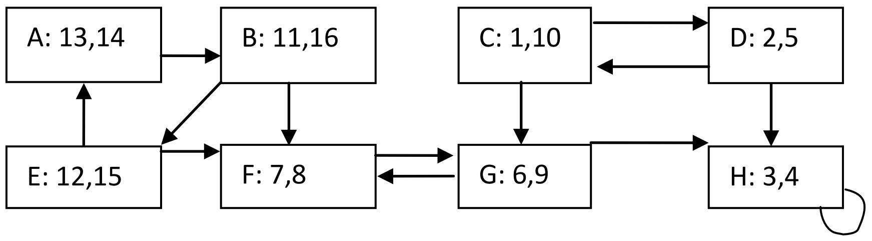
1)

```
def solutionFinder(system of inequalities):
    Convert system of inequalities into graph
        If $x_1<x_3$, there would be a directed edge from $x_1$ to $x_3$
    Run a topological sort on new graph, G
        DFS(G) will compute finishing times, will color vertices as they are finished
        Insert colored, finished vertices in linked list
        Return this linked list
    In running the topological sort, if cycle exists, then we know that there is no existing solution
        This could lead to $x_1<x_3<x_1$, resulting in  a false equality.
    Since there are no cycles, we can return the topologically sorted list of variables
        Since it is topologically sorted, all variables are organized in either increasing or decreasing order, depending on the desired result
```

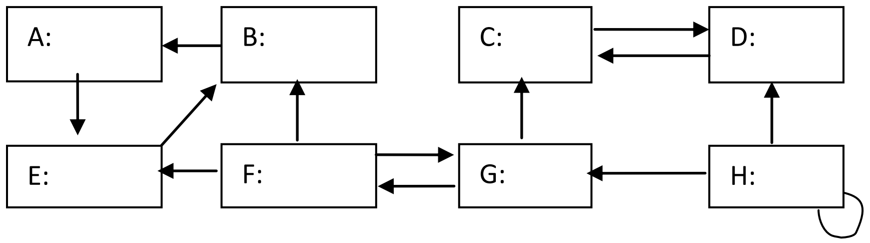
2)

```
Ordinary SCC:
    DFS(G) to get finishing times
    Compute $G^T$
    DFS($G^T$)
    Output vertices of each tree in depth first forest
```

Can we improve SCC algorithm by making second DFS call DFS(G) and ordering vertices in increasing order?



For the above sample graph G, we will run our first SCC algorithm to find the correct answer for later comparison to Gru's method. The first DFS(G) call will result in the above calculations of start and finish times for each vertex of the graph, assuming we start with vertex C. The ordered list of vertices based on finishing time for this example is (B,E,A,C,G,F,D,H). Next, we compute G^T , which is the graph below.



When we call our second DFS(G^T), we use the list of vertices from the first DFS call and check whether or not each vertex is a strongly connected component. We use the transpose of the original graph to do this. When this is done, we receive the following output: <BAE, CD, GF, H>

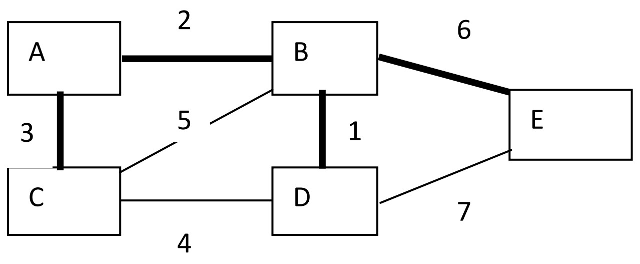
As for Gru's suggestion, we start with the first graph as pictured above. The first DFS call still returns the ordered list of (B,E,A,C,G,F,D,H). The difference comes from the fact that instead of computing the transpose, we simply invert the order of the ordered list and call DFS on the original graph. The list (B,E,A,C,G,F,D,H) becomes (H,D,F,G,C,A,E,B). When we run the second DFS call on this however, we receive a different output than before: <H,DCGF, ABE>. Due to several components either being more connected or suddenly less connected, we cannot say that Gru's suggestion will provide the correct result.

3)

$$w'(u,v) = \begin{cases} w(u,v) & \text{if } (u,v) \mathrel{!}= (x,y) \\ w(x,y) - k & \text{if } (u,v) = (x,y) \end{cases}$$

Given: graph G, edge weight set w, and MST T of G
Question: Is T an MST of G', where G' is formed by decreasing the weight of exactly one edge in T

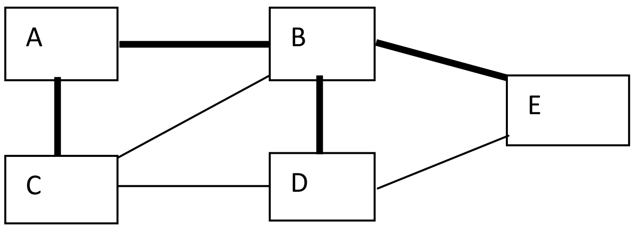
In creating the MST T, we know that an MST by definition is composed of the lowest value edge weights in the tree. As such, any changes to one edge by reducing its value would lead to the edge still being a minimum edge, just with a lower value attached to it. For example, if our initial graph G is represented by:



(T is designated by the bold lines)

For the sake of argument, we will decide that G' will be formed by decreasing the weight of (B,E). We will assume that the value k will be equal to 3, so the new weight of (B,E) will be 3. As a result, there will be no change to the edges found in T, as all we have done is reduced an already minimum weighted edge in T. Had we chosen any edge in G however, we may have had a different tree to deal with.

4)



For an example G, we will use the above graph. Assume that the bolded lines mark the MST T of G. To have a cut of G, we must remove and edge so that there is a partition of the vertices of G. If we chose some edge (u,v) to remove from the graph, then we know that the partition will be created. This is initially true for any edge in graph, as there will be at least one vertex partitioned. Due to the fact that the edge (u,v) would cross this cut and it is part of the MST, it can be considered a light edge. To show that G has a unique MST, we will assume that there exists another MST Z in which the edge (a,b) spans the cut. Similar to the edge (u,v), we know that (a,b) must also be a light edge due to it being in an MST and that it crosses the cut. Since we were given the fact that there exists a light edge crossing the cut, we know that (u,v) and (a,b) must fall into this property of uniqueness and are in fact the same edge. This can be applied to the rest of the edges in the graph as we found earlier that any edge that is cut will result in the desired partition and light edge relationship.

Converse: If a light edge crossing the cut exists, then G has a unique MST

Let’s assume that in the pictured graph above, all edges are equal to one another. We will also cut the graph down the middle, leaving vertices A and C on the left with B,D, and E on the right. Our assumption that all edge weights are the same leads us to conclude that any edge that crosses the cut would be a light edge. Therefore, we know that in this case above, edges (A,B) and (C,D) would both be light edges that cross the cut made down the middle of the graph. With this fact combined with the fact that the MST can be drawn on any of the equal edges, we cannot say that the converse statement is true.