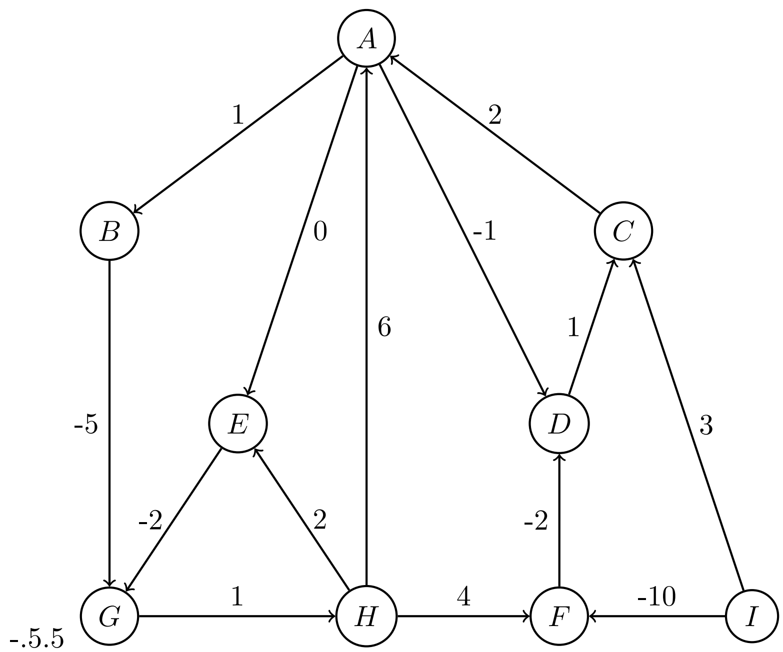
CS 170 DIS 05

Released on 2018-02-18

1 Bellman-Ford Practice

(a) Run the Bellman-Ford algorithm on the following graph, from source A. Relax edges (u, v) in lexicographic order, sorting first by u then by v.



(b) What problem occurs when we change the weight of edge (H, A) to 1? How can we detect this problem when running Bellman-Ford? Why does this work?

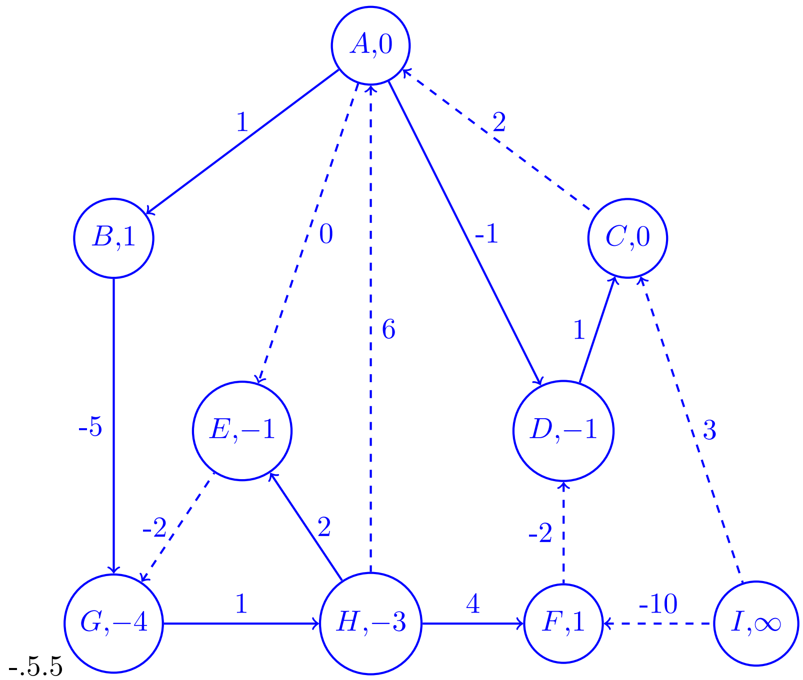
(c) Let $G = (V, E)$ be a directed graph. Under what condition does the Bellman-Ford algorithm returns the same shortest path tree (from source $s \in V$) regardless of the ordering on edges?

Solution:

(a) The resulting shortest path tree (dashed edges are non-tree edges):

Remember that you can terminate the Bellman-Ford algorithm as soon as a relaxation step does not change any distances.

(b) Changing the weight of (H, A) to 1 introduces a negative cycle. We can detect this by checking whether, after making $|V| - 1 = 9$ passes over the edges (relaxation steps), we can still update the distances. Try it out! The reason it works is that after k passes, we have found the length of all shortest paths from s with at most k edges. Since a simple path can only have at most $|V| - 1$ edges, if we can update the distances on the $|V|$-th pass, the new shortest path contains a cycle. A cycle can only be part of a shortest path if it is of negative weight.



(c) If and only if the shortest path tree rooted at s is unique, i.e. if for each $v \in V$ there exists a unique shortest path from s to v. Note that this is not the case in the example: there are two paths of length -1 from A to D.

2 MST Basics

For each of the following statements, either prove or supply a counterexample. Always assume $G = (V, E)$ is undirected and connected. Do not assume the edge weights are distinct unless specifically stated.

(a) Let e be any edge of minimum weight in G. Then e must be part of some MST.

(b) If e is part of some MST of G, then it must be a lightest edge across some cut of G.

(c) If G has a cycle with a unique lightest edge e, then e must be part of every MST.

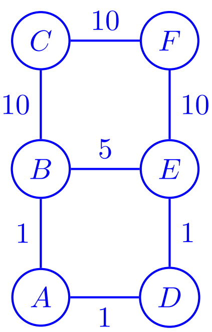
(d) For any $r > 0$, define an r-path to be a path whose edges all have weight less than r. If G contains an r-path from s to t, then every MST of G must also contain an r-path from s to t.

Solution:

1. True, e will belong to the MST produced by Kruskal.
2. True, suppose (u, v) is the edge. Let one side of the cut be everything reachable in the MST from u without using the edge (u, v). If this cut has an edge lighter than (u, v) then we could add this edge to the MST and remove (u, v). We know this edge is not

already in the MST because otherwise both its endpoints would be reachable from u without using (u, v).

3. False. Let e be also the heaviest edge of a different cycle; then, we know that e can't be part of the MST. Concretely, in the following graph, edge (B, E) satisfies this condition, but will not be added to the graph.



-.5.5

4. True. Let $v_1, v_2, \ldots, v_n$ denote the r path in G from $s = v_1$ to $t = v_n$. Consider the greatest i such that the MST has an r path from s to v_i and assume for contradiction that $i < n$. Then the edge (v_i, v_{i+1}) is not in the MST. Adding this edge to the MST forms a cycle, which must have edges of length less than r since otherwise we could replace one of them with (v_i, v_{i+1}). Thus there is an r path from v_i to v_{i+1} and hence from s to v_{i+1}, contradicting our initial assumption.

3 Minimum Spanning Trees (short answer)

(a) Given an undirected graph $G = (V, E)$ and a set $E' \subset E$ briefly describe how to update Kruskal's algorithm to find the minimum spanning tree that includes all edges from E'.

(b) Suppose we want to find the minimum cost set of edges that suffices to connect a given weighted graph $G = (V, E)$; if the weights are non-negative then we know that the optimum will be a MST. What about the case when the weights are allowed to be negative? Does it have to be a tree if the weights are allowed to be negative? If not, how would you find this minimum-cost connected subgraph?

(c) Describe an algorithm to find a maximum spanning tree of a given graph.

Solution:

(a) Assuming E' doesn't have a cycle, add all edges from E' to the MST first, then sort $E \backslash E'$ and run Kruskal's as normal.

(b) In case the weights are allowed to be negative, the minimum-cost connected subgraph is not necessarily a tree; consider for example the cycle graph on n vertices with each edge having negative weight. The minimum-cost connected subgraph is the complete graph since removing any single edge only increases the cost of the selected component.

Let us denote this component by $G' = (V, E')$ where $E' = \phi$ is the set of empty edges intially. Add all the negative weight edges to E'. If the component is still not connected, run Kruskal's algorithm beginning from this set E' of the edges.

(c) Negate all edge weights and apply MST algorithm by scaling up all edge weights to ensure they are all positive.

4 Picking a Favorite MST

Consider an undirected, weighted graph for which multiple MSTs are possible (we know this means the edge weights cannot be unique). You have a favorite MST, F. Are you guaranteed that F is a possible output of Kruskal's algorithm on this graph? How about Prim's? In other words, is it always possible to "force" the MST algorithms to output F without changing the weights of the given graph? Justify your answer. **Solution:** Yes; for both MST algorithms, it's possible to ensure they output F, provided it is indeed an MST.

First, consider Kruskal's algorithm. Make sure that the edges of F are always before any other equally-weighted edges after the sort. Now, it will add all such edges as early as possible. Consider towards a contradiction the case where at any point Kruskal's declines to add an edge e in F to the MST. This means that e would have created a cycle with other equally-weighted edges in F, and/or lighter edges (possibly in F). (Before this point, Kruskal's may have added edges not in F to the MST.) But then it's impossible that both e and all of the other equally-weighted edges in F in this cycle can be in any MST, as one of them is the heaviest edge in some cycle of the graph, and such edges cannot be in any MST.

Since we assumed that F is an MST, this is a contradiction. Therefore we conclude that Kruskal's algorithm will add all edges in F to the MST. And since all MSTs have the same number of edges, this means it cannot add any edges not in F.

Now, consider Prim's algorithm. As it expands the fringe, have it only choose edges in F (so when there are multiple lightest edges to choose, choose a lightest edge in F). If this strategy fails, there must have been some cut across which none of the lightest edges were in F. But if this is the case, F cannot have been an MST (one of the lightest edges across any cut must be in any MST). Given that F must be an MST, this strategy will work.