CSC 373H Midterm #1

Spring 2016

St. George Campus

Duration — 50 minutes

SOLUTIONS

Question 1. [30 MARKS]

Let $G = (V, E, w)$ be a undirected, connected, weighted graph and assume all the following:

(1.1) The weights $w(e)$ are all integers (possibly negative) and distinct, that is, for any $e, f \in E$ with $e \neq f$, we have $w(e) \neq w(f)$;

(1.2) Suppose $T \subset E$ is such that (V, T) is a spanning tree of G;

(1.3) Suppose $e = (u, v) \in E \backslash T$ is a particular edge such that the simple path from u to v in T consists of exactly the two edges $f_1 = (u, a)$ and $f_2 = (a, v)$ in T (for some $a \in V$);

(1.4) Suppose $w(f_1) = 1$ and $w(f_2) = 10$; and

(1.5) Suppose $|V| > 3$, that is, there are more than three vertices in G.

Given the above assumptions state whether each of the claims below are true or false. **In either case PROVIDE A SHORT PROOF** of your answer.

Part (a) [10 MARKS]

Claim A: In addition to assumptions (1.1) through (1.5) above, suppose $w(e) > w(f_2) > w(f_1)$. Then the edge e is not in any minimum spanning tree (MST) of G.

Solution:
True.

By the strong cycle property of MST's. Specifically, the Strong Cycle Property states that if f is the edge with the unique maximum weight in a simple cycle C, the no MST contains f.

In this case e is such an edge in the cycle C = (u, v, a, u).
Alternative Solution: (Sketch) By contradiction. Suppose (V, T) is a MST which contains e. Then swap e with f_1 to find a spanning tree (V, T') with value $v(T') = v(T) - (w(f_1) - w(e)) < v(T)$. To get full marks the student needs to clearly prove that T' is a spanning tree, see part (c) solution below.

Part (b) [10 MARKS]

Claim B: In addition to assumptions (1.1) through (1.5) above, suppose $w(e) < w(f_1) < w(f_2)$. Then the edge e is in some MST of G.

Solution:
False. Show a counter-example. Suppose $w(e) = 0$, and suppose the graph G consists of $V = \{s, u, a, v\}$ and $E = \{e, f_1, f_2, (s, u), (s, v), (s, a)\}$. The weights on the first three edges are as above (i.e., 0, 1, 10, respectively), while the last three edges all have weight -1. The MST of G is the $(V, \{(s, u), (s, v), (s, a)\}$ with value -3. (Marker can take this as the obvious MST.) This MST does not contain e.

Part (c) [10 MARKS]

Claim C: In addition to assumptions (1.1) through (1.5) above, suppose $w(e) < w(f_2)$. Then there exists a spanning tree T' of G with value $v(T') = v(T) - w(f_2) + w(e)$.
Use only basic facts about spanning trees in your proof. Specifically, use only that spanning trees are necessarily connected, have $|V| - 1$ edges, and are acyclic, along with the corresponding sufficient conditions for a sub-graph to be a spanning tree.

Solution: The claim is true.

Let $T' = T \backslash \{f_2\} \cup \{e\}$. We will show (V, T') is a spanning tree below.

Then T' has the same number of edges as T since $f_2 \in T$ and $e \notin T$. Therefore, $|T'| = |T| = |V| - 1$.

Note that, for the same reason as in the previous line, the sum of the weights in T', i.e., $v(T')$ is $v(T') = v(T) - w(f_2) + w(e)$.

Next, we show the graph (V, T') is also connected. Let x, y be any pair of vertices in V. Then there is a simple path P from x to y in (V, T) (since (V, T) is a spanning tree). Any occurrence of the edge $f_2 = (a, v)$ in P can be swapped with the path (a, u, v). And similarly if the reverse direction edge (v, a) is on P. The result is a path connecting x and y in (V, T'). Therefore we've shown (V, T') is connected.

From the lecture notes we know that any connected graph with $|V| - 1$ edges is necessarily a spanning tree.

We've therefore proved that (V, T') is a spanning tree with the required value.

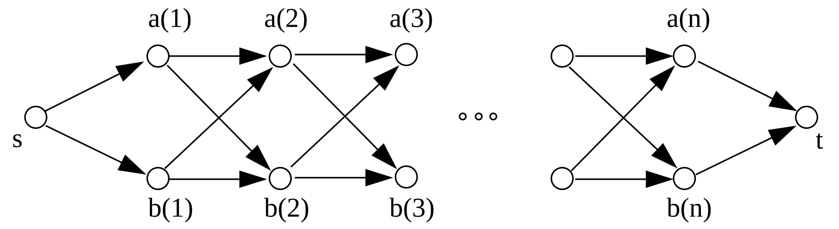
Question 2. [30 MARKS]

Suppose we are given a connected, directed and weighted graph $G = (V, E, w)$ of the form sketched below. Specifically, for some $n > 0$ we have

(2.1) $V = \{s, t\} \cup \{a(k), b(k)\}_{k=1}^{n}$;

(2.2) $E = \{(s, a(1)), (s, b(1)), (a(n), t), (b(n), t)\} \cup \{(a(k-1), a(k)))\}_{k=2}^{n} \cup \{(b(k-1), a(k)))\}_{k=2}^{n} \cup \{(a(k-1), b(k)))\}_{k=2}^{n} \cup \{(b(k-1), b(k)))\}_{k=2}^{n}$ (see sketch below); and

(2.3) For any $e \in E$, the weight $w(e)$ is an integer (possibly negative).



For any path P in G, define the length of P to be the sum of the weights of the edges in P. The problem is to find the longest path from s to t using a specific dynamic programming approach.

Part (a) [10 MARKS]

Consider sub-problems which, for each $i = 0, 1, 2, \ldots, n+1$, are restricted to finding a longest path from s that have at most i edges (many of these paths do not reach t). You may consider separating these sub-problems into further sub-cases, but you **must consider sub-problems that consist of sets of paths which start at s and consist of at most i edges**. Clearly define the sub-problems you decide to use, along with a recurrence relation for the maximum path length in each sub-problem. Explain why your recurrence relation is correct.

Solution:

Sub-Problems: For $1 \leq k \leq n$:
a) find the longest path from s to a(k);
b) find the longest path from s to b(k).

Optimal Values: Let $O(k, a)$ and $O(k, b)$ denote the length of the longest path for each of the above two sub-problems (respectively). Define $O(t)$ to be the maximum length of a path from s to t.

Initial Values: Since the corresponding single edge forms the unique path from s to a (or b), we have $O(1, a) = w((s, a))$ and $O(1, b) = w((s, b))$.

Recurrence Relation: For $1 \leq k < n$:

$$O(k+1, a) = \max \begin{cases} O(k, a) + w((a(k), a(k+1)), \\ O(k, b) + w((b(k), a(k+1)). \end{cases} \tag{1}$$

$$O(k+1, b) = \max \begin{cases} O(k, a) + w((a(k), b(k+1)), \\ O(k, b) + w((b(k), b(k+1)). \end{cases} \tag{2}$$

And, finally, the longest s-t path has length (see below for explanation):

$$O(t) = \max \begin{cases} O(n, a) + w((a(n), t)), \\ O(n, b) + w((b(n), t)). \end{cases} \tag{3}$$

Explanation: Any path from s to $a(k+1)$, for example, must include one of the sub-paths from s to $a(k)$ or s to $b(k)$, followed by a single edge from the last point to $a(k+1)$. That is, this is the exhaustive list of options. The expression in equation (1) above is the maximum over this exhaustive list. Note we only need consider the length of the longest paths between s to $a(k)$ or s to $b(k)$. A similar argument applies for equations (2) and (3).

Part (b) [10 MARKS]

Clearly explain how a longest s-t path can be computed given the optimum values you found in part (a).

Solution:

```
Set $Q \leftarrow (t)$
Set $V \leftarrow O(t)$, where $O(t)$ is the length of the longest path (computed above)
for each k = n, n-1, ..., 1
    Let $v \leftarrow Q(1)$ (i.e., here $Q(1)$ the first vertex of path $Q$)
    If $V == O(k,a) + w(a(k), v)$,
        $Q \leftarrow (a(k), Q)$ (i.e., prepend $a(k)$ to path $Q$)
        $V \leftarrow O(k,a)$
    Else (it must be the case $V == O(k,b) + w(b(k), v)$),
        $Q \leftarrow (b(k), Q)$
        $V \leftarrow O(k,b)$
end for
$Q \leftarrow (s, Q)$.
return $Q$ as a longest s-t path.
```

Part (c) [10 MARKS]

Consider a similar problem where, in addition to the edges described above, E also includes several edges of the form (u,v) with $u \in S_{k-2} \equiv \{s\} \cup \{a(j), b(j)\}_{j=1}^{k-2}$ and $v \in \{a(k), b(k)\}$ for any $k \in \{2, 3, \ldots, n\}$. For example, the edges $(s, a(2))$ and $(a(1), b(5))$ could be added to those already in E. Clearly explain how your recurrence relation for the maximum path lengths would change from part (a). (You do not need to explain how to find an actual path of maximal length.)

Solution: The sub-problems are as above, with the same initialization of $O(1,a)$ and $O(1,b)$. The recurrence relations are updated as follows:
Recurrence Relation: For $1 \le k < n$

$$\begin{aligned} O(k+1,a) = \max[&\{w((s,a(k+1))) \mid (s,a(k+1)) \in E\} \cup \\ &\{w((a(j),a(k+1))) + O(j,a) \mid (a(j),a(k+1)) \in E \text{ for } 1 \le j \le k\} \cup \\ &\{w((b(j),a(k+1))) + O(j,b) \mid (b(j),a(k+1)) \in E \text{ for } 1 \le j \le k\}]. \end{aligned} \tag{4}$$

The correpsonding modification should also be made to equation (2) (omitted).
Equation (3) in part (a) remains as it is.
[**Notes for tutorial sections**] Optional questions, if you have time: i) How to find a maximal length path in case (c) above; ii) What is the runtime of part (c)?; iii) How does that runtime compare to the worst case analysis of Bellman-Ford?; or iv) If the weights were all non-negative, would the Dijkstra algorithm be faster than the algorithm for part (c) plus the runtime to recover the longest path (without using breadcrumbs)?