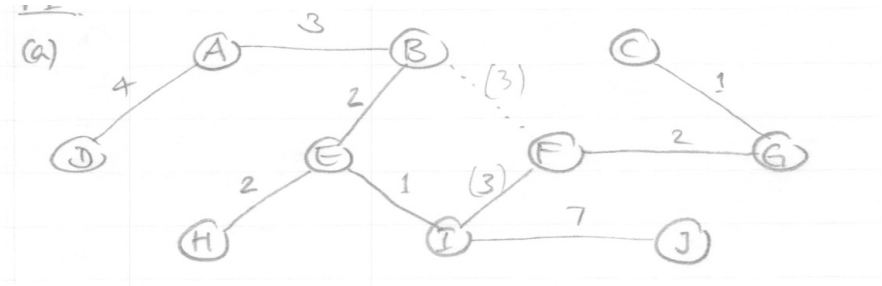
CS340 – Advanced Data Structures and Algorithm Design – Fall 2020
Assignment 8 – November 27, 2020

Dr. Sandra Zilles, Department of Computer Science, University of Regina

answer key

Problem 1 (6+1 marks).

(a) There are two options, as indicated in the following graphical representation of an MST:



[2 marks, even if only one of the two options is given.]

Order in which the edges are output by Prim's algorithm: [2 marks]

(A, B), (B, E), (E, I), (E, H), either (I, F) or (B, F), (F, G), (G, C), (A, D), (I, J)

Note: here I started at vertex A, but one could also start at any other vertex, which would then of course yield a different order.

Order in which the edges are output by Kruskal's algorithm: [2 marks]

(E, I), (G, C), (F, G), (E, H), (B, E), (A, B), either (I, F) or (B, F), (A, D), (I, J)

(b) No. Both algorithms have the choice between (I, F) and (B, F) at some point. Only one of the two edges will be in the MST. [1 mark]

Problem 2 (2+2 marks).

(a) A, B, C, E, G, D, F [2 marks]

(b) A, B, C, D, F, E, G [2 marks]

Problem 3 (4 marks). Such a vertex w corresponds to a row that contains only zeroes and a column that contains only ones (except for the entry on the diagonal, which is a zero).

Rough idea: Starting with $i = j = 0$, we parse the ith row from column j to the right as long as each entry is zero. If the jth entry in the ith row is a 1, then the vertex i is no longer a candidate for w. But for $j > i$, we know more than that: the vertices corresponding to indices k with $i < k \leq j - 1$ are no longer candidates either, because each of them have a zero in column k (off the diagonal). Then we go down in the matrix from position $[i][j]$ for as long as we see ones. If we hit a zero, we go right again, and so on. When we hit the right or bottom end of the matrix, the current row index or column index, respectively, is the only possible candidate for w, but still has to be verified.

In pseudocode:

1. Set $i = 0$ and $j = 0$.
2. While $i < |V|$ and $j < |V|$:
   if $A[i][j] = 0$, then $j++$, else $i++$.
3. If $j < V$, then check whether row j has all zeroes. If yes, return 'yes'.
4. If $i < V$, then check whether column i has all ones except in the diagonal. If yes, return 'yes'.

5. Return ‘no’.

Together, i and j are incremented at most $|V|$ times. The check after the while loop obviously takes time linear in $|V|$. Hence, the overall running time is $\Theta(|V|)$.

Problem 4 (3 marks).

The following reduction mapping will be used: Given a problem instance for $\mathcal{P}$, consisting of a graph $G = (V, E)$ and an integer $k \leq |V|$, the output of the reduction mapping is a problem instance for $\mathcal{P}'$, consisting of a graph $G' = (V', E')$ and an integer $k' \leq |V|$, where

- $k' = k$,
- $V' = V$,
- $E' = \{(v, w) \in V \times V \mid (v, w) \notin E\}$, i.e., E' makes exactly those pairs of vertices in V adjacent in G' that are *not* adjacent in the original graph G.

Since E' can be computed in time quadratic in $|V| + |E|$, this reduction mapping can be computed in polynomial time.

[1 mark for the correct mapping, 1 mark for explaining why it runs in polynomial time.]

It is now easy to see the following:

- If (G, k) is a positive instance of $\mathcal{P}$, then (G', k') is a positive instance of $\mathcal{P}'$. [0.5 marks]
- If (G, k) is a negative instance of $\mathcal{P}$, then (G', k') is a negative instance of $\mathcal{P}'$. [0.5 marks]