Homework 4

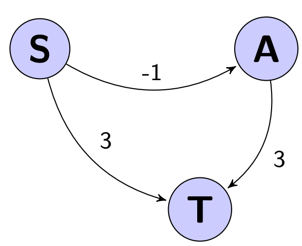
Victor Frolov, J.B. Morris, Trixie Roque, Khalid Seirafi

05/04/15

1. Using the Bellman-Ford algorithm, this table is created:

Nodes \ Iterations	0	1	2	3	4	5
S	0	0	0	0	0	0
A	∞	7	7	7	7	7
B	∞	∞	11	11	11	11
C	∞	6	5	5	5	5
D	∞	∞	8	7	7	7
E	∞	6	6	6	6	6
F	∞	5	4	4	4	4
G	∞	∞	∞	9	8	8
H	∞	∞	9	7	7	7
I	∞	∞	∞	∞	8	7

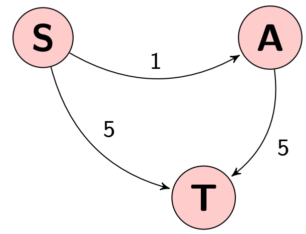
2. The method proposed by Professor F. Lake is incorrect. A possible counterexample to this method is if the shortest path of the original graph has negative edge weights:



Original Graph

This graph's shortest path from S to T costs 2: $S \Longrightarrow A \Longrightarrow T$

Using Professor F. Lake's method, the edge weights will all be positive and the shortest path will be different:



Modified graph

Here, the shortest path costs 5: $S \Longrightarrow T$

3. Proof:

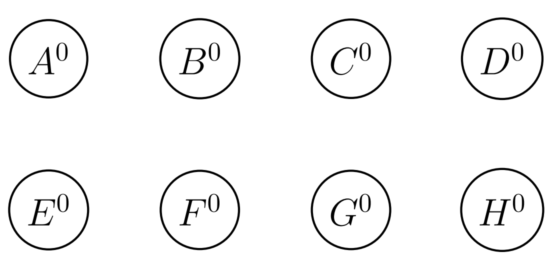
Since we only need to prove O, we have to find an upper bound so that $O(W|V|+E)$. Given a graph whose edge weights are integers in the range $0, 1, ..., W$, where W is a relatively small number, the maximum weight that any of the edge can have is W. Also, given V number of vertices, updating a vertex using Dijkstra's algorithm will only take at most $|V|-1$ times since the vertex itself will not be looping to itself. Then, using Dial's implementationusing of using doubly-linked lists which have the properties that allow constant time checking whether a graph is empty/nonempty, and deleting/adding a node, the edges will only be $O(E)$. Thus, we can have $O(W|V|+E)$ complexity.

(Source: http://www.ece.northwestern.edu/ dda902/336/hw5-sol.pdf)

4. (a) Prim's algorithm produced the table below

Set S	A	B	C	D	E	F	G	H
A	0/nil	1/A	∞/nil	∞/nil	4/A	8/A	∞/nil	∞/nil
A, B			2/B	∞/nil	4/A	6/B	6/B	∞/nil
A, B, C				3/C	4/A	6/B	2/C	∞/nil
A, B, C, G				1/G	4/A	1/G		1/G
A, B, C, G, D					4/A	1/G		1/G
A, B, C, G, D, F					4/A			1/G
A, B, C, G, D, F, H					4/A			

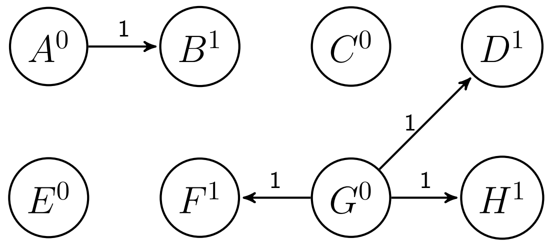
(b) We start with disjoint nodes



Then using Kruskal's algorithm, we take the edges with the smallest weight. In this case, we start with length = 1: (A, B), (G, D), (G, F), (G, H) so we have:

$A^0 \implies B^1$, $G^0 \implies D^1$, $G^0 \implies F^1$, $G^0 \implies H^1$

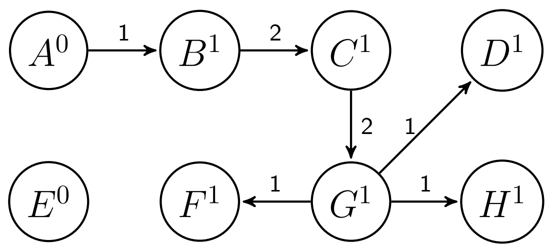
Then we have the sets {A, B}, {D, G} {F, G}, {H, G}, but since three of these sets are not disjoint, we actually have the disjoint sets {A, B} and {D, F, G, H} in order to cover the edges with length 1.



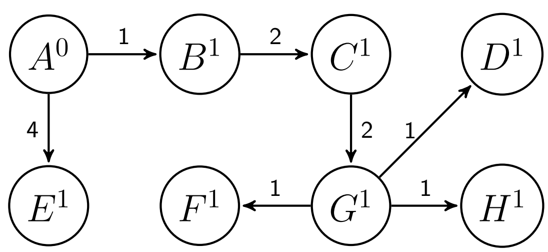
Then, length = 2 edges are (B, C), (C, G) so we have:

$A^0 \implies B^1 \implies C^1$, $C^0 \implies G^1$

Since these aren't disjoint sets, we actually have the set {A, B, C, D, G, F, H} such that



Then finally adding node E, since we can only use edges (A, E) with length = 4 and (F, E) with length = 5, we choose the edge with the smallest weight, (A, E), just like how we have been doing the other edges. Then we have the set {A, B, C, D, E, G, F, H}



5. See SubsetSum.java in GitHub repo trixr4kdz/cmsi282/homework4.

6. See BoardingSchool.java in GitHub repo trixr4kdz/cmsi282/homework4.