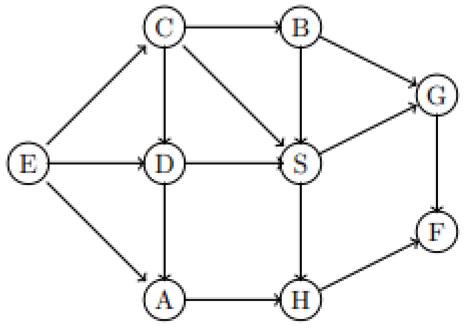
Algorithms Midterm #2

Ozaner Hansha

April 7, 2020

Problem 1

For the following question consider the following directed graph:



Part a: Do a DFS starting at C, assuming vertices are to be considered in alphabetical order. List each vertex along with their discovery and finish times. Then classify each edge as a tree edge, back edge, forward edge, or cross edge.

Solution: The discovery and finish times for each vertex are given below (note that time starts at $t = 1$):

Vertex	Discovery Time	Finished Time
A	13	14
B	2	11
C	1	16
D	12	15
E	17	18
F	4	5
G	3	6
H	8	9
S	7	10

Note that after we finish node C at time 16, we add the undiscovered node E to the DFS stack at time 17. Recall that DFS returns a forest rather than just a tree. Each of the forest's edges is classified below:

Tree Edges	Forward Edges	Cross Edges	Backward Edges
$(C, B), (C, D), (B, G), (B, S), (D, A), (G, F), (S, H)$	(C, S)	$(D, S), (S, G), (H, F), (A, H), (E, A), (E, C), (E, D)$	

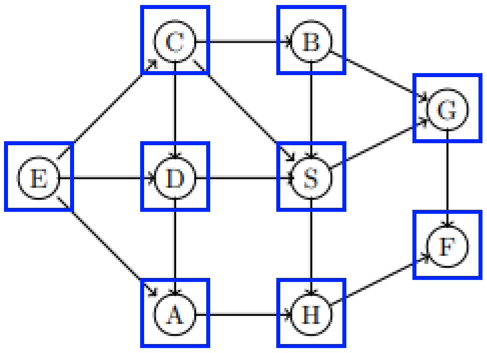
Part b: Use the DFS from part a to do a topological sort on this graph. List the vertices on a line in topological sorted order.

Solution: Recall that performing a topological sort on a DAG, of which the above graph is due to not having any back edges, is equivalent to listing the finish times of of DFS in reverse order. Thus, our topological sort returns the following ordered list of vertices:

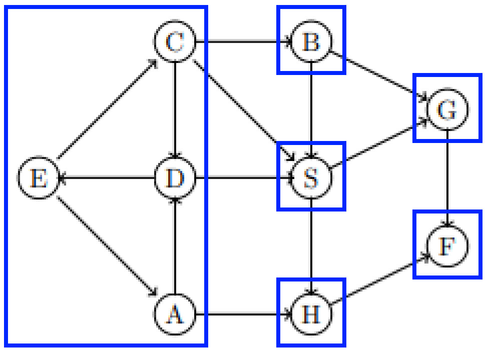
$$E, C, D, A, B, S, H, G, F$$

Part c: Identify the strongly connected components (SCCs) of the graph. Suppose now we switch the direction of the edges (D, A) and (E, D), what are the SCCs of the new graph?

Solution: Note that since the original graph was a DAG, every vertex is itself an SCC (every SCC is denoted by a blue box):



In the case of the modified graph, the SCCs are given by:



Problem 2

Problem: Suppose n people attend a party and some shake hands with others. Prove there are at least two people who have shaken hands with the same number of people.

Solution: Consider a function $f(x)$ that returns the number of people the xth person at the party has shaken hands with. The domain of the f is the integer interval $[1..n]$. The range is any number from 0 to $n-1$, since a person can shake anywhere from no hands to every other person's hands.

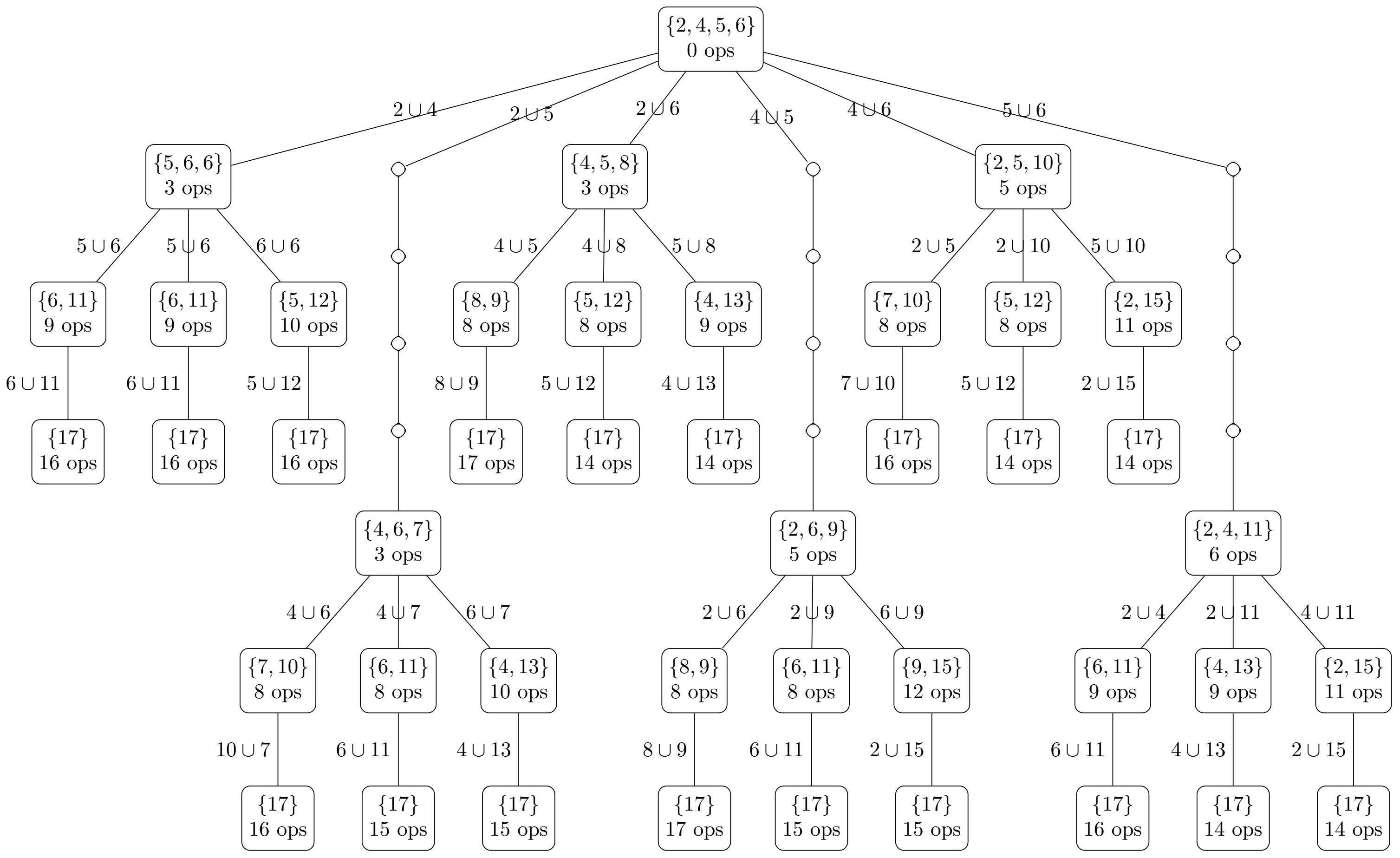
However, note that $f(i) = n-1$ and $f(j) = 0$ are mutually exclusive for any i and j. This is because if a person i has shaken every other person's hands, there can be no person j who has shaken no hands as j would have at least shaken i's hand. As such, no matter how the hand shaking takes place, f can only output $n-1$ different values at most. Since there are n possible inputs (i.e. n people) the pigeonhole-principle tells us that that $f(i) = f(j)$ for some i and j (i.e. at least 2 people must have shaken hands the same number of times).

Problem 3

Problem: Suppose in the process of computing the MST of a graph with 17 vertices via Kruskal's algorithm we have obtained connected components having sizes 2,4,5,6. Using weighted-union, give all possible scenarios as well the minimum and maximum number of possible operations needed to get an MST.

(An operation is changing a link. For example, if the component with 2 elements gets added to the component with 4 elements, we change two links for the smaller set plus linking the head of shorter to the tail of the longer, for a total of 3 operations.)

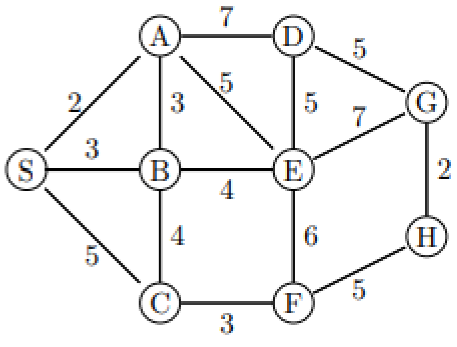
Solution: Below is a tree detailing all 18 possible ways these 4 connected components could form an MST.



As we can see, the min and max number of operations combining these sets would take are 14 and 17 respectively.

Problem 4

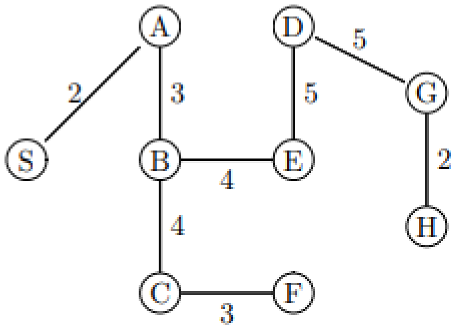
Part a: Compute the MST of the following graph using Kruskal's algorithm. Show the order of the selected edges as well as the sets formed. If there is a tie use alphabetical ordering.



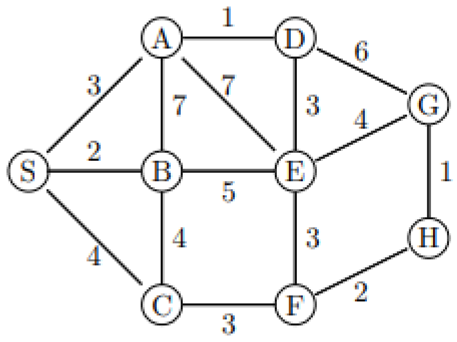
Solution: The sets of spanned vertices and corresponding selected edge are given in the following table:

Time Step	Selected Edge	Added?	Sets of Spanned Vertices
0	-	-	$\{A\},\{B\},\{C\},\{D\},\{E\},\{F\},\{G\},\{H\},\{S\}$
1	(A,S)	Yes	$\{A,S\},\{B\},\{C\},\{D\},\{E\},\{F\},\{G\},\{H\}$
2	(G,H)	Yes	$\{A,S\},\{B\},\{C\},\{D\},\{E\},\{F\},\{G,H\}$
3	(A,B)	Yes	$\{A,B,S\},\{C\},\{D\},\{E\},\{F\},\{G,H\}$
4	(B,S)	No	$\{A,B,S\},\{C\},\{D\},\{E\},\{F\},\{G,H\}$
5	(C,F)	Yes	$\{A,B,S\},\{C,F\},\{D\},\{E\},\{G,H\}$
6	(B,C)	Yes	$\{A,B,C,F,S\},\{D\},\{E\},\{G,H\}$
7	(B,E)	Yes	$\{A,B,C,E,F,S\},\{D\},\{G,H\}$
8	(A,E)	No	$\{A,B,C,E,F,S\},\{D\},\{G,H\}$
9	(C,S)	No	$\{A,B,C,E,F,S\},\{D\},\{G,H\}$
10	(D,E)	Yes	$\{A,B,C,D,E,F,S\},\{G,H\}$
11	(D,G)	Yes	$\{A,B,C,D,E,F,G,H,S\}$

Once (D,G) is added to the spanning tree, our sets have all been combined and our MST is complete:



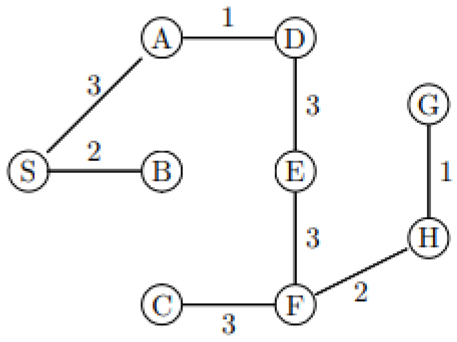
Part b: Compute the MST of the following graph using Prim's algorithm. Start at S and give the order of the selected edges.



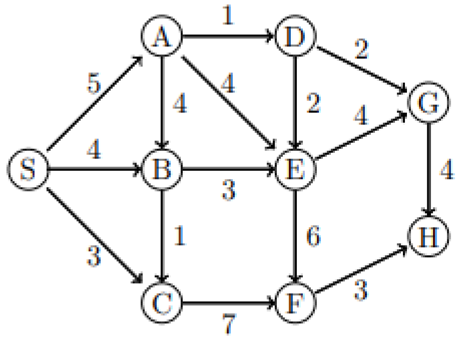
Solution: Running Prim's algorithm starting at S nets us:

Order	Edge	Spanned Vertices
1	(B, S)	B, S
2	(A, S)	A, B, S
3	(A, D)	A, B, D, S
4	(D, E)	A, B, D, E, S
5	(E, F)	A, B, D, E, F, S
6	(F, H)	A, B, D, E, F, H, S
7	(H, G)	A, B, D, E, F, G, H, S
8	(C, F)	$A, B, C, D, E, F, G, H, S$

Once (C, F) is added to the spanning tree, all vertices have been spanned and our MST is complete:



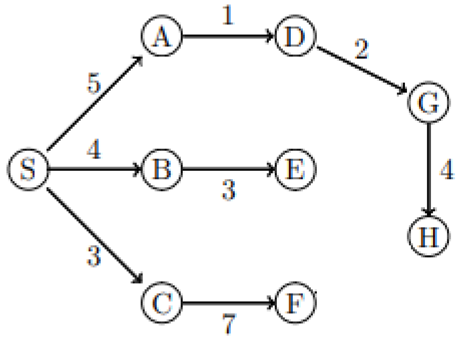
Part c: Using Dijkstra's algorithm find the shortest path from S to all other vertices. Show the complete update history of each vertex v, starting with infinity as the value of $d(v), v \neq S$, and finishing with the length of the shortest path.



Solution: Each iteration of Dijkstra's algorithm is given below:

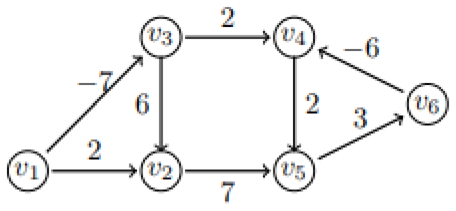
Iteration	***S***	***A***	***B***	***C***	***D***	***E***	***F***	***G***	***H***
0	0, Null	∞, Null	∞, Null	∞, Null	∞, Null	∞, Null	∞, Null	∞, Null	∞, Null
1	**0, Null**	5, S	4, S	3, S	∞, Null	∞, Null	∞, Null	∞, Null	∞, Null
2	-	5, S	4, S	$\mathbf{3, \boldsymbol{S}}$	∞, Null	∞, Null	10, C	∞, Null	∞, Null
3	-	5, S	$\mathbf{4, \boldsymbol{S}}$	-	∞, Null	7, B	10, C	∞, Null	∞, Null
4	-	$\mathbf{5, \boldsymbol{S}}$	-	-	6, A	7, B	10, C	∞, Null	∞, Null
5	-	-	-	-	$\mathbf{6, \boldsymbol{A}}$	7, B	10, C	8, D	∞, Null
6	-	-	-	-	-	$\mathbf{7, \boldsymbol{B}}$	10, C	8, D	∞, Null
7	-	-	-	-	-	-	10, C	$\mathbf{8, \boldsymbol{D}}$	12, G
8	-	-	-	-	-	-	$\mathbf{10, \boldsymbol{C}}$	-	12, G
9	-	-	-	-	-	-	-	-	$\mathbf{12, \boldsymbol{G}}$

From this table we can construct the following shortest path tree with root S:



Problem 5

Problem: Write the system of linear inequalities corresponding to the following graph, then use the approach based on Bellman-Ford to find a solution to the system of inequalities using inspection or prove there is no solution. Do the same when the weight of the edge (v_6, v_4) is changed to -5.



Solution: The corresponding inequalities are given by:

$$\begin{array}{llll} x_3 - x_1 \le -7 & x_2 - x_1 \le 2 & x_2 - x_3 \le 6 & x_5 - x_2 \le 7 \\ x_4 - x_3 \le 2 & x_5 - x_4 \le 2 & x_4 - x_6 \le -6 & x_6 - x_5 \le 3 \end{array}$$

Upon inspection, we note that there is a negative cycle (v_4, v_5, v_6) summing to -1. As a result, there is no solution to the corresponding system of inequalities.

Changing (v_6, v_4), and its corresponding inequality, to have a weight of -5, we no longer have a negative cycle and produce the following iteration of Bellman-Ford with the additional node v_0:

Iteration	$\boldsymbol{v_0}$	$\boldsymbol{v_1}$	$\boldsymbol{v_2}$	$\boldsymbol{v_3}$	$\boldsymbol{v_4}$	$\boldsymbol{v_5}$	$\boldsymbol{v_6}$
0	0, Null	∞, Null	∞, Null	∞, Null	∞, Null	∞, Null	∞, Null
1	0, Null	0, v_0	0, v_0	-7, v_1	-5, v_3	-3, v_4	0, v_0
2	0, Null	0, v_0	-1, v_3	-7, v_1	-5, v_3	-3, v_4	0, v_0
3	0, Null	0, v_0	-1, v_3	-7, v_1	-5, v_3	-3, v_4	0, v_0
4	0, Null	0, v_0	-1, v_3	-7, v_1	-5, v_3	-3, v_4	0, v_0
5	0, Null	0, v_0	-1, v_3	-7, v_1	-5, v_3	-3, v_4	0, v_0
6	0, Null	0, v_0	-1, v_3	-7, v_1	-5, v_3	-3, v_4	0, v_0

After $7 - 1 = 6$ iterations we can be sure that we have our solution:

$$\begin{array}{lll} x_1 = 0 & x_2 = -1 & x_3 = -7 \\ x_4 = -5 & x_5 = -3 & x_6 = 0 \end{array}$$

Problem 6

Part a: Assume $G = (V, E)$ is an undirected connected graph consisting of a spanning tree plus one more edge. The edges have weights. Describe an efficient algorithm for computing MST and its complexity in terms of $n = |V|$.

Solution: Since the graph was a spanning tree before the extra edge was added, there must be one cycle in the graph. Our goal is to find this cycle and remove the edge in that cycle with the highest weight. Since we can only remove one edge before the graph is a spanning tree, removing the highest one ensures it is a MST. We do this by performing the following algorithm:

Perform DFS on any of the nodes, making sure to store each visited edge in order on a stack as well as to mark each node visited when discovered. The search will eventually reach a node that has been visited before (because it is connected and must have a cycle as previously mentioned). When this happens we store that node as `twiceNode` and store the edge that led to it as `largestEdge`. From this point on we abandon the DFS and instead backtrack through the edge stack we have stored, making sure to compare each edge with `largestEdge` and replacing it if larger. We stop this backtracking once we reach `twiceNode` again.

At this point our algorithm is complete. We now simply remove `largestEdge` (which is the largest edge in the cycle) from the given graph and we are left with the MST.

In terms of complexity, in the worst case the above algorithm will move through all edges twice (once forward and once backwards) as well as encountering one vertex twice and the rest once. The complexity of encountering an edge or vertex is constant and so too is the complexity of removing the largest edge at the end of the algorithm. This leaves us with $k_1|E| + k_1(|V| + 1)$ operations which is $O(|E| + |V|)$. However, note that since this graph is a spanning tree (which all have exactly $|V| - 1$ edges) plus an extra edge, we have that there are $|V| - 1 + 1 = |V|$ edges on this graph. And so the complexity reduces to: $O(|E| + |V|) = O(|V| + |V|) = O(|V|) = O(n)$.

Part b: Let $G = (V, E)$ be an undirected complete graph with n vertices. Suppose after we have computed the MST of G we realize that we missed one of the nodes so that G is actually a complete graph with $n + 1$ nodes. Describe an efficient algorithm for computing the MST of the augmented graph.

Solution: Given an MST of the graph *without* the new node, we only need to add the edge with the smallest weight that connects to the new node. This will result in an MST of the augmented graph.

In the worst case, this new node will be connected to every other node (i.e. have n edges). As such, it will take a selection algorithm $O(n)$ time to find the (not necessarily unique) smallest edge and constant time to add it to the old MST, resulting in a total complexity of $O(n)$.

Part c: Suppose $G = (V, E)$, where $n = |V|$ and $m = |E|$, is an edge weighted graph where the edge weights are distinct (no two edges have the same weight). Prove the MST of G is unique.

Solution: Suppose there are two distinct MSTs A and B. Let e be the smallest edge in $A \Delta B$ (i.e. the smallest edge not shared by A and B). W.l.o.g. we can say $e \in A$.

Now call $e = (P, Q)$. Since $e \notin B$, B must contain some other path from P to Q since it is a spanning tree. As such, the union $B \cup \{e\}$ must contain a cycle since it contains two paths from P to Q. Note that if all the edges that were part of that cycle were in A, then A would also have a cycle which violates its definition as an MST. As such, there must be at least one edge $f \in B$ such that it is part of that cycle but not in A.

Since $f \in A \Delta B$ we have, by definition, that $w(e) < w(f)$ since e is the minimum edge in $A \Delta B$ and each edge has a distinct weight. As such, if we create a new graph $C = B \cup \{e\} \setminus \{f\}$, we find that it too is a spanning tree (as the removal of f removes the cycle). However, you'll note that C has strictly smaller total weight than B, since $w(e) < w(f)$. This contradicts our initial assumption that there are distinct minimum spanning trees A and B, thus that assumption was false and there is only one unique MST.

Problem 7

Problem: Are the following statements true or false?

a) In an undirected graph the sum of the degrees of vertices is always twice the number of edges.

b) We can use BFS to decide if a graph is bipartite.

c) Given a complete graph on n nodes, the number of cycles with at most 4 vertices is $O(2^n)$.

d) We can use any minimum spanning tree algorithm to compute the maximum spanning tree, i.e. a spanning tree where the sum of the weights of its edges is maximum.

Solution:

a) True

b) True

c) False

d) True