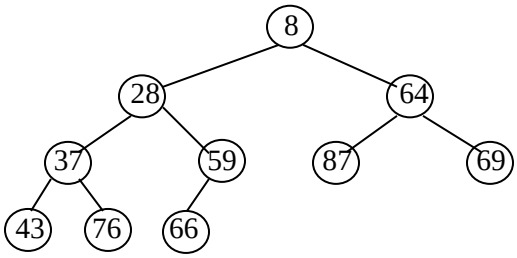
ECS 60 **Preparation for Midterm #2 Key**

Topics: ADTs, heaps, and graphs.

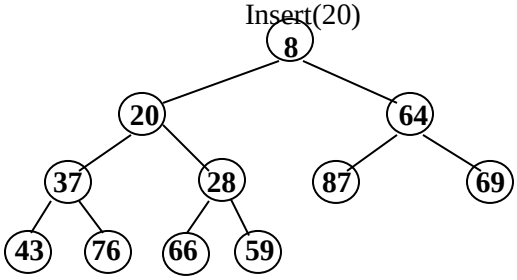
1. (0-25 points) Programs Problem. This question will attempt to determine whether you wrote the code for programming assignments 3 and 4. There is no need for you to see sample questions if you actually participated in writing the code.

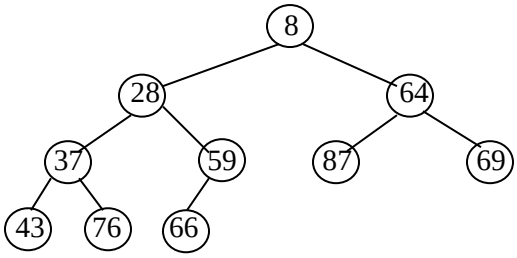
2. (30 points) Timetest3 Problem. This question will address what you learned writing your paper for timetest3.cpp. Questions could be of the form where you are given a description of a new File5.dat, and you are asked for the expected result for given ADTs compared to one of the original data files. Questions could also involve determining whether you clearly understand the source(s) of the anomalous behavior of the different ADTs.

3. (15 points) Heap Problem. There will be one problem dealing insert, deleteMin, or BuildHeap operations with a priority queue. For each of the following binary heaps, show the state of heap after operation specified.

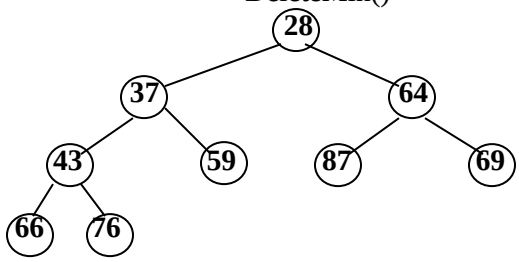


Insert(20)





DeleteMin()



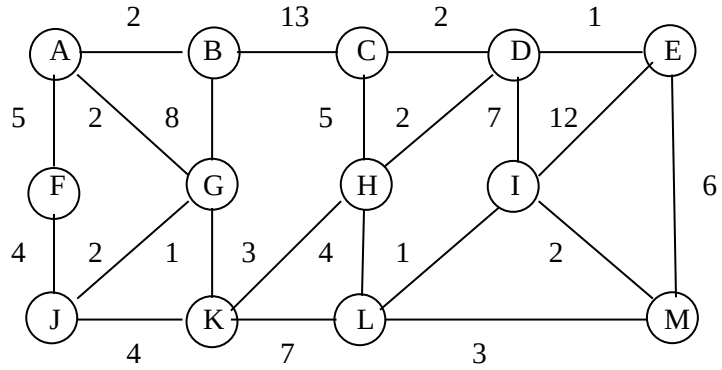
4. Graph Problems (~50 points) There will be two graph problem using one of the following formats.
   a) Minimum Spanning Tree. Fill in the following tables using Kruskal's algorithm to find the minimum spanning tree of the graph. Use union-by-size, but not path compression. For a union of sets of the same size, the new root should be the old root with the lower array index. Stop the process when you have a minimum spanning tree. Weights are to the left of internal edges.
   b)

Edge	Weight	Action
(A, B)	1	**accept**
(E, H)	2	**accept**
(D, G)	3	**accept**
(E, F)	4	**accept**
(D, F)	5	**accept**
(F, H)	6	**reject**
(A, C)	7	**accept**
(F, G)	8	**reject**
(B, C)	9	**reject**
(C, F)	10	**accept**
(A, H)	11	
(C, H)	12	
(C, G)	13	

Find-Union Array

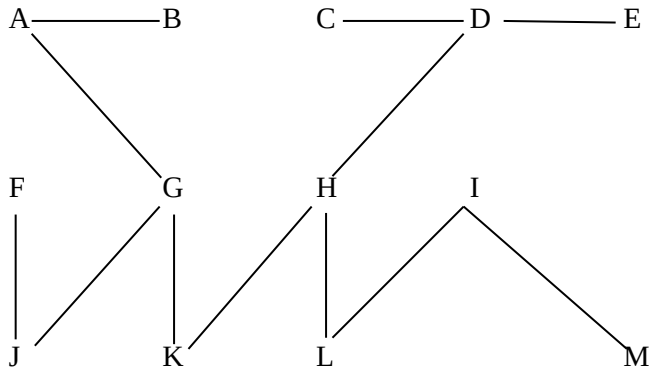
A	B	C	D	E	F	G	H
0	1	2	3	4	5	6	7
4	**0**	**0**	**4**	**-8**	**4**	**3**	**4**

b) Graph II (30 points) Fill in the table using Prim's algorithm to determine the shortest paths from vertex E. The weight of each interior edge is to the left of the edge. **(1 point for each d_v, and p_v, 4 points for all known as T.)**



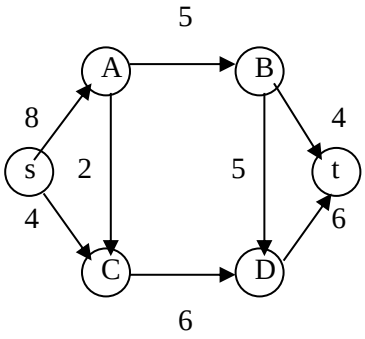
Vertex	known	d_v	p_v
A	**T**	**2**	**G**
B	**T**	**2**	**A**
C	**T**	**2**	**D**
D	**T**	**1**	**E**
E	**T**	**0**	**-**
F	**T**	**4**	**J**
G	**T**	**1**	**K**
H	**T**	**2**	**D**
I	**T**	**1**	**L**
J	**T**	**2**	**G**
K	**T**	**3**	**H**
L	**T**	**4**	**H**
M	**T**	**2**	**I**

(5 points) Based on your tables, draw the minimum spanning tree of the graph. (Don't bother with weights.)

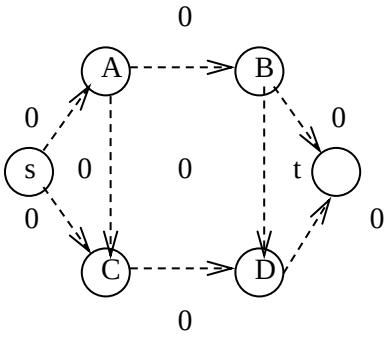


c) Network Flow. Choosing the augmenting path that allows the largest increase in flow, find the maximum network flow for the following graph, G. Internal flows are to the left of their respective edges. Your score will be based on your completion of the augmented path line, G_f and G_r , including their initial states and back-flow edges where appropriate.

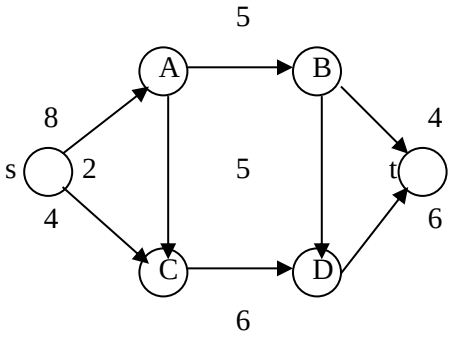
G



G_f

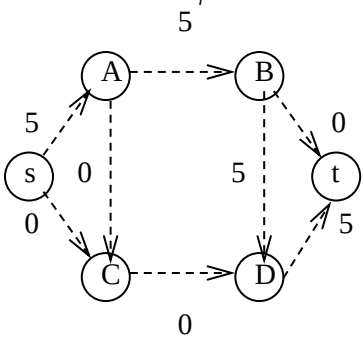


G_r

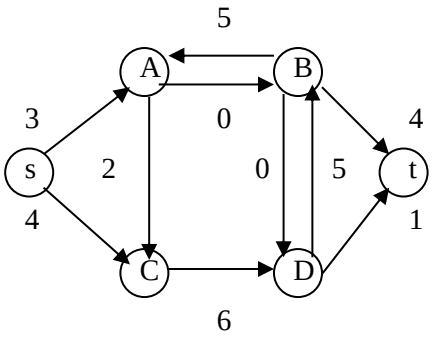


1st augmented Path: s, A, B, D, t (5)

G_f

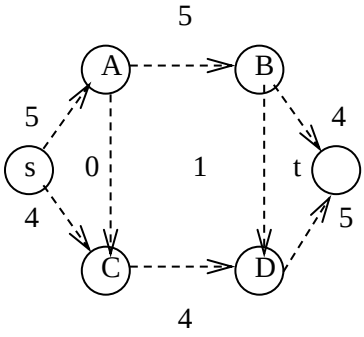


G_r

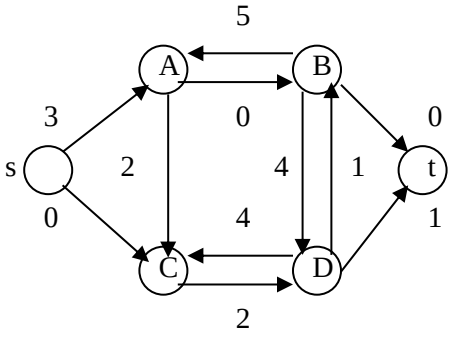


2nd augmented Path: s, C, D, B, t (4)

G_f

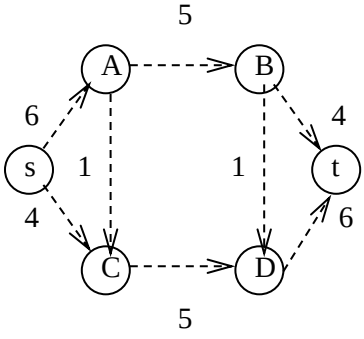


G_r

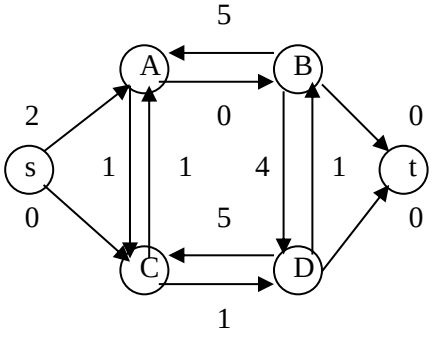


3rd augmented Path: s, A, C, D, t (1)

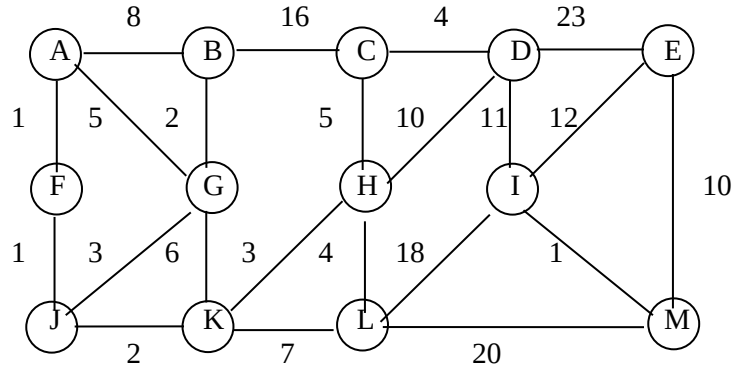
G_f



G_r

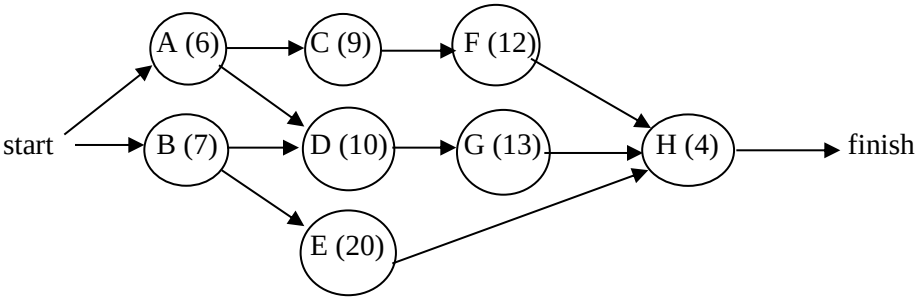


d) Shortest Path. Fill in the table using Dijkstra's algorithm to determine the shortest paths from vertex B The weight of each interior edge is to the left of the edge.

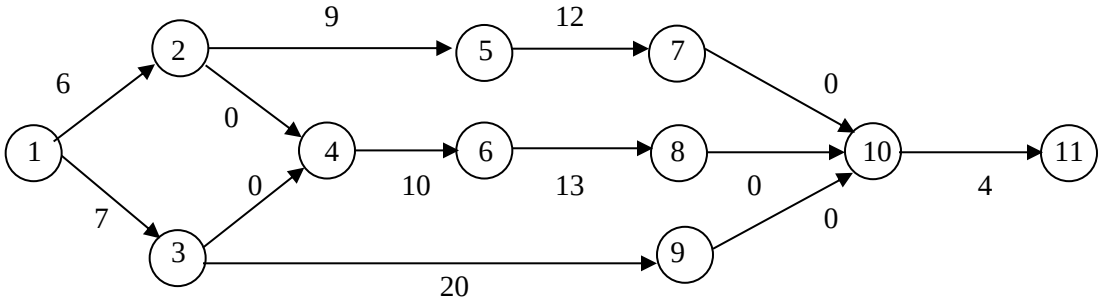


Vertex	known	d_v	p_v
A	T	7	G
B	T	0	--
C	T	15	H
D	T	19	C
E	T	41	M
F	T	6	J
G	T	2	B
H	T	10	K
I	T	30	D
J	T	5	G
K	T	7	J
L	T	14	K
M	T	31	I

e) Event Node Graph. For the following Activity-node graph:



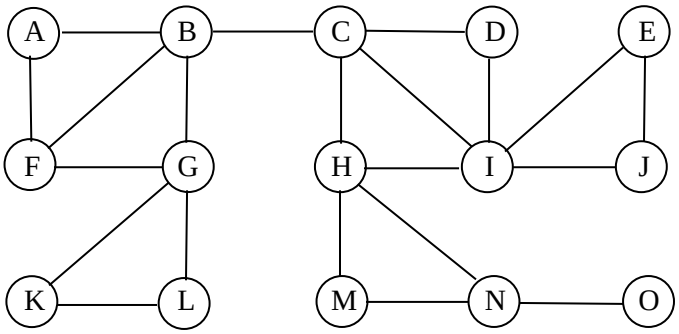
Draw the corresponding Event-node graph using numbered nodes (start numbering at left please).



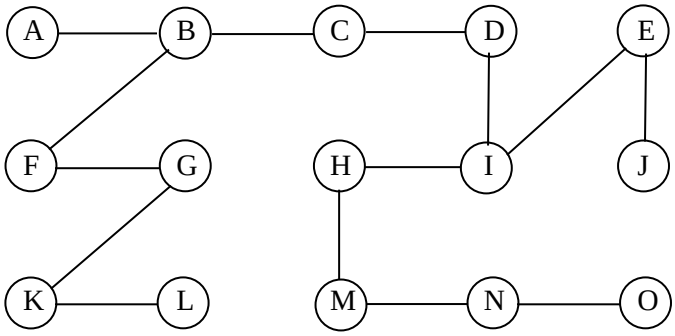
Based on <u>your</u> graph fill in the values for the following table. There may be more columns than needed.

Nodes:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Earliest completion time	**0**	**6**	**7**	**7**	**15**	**17**	**27**	**30**	**27**	**30**	**34**				
p_v	**1**	**1**	**1**	**3**	**2**	**4**	**5**	**6**	**3**	**8**	**10**				

f) (35 points) Articulation Points. Using DFS, starting at A, searching in alphabetical order whenever multiple vertices may be processed, give each vertex a number followed by its low. Then list the articulation points



(5 points) Show DFS tree(s) below. You need not show the back edges, but may if you wish.



2 points for each completely correct column

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Vertex number (5 points)	**1**	**2**	**3**	**4**	**6**	**12**	**13**	**8**	**5**	**7**	**14**	**15**	**9**	**10**	**11**
Low (5 points)	**1**	**1**	**3**	**3**	**5**	**1**	**2**	**3**	**3**	**5**	**13**	**13**	**8**	**8**	**11**
Articulation Point Y or N (5 points)	**N**	**Y**	**Y**	**N**	**N**	**N**	**Y**	**Y**	**Y**	**N**	**N**	**N**	**N**	**Y**	**N**

5. ADT and Algorithm Essay (45-50 points) This an ECS 60 problem placed within a familiar context. For the war in Iraq, the Defense Department had to plan how and when to transport troops and materiel to Iraq so that they were all there and operational on March 1st. However, having units arrive earlier than needed would degrade their performance. For each of the 1000 battalions, the planners were given a list of the battalions that it directly required to be operational before that battalion could start setting itself up. It took each battalion 10 days to become operational once ALL of the listed battalions upon which it was dependent were operational.

   The planners had 100 ships that took 30 days to travel from the U.S.A. to Kuwait, and 30 days to travel from Kuwait to the U.S.A. Each ship could carry five battalions. How would you approach the planner's problem of scheduling battalion departures? Specifically, describe the ADT(s) and algorithms you would use to determine: 1) The order of departure; and 2) the time of departure of each battalion. Provide Big-Ohs wherever possible. For the Big-Os describe what V, E, N, etc. relate to in this problem. (Note that sorting N items will take NlogN time.)

   Algorithm selection : 8 points.
   Description of graph(s) : 8 points
   Graph representation : 2 points
   Big–O analysis : 8 points
   Explanation of departure ordering process: 12 points.
   Explanation of departure times process: 12 points

This is a two phase question. First, I can create an event node graph using an adjacency list. Each battalion would have three events associated with it: #1) Departing U.S.; #2) Arriving in Kuwait; and #3) Becoming operational. There would be a single edge of 30 days between node #1 and node #2 of each battalion. There would also be edges of 0 weight leading back from node #2 to the node #3 of all the battalions upon which it was dependent. There would be a single edge of 10 days between node #2 and node #3 each battalion. All node #1 will have an incoming edge from a single starting node.

Having created the event node graph, I would then do a critical path analysis. This requires a topological sort of the battalions. Since there should be no cycles in the dependencies, I can use a queue and do the sort in O(V + E) where V is the number of battalions, and E is the total number of items on the dependency lists plus 2 * V.

I can use the latest start times for node #1 from the analysis to determine the ordering of departure of the battalions. Unhappily, this ordering does not have to match the topological sort. To determine the sorted order will take N logN time where N is the number of battalions.

You will note that after removing the starting node, none of the node #1 has an incoming edge! This would indicate that all could depart at the same time. But the 100 ships can only carry 500 battalions at a time. To allow for the time for the ships to return to U.S.A., I will add an edge of 30 days from node #2 of the first node in the ordering to the node #1 of the 501^{st} node in the ordering. I will add similar edges from the 2^{nd} node to the 502^{nd} node and so on. The latest starting times determined by a critical path analysis of this new graph, will now be accurate. Note that ordering of the battalions will be the same as before.

The planners can now group five battalions together based on their ordering, and have their ship depart on the date of latest start time of the battalion earliest in the ordering.