1) 10 points

This problem works through the memory tradeoff of representing a List as a statically-allocated array (an array that does not grow or shink) versus a single linked list. We assume the size of the list is large enough that any object overhead is negligible compared to the memory needed to store the List elements.

We define four variables:

n : the number of items currently in the List
C : the maximum capacity of the List
D : the size of a data element
E : the size of a reference to another node

a) In terms of n, C, D, and E, how much space is used by the statically-allocated array?

$$CD$$

b) In terms of n, C, D, and E, how much space is used by the single linked list?

$$n(D+E)$$

c) In terms of C, D, and E, at what value of n do the two representations use the same amount of memory?

$n(D+E) = CD$ $\qquad$ $n = \dfrac{CD}{D+E}$

d) If D equal E (as with 4-byte integer data elements and 4-byte references), how large must n be for the statically-allocated array to be more space efficient?

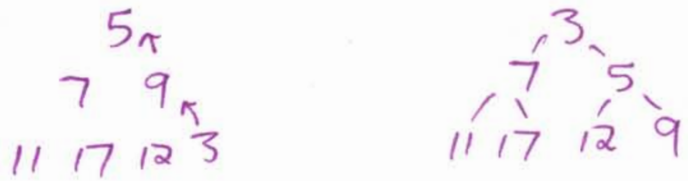
$$n \geq \frac{C}{2}$$

Unique ID: 55

2) 10 Points

Starting with an empty binary min-heap:

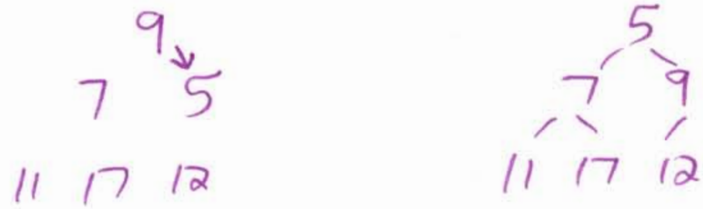
a) Insert, one at a time, the sequence 5, 7, 9, 11, 17, 12, 3. Draw the resulting min-heap.



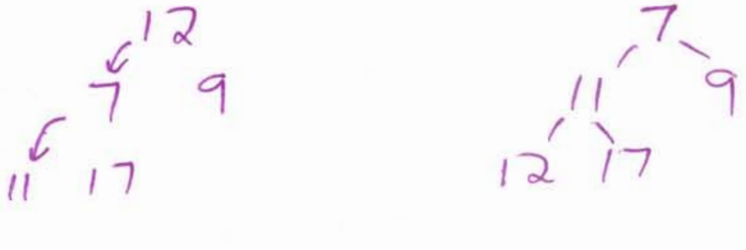
b) Draw an array representation of your binary min-heap from a).

7 | 3 7 5 11 17 12 9

c) Perform a deleteMin on your binary min-heap. Draw the resulting binary min-heap.



d) Perform a deleteMin on your binary min-heap. Draw the resulting binary min-heap.



e) If you started with your min-heap from a), and continued performing deleteMin until your min-heap was empty, in what order would you delete each key from the heap?

3 5 7 9 11 12 17

f) What is the worst-case O (Big-Oh) cost of creating a heap from an array containing N keys, then repeatedly executing deleteMin until that heap is empty?

$O(N \log N)$

g) What is the name for this procedure?

Heap sort

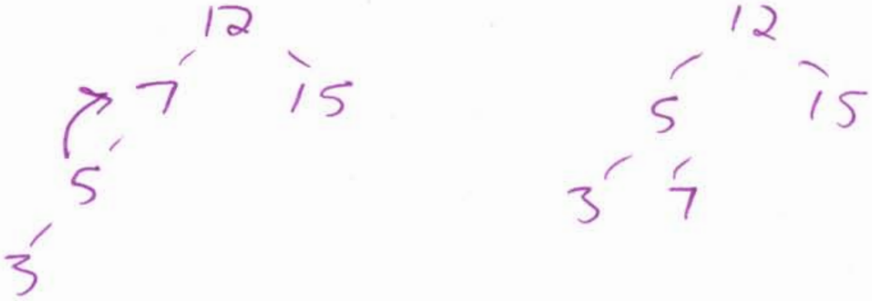
h) If you want to perform the procedure from g) *in-place*, yielding entries in your array in the same order as e), while minimizing any pre-processing or post-processing overhead, what would you do differently?

Use a max-heap, placing deleted entries at the first non-heap location in the array at each step

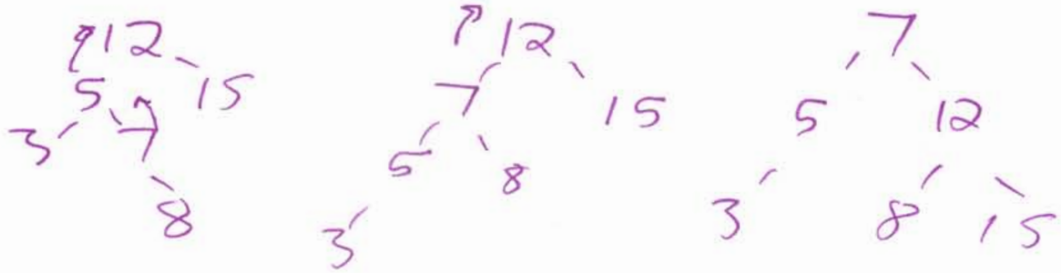
3) 10 points

Starting with an empty AVL tree:

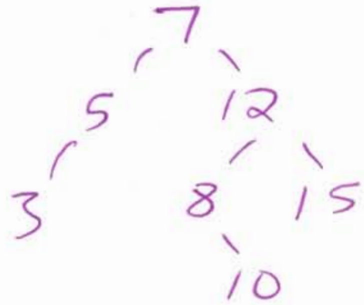
a) Insert the sequence 12, 7, 15, 5, 3. Draw the final AVL tree.



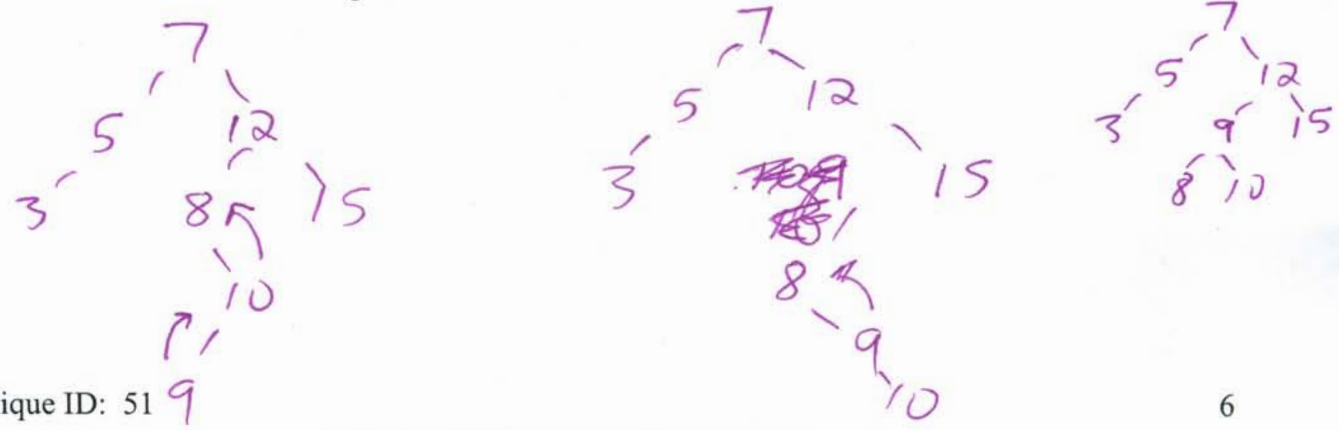
b) Insert 8. Draw the resulting AVL tree.



c) Insert 10. Draw the resulting AVL tree.

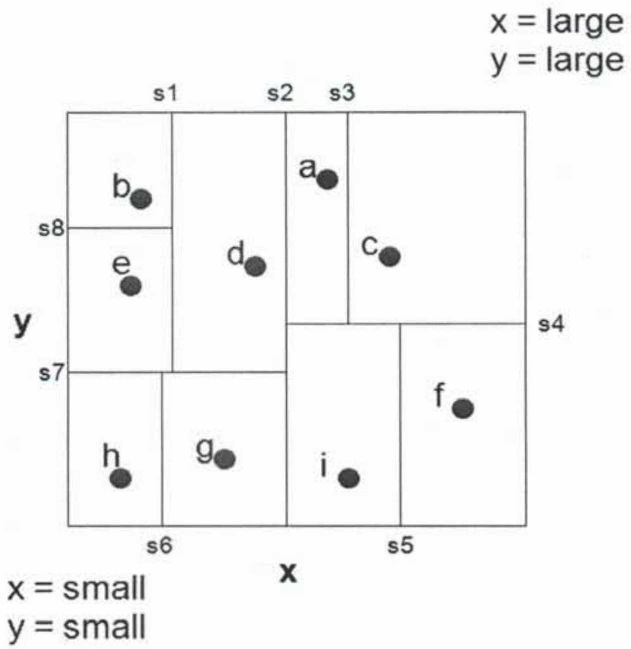


d) Insert 9. Draw the resulting AVL tree.

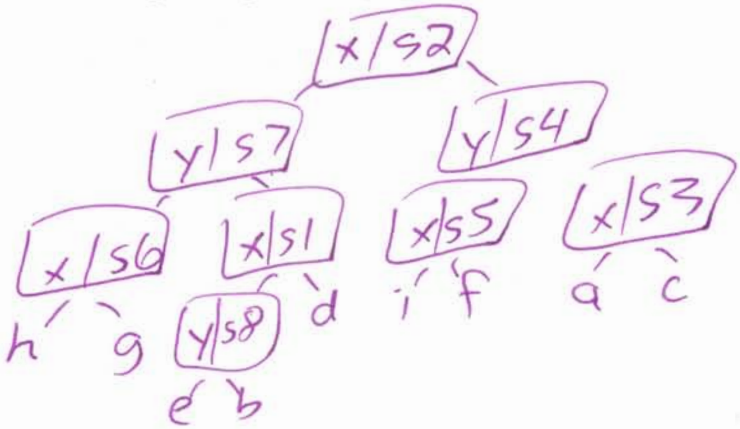


4) 10 points

Consider this visualization of a k-d tree in two dimensions, x and y. The lower-left corner corresponds to lesser values of both x and y, the upper-right corner corresponds to greater values of both x and y. Each object has a label (between a and g), and each split point has a label (between s1 and s8).



a) Draw a k-d tree corresponding to this split of this data.



b) Is there more than one correct answer to a)? Why?

No. At each point only one potential split corresponded to figure.

(this is a better question if you had to choose among two edges at same point)

5) 10 Points

You have a lot of turkeys, and you need to keep track of vacation and dinner plans for each of them. You will store information about each turkey in a Dictionary implemented using a B-tree (specifically, the variant of B-trees discussed in class and in Chapter 4).

Assume your B-tree will run on an old farm machine that stores and retrieves data in blocks of 256 bytes. Further assume that that storing a reference requires 4 bytes. Assume that you are *not* storing sibling or parent references within the tree.

Each turkey wears a stylish collar with a unique identifier encoded in 8 bytes. For each turkey, a record of their breeding, their feeding, and their favorite type of gravy requires 32 bytes (not including the turkey's identifier). Turkeys are equally happy with all vacation destinations, so you do not store any information about this.

You should provide a numeric answer for questions a) and b), but providing the formula you used to obtain your answer will allow us to give full or partial credit if your answer is incorrect due to a simple arithmetic error.

a) Choose an appropriate value for *M*, the branching factor of internal nodes.

$4M + 8(M-1) \leq 256$

2 $12M - 8 \leq 256$ $M = 22$

$12M \leq 264$

b) Choose an appropriate value for *L*, the number of elements per leaf.

$8\cancel{4}L + 32L \leq 256$

2 $40\ \cancel{36}L \leq 256$ $L = \cancel{7}6$

c) In terms of *M*, *L*, and *h*, what is the maximum number of turkeys about which you can store information in a B-tree of height *h*?

3 h inner levels → M^h nodes one level above bottom

$L * M^h$

d) In terms of *M*, *L*, and *h*, what is the minimum number of turkeys you must have in order for your B-tree to be of height *h*?

3

everybody is at least half full except root could just have 2

$2 * \left\lceil \frac{L}{2} \right\rceil * \left\lceil \frac{M}{2} \right\rceil^{h-1}$

is h-1 because of the 2

6) 10 points

Using a hash table of size 11 with open addressing and linear probing, perform the following sequences of actions. Objects can be represented simply by their hashcodes. Do not grow the size of the hashtable.

+3

a) Hash a sequence of objects with hashcodes 36, 27, 47, 19.

0	1	2	3	4	5	6	7	8	9	10
			36	47	27			19		

36%11=3   27 %11 = 5   47%11 = 3   19 %11 =8

b) Delete the object with hashcode 36.

+2

0	1	2	3	4	5	6	7	8	9	10
			×	47	27			19		

↑
marker

+3

c) Insert a sequence of objects with hashcodes 90, 35.

0	1	2	3	4	5	6	7	8	9	10
		90	35	47	27			19		

90%11 = 2   35%11 = 2

d) What is the load factor of your table after c)?

+2

$\frac{5}{11}$

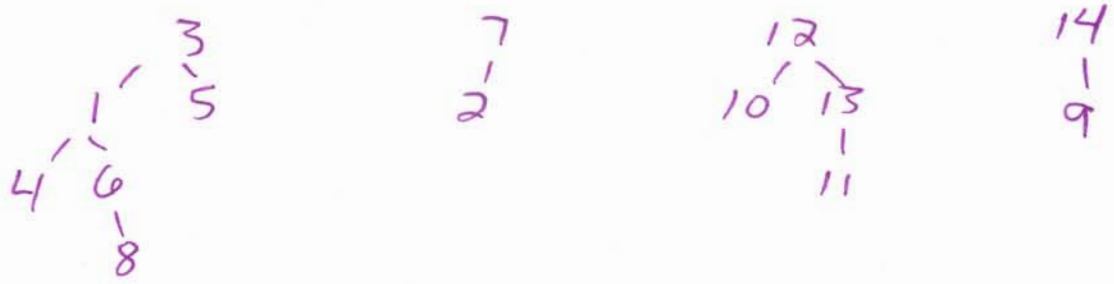
7) 10 points

The uptrees used to represent sets for manipulation by union-find algorithms can be stored in a single *n*-element array, with positive numbers representing parent pointers and negative numbers representing the weight of a root. Consider this collection of sets:

1	2	3	4	5	6	7	8	9	10	11	12	13	14
3	7	-6	1	3	1	-2	6	14	12	13	-4	12	-2

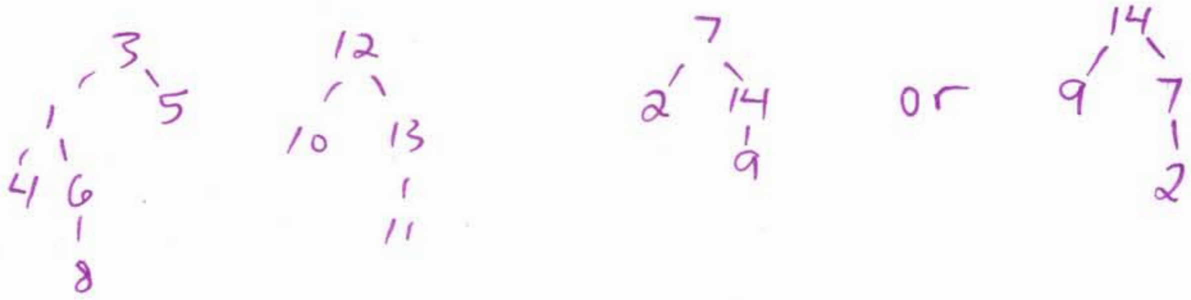
You will manipulate this forest of uptrees, assuming that find operations apply path compression and union operations apply union-by-size.

a) Draw a picture of the uptrees represented by the data in the above arrays.



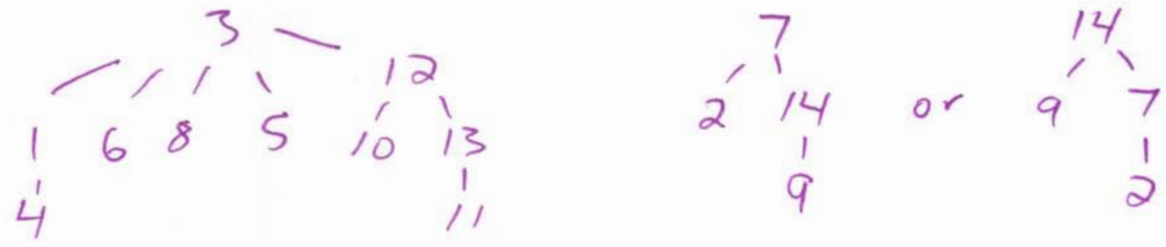
b) Draw the uptrees that result from executing:

```
union(find(2), find(14));
```



c) Continuing based on your result for b), draw the uptrees that result from executing:

```
union(find(13), find(8));
```



d) Show the contents of the array after completing c).

1	2	3	4	5	6	7	8	9	10	11	12	13	14
3	7	-10	1	3	3	-4	3	14	12	13	3	12	7
						14		14					-4

e) Is there more than one correct answer for d)? Why?

Yes. Arbitrary choice was made in b) regarding which tree to keep as root (because of their equal size).

8) 10 points

Prove by induction:

An undirected graph, without loop edges, has at most $\frac{|V|(|V|-1)}{2}$ edges.

Base case: V=1 E=0 O

V=2 E=1 O—O

These graphs are complete. No more possible edges.

Assuming true for V, Show for V+1

Add new vertex, add edge to every existing vertex. Total of V new edges.

Actual Edges $\frac{|V|(|V|-1)}{2} + |V| = \frac{|V|^2 - |V| + |v|}{2} = \frac{|v|^2}{2}$

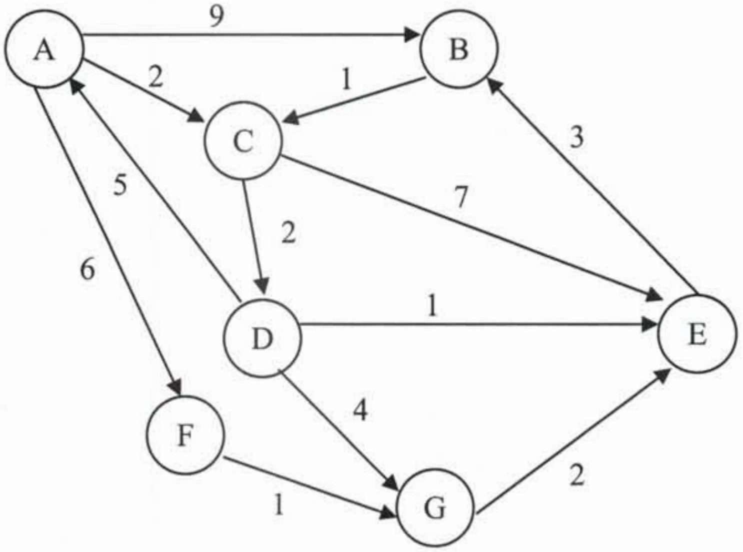
$\frac{(V+1)(V+1-1)}{2} = \frac{(V^2+|V|}{2}$

Allowable Edges with V+1

$\frac{|V|^2}{2} ? \frac{|V|^2 + |V|}{2}$ $0 < \frac{|V|}{2}$ for $|v| > 0$

9) 10 points

Consider the following directed, weighted graph:



a) Step through Dijkstra's algorithm to calculate the single-source shortest paths from *A* to every other vertex. Show your steps in the table below. Cross out old values and write in new ones, from left to right within each cell, as the algorithm proceeds. Also list the vertices in the order which Dijkstra's algorithm marks them known:

Known vertices (in order marked known): A C D E F G B

Vertex	Known	Distance	Path
A	✗	0	–
B	✗	~~9~~ 8	~~A~~ E
C	✗	2	A
D	✗	4	C
E	✗	~~9~~ 5	~~C~~ D
F	✗	6	A
G	✗	~~8~~ 7	~~D~~ F

b) What is the lowest-cost path from *A* to *E* in the graph, as computed above?

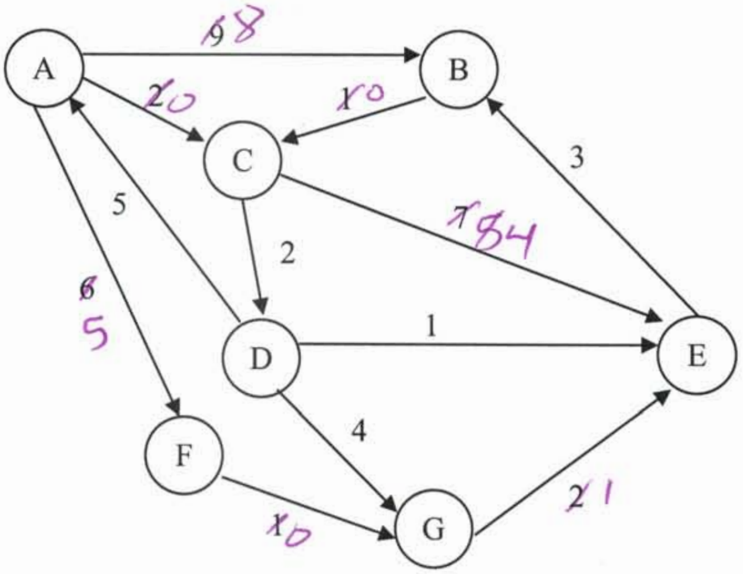
A → C → D → E

c) Is there more than one correct table for a)? Why?

No. Distances were unique at each step. Only one minimum available, no arbitrary choice.

10) 10 points

Consider again the following directed, weighted graph:



You will need to compute a maximum flow from *A* to *E*, using the Ford-Fulkerson method. This page is for your work. Before running the algorithm, read ahead to know what information you need to provide for the different parts of this problem.

a) List each augmenting path you discover, in the order that you discover them, as well as the volume you are able to send down that path.

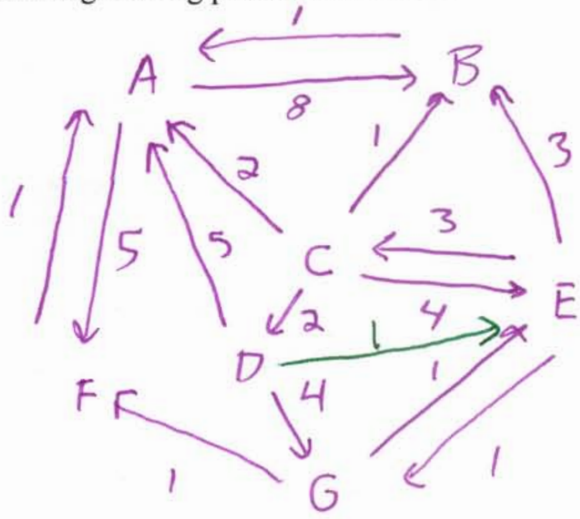
~~2~~ ABCE:1 A → C → E 2

~~3~~ ACE:2 A → B → C → E 1

2 AFGE:1 A → F → G → E 1

b) Draw the residual graph that you have at the completion of the algorithm, when no more augmenting paths are available.

6 pts
~~4~~
~~3~~
4



4 : perfect
3 : minor typos
2: major typos
1: no back-edges
0: no graph

c) Is there more than one correct answer for b)? Why?

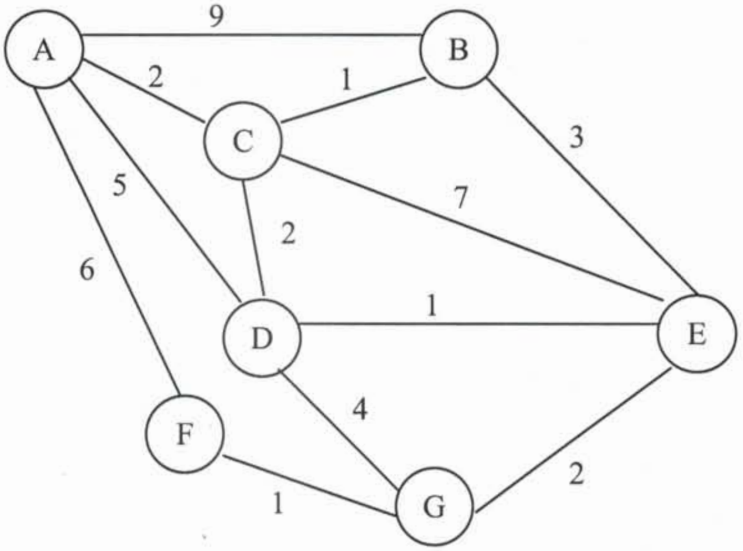
2 Yes, multiple paths available. Instead of
A, B, C, E
I could have chosen
A, B, C, D, E

d) Give a minimum cut of the vertices in the graph, with source *A* and sink *E*.

2 {A, B, F} {C, D, E, G}

11) 10 points

Consider the following undirected, weighted graph:



6

a) Step through Prim's algorithm to calculate a minimum spanning tree. Show your steps in the table below. Cross out old values and write in new ones, from left to right within each cell, as the algorithm proceeds. Also list the vertices in the order which Prim's algorithm marks them known:

Known vertices (in order marked known): A C B D E G F

Vertex	Known	Distance	Path
A	x	–	–
B	x	9 1	A C
C	x	2	A
D	x	5 2	A C
E	x	7 3 1	C B D
F	x	6 1	A G
G	x	4 2	D E

2 b) What are the edges in the minimum spanning tree, as computed above?

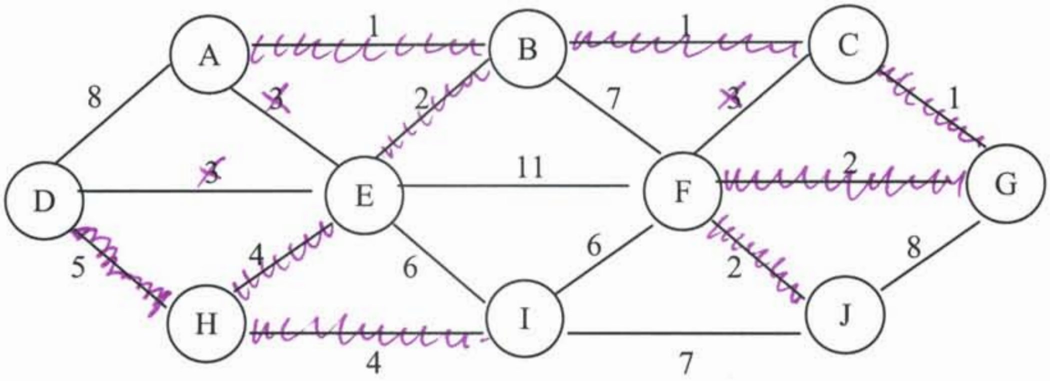
B,C C,A D,C E,D F,G G,E

2 c) Is there more than one correct table for a)? Why?

Yes. Start vertex is arbitrary.

12) 10 Points

Consider the following undirected, weighted graph:



Apply Kruskal's algorithm to compute a minimum spanning tree. In the designated spaces below, write down the edges in the order they are considered by Kruskal's algorithm. If the edge is part of the minimum spanning tree found by the algorithm, write it down in the first list of edges that form the MST. Write down the other edges considered by the algorithm in the order they were considered. Assume that the algorithm terminates as soon as the MST has been found.

a) Edges that form part of the MST, in order considered:

1 (A,B B,C C,G) (B,E F,G F,J) 2 (D,E) 3

4 (H,E H,I) ~~(D,H)~~ 5

b) Other edges considered, but not included in the MST, in order considered:

3 (A,E ~~(D,E)~~ C,F)

c) Is the MST for this graph unique? Why?

Yes. In each set of edges with the same weight, we needed all or none of them in the MST. No arbitrary choice was made.

+2

Do **not** write any confidential information on this page.

2 Points Extra Credit for the Best Response in the Class

Why did the turkey go to Pullman?

Unique ID: 55