Midterm 2

Name:

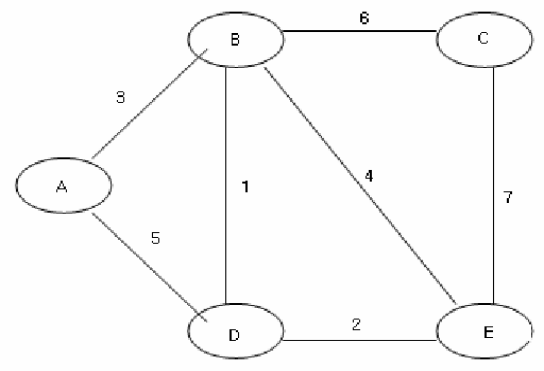
TA:

Answer all questions. Read them carefully first. Be precise and concise. The number of points indicate the amount of time (in minutes) each problem is worth spending. Write in the space provided, and use the back of the page for scratch. Good Luck!

1	
2	
3	
4	
total	

Problem 1

(20 points)



In the graph shown above:

(a) In what order are the vertices deleted from the priority queue in Dijkstra's algorithm for the shortest path? (Start node: A.)

(b) In Prim's algorithm for the minimum spanning tree? (Start node: A.)

(c) In what order are the edges added in Kruskal's algorithm?

(d) show the union-find trees at the end of Kruskal's algorithm (in case of a tie in rank, the lexicographically first node becomes root).

Problem 2

True or false? Provide a brief explanation. The space provided should suffice.

1. (3 points) Suppose all edge weights are different. Then the shortest edge must be in the minimum spanning tree.

2. (3 points) Suppose all edge weights are different. Then the longest edge cannot be in the minimum spanning tree.

3. (3 points) Suppose all edge weights are different. Then the minimum spanning tree is unique.

4. (3 points) Suppose all edge weights are different. Then the shortest path from A to B is unique.

5. (3points) There is an efficient algorithm for finding the longest (highest length) path in a dag.

6. (3 points) There is an efficient algorithm for finding the longest (highest length) path in a graph with only positive weights.

7. (3 points) There is an efficient algorithm for finding longest (highest length) paths in a graph with only negative weights.

8. (3 points) Any operation in the union-find data structure takes at most O(log n) time.

9. (3 points) Even without path compression, any operation in the union-find data structure takes at most O(log n) time.

10. (3 points) Any operation in the union-find data structure takes at most O(log* n) time.

Problem 3

(15 points) Suppose that, in the single-source shortest path problem, we wish to find not just any shortest path, but among those the shortest path that has the fewest hops.

There are at least two ways to do this: (a) Modify slightly Dijkstra's algorithm by adding an array (besides **`dist`** and **`prev`**) and adding a line to the update step, or (b) Apply the original Dijkstra algorithm but with the edge weights changed slightly to reflect our new interest in few hops.

Describe briefly one of these ways (or, if you prefer, your own new idea for an algorithm for this problem).

Problem 4

(15 points) Given an array of integers $a_1,a_2,\ldots,a_n$, positive and negative, such as -1, 3, 2, -7, 4, 2, -2, 3, -1, you want to find the largest sum of contiguous integers; in this case it would be 4 + 2 – 2 + 3 = 7.

We can accomplish this by (you guessed it!) dynamic programming. For each i = 0,…,n define **`maxsum[i]`** to be the value of the largest sum seen so far. We also need **`maxsuff[i]`** to be the largest sum of a suffix ending at a_i. (In the array above, **`maxsum[4]`** = 5, **`maxsuff[4]`** = 0).

Fill in the blanks in the dynamic programming algorithm:

(a) initialize: **`maxsum[0]`** = **`maxsuff[0]`** =

(b) iteration: for i = 0, …, n-1,
`maxsum[i+1]` =

`maxsuff[i+1]` =

(c) What other data structure do you need in order to recover in the end the beginning and end of the maximum contiguous sum?

(d) What is the running time of the algorithm?