CSE 373: Practice Final

1 Short Answer

a) Provide two orderings [0,1,2,3,4,5,6,7] that are worst-case for quick sort. Assume that you select the first element as the pivot. Explain why this is the worst-case. You do not have to perform the quicksort

b) Explain why a divide-and-conquer approach would or would not be effective in finding an element in an unsorted array.

c) You are given a batch of UW ID numbers. Given that they are all seven digits, and that the first two numbers correspond to the year of the student's entry, propose a method to put these into sorted order. Provide the time complexity of your algorithm and explain why you believe it is fastest.

d) Provide and explain the two types of locality relevant to caching and memory accesses.

e) For each of the following sorts, explain their best-case and worst case runtime, whether or not they are stable (in the method given in class), whether they can be halted to provide the top k elements and whether or not they are done in place.

Insertion sort:

Selection sort:

Merge sort:

Quick sort:

Heap sort:

f) For the Uptree data structure, show the worst-case runtime for find() with and without the weighted union optimization and with and without the path compression optimization. In all, you should provide four worst-case runtimes.

g) Provide pseudocode for an iterator that returns all of the vertices that are exactly two edges away from an input node in an undirected, unweighted graph.

h) Explain the difference between primary and secondary clustering in hash tables. Which types of clustering are relevant for which probing techniques?

i) Draw an unweighted graph of at least five vertices which has two unique minimum spanning trees.

2 Big O notation

For the following functions, determine the tightest bigO upper bound in terms of n. Write your answers on the line provided.

a)
```
void f1(int n) {
    int i = 1
    while(i < n^4){
        j = n;
        while (j > 1) {
            j = j / 2;
        }
        i = i + n
    }
}

O(______)
```

b)
```
void f2(int n) {
    for(int i=0; i < n; i++) {
        for(int j=0; j < 10; j++) {
            for(int k=0; k < n; k++) {
                for(int m=0; m < 10; m++) {
                    System.out.println("!");
    }   }   }   }
}



O(______)
```

c)
```
int f3(int n){
    if (n < 10) return n;
    else if(n < 1000) return f3(n-2);
    else return f3(n/4)+f3(n/4);
}

O(______)
```

d)
```
int f4(int n){
        if(n < 1000){
             return n;
        }
        if (n%5 == 0){
             return f4(n/2)+f4(n/2)+f4(n/2);
        } else {
             return f4(n-2);
   }

   O(______)
```

3 AVL insertions

Show an AVL tree after performing the following inserts:

$$[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$$

You only need to show the final result, but showing intermediate steps may earn you partial credit if a mistake is made.

Provide the pre-order, in-order, post-order and BFS traversal for the AVL tree you produced

Pre-order:

In-order:

Post-order:

Breadth-first Search:

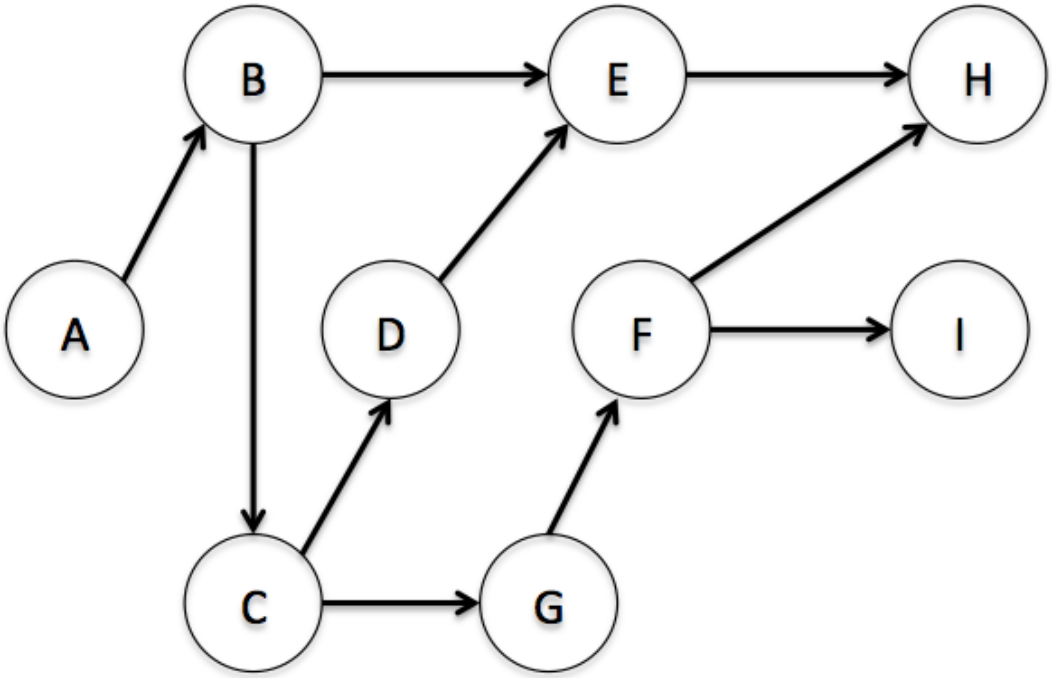
4 Debugging

Debug the following Java code which attempts to implement a Queue. Circle any incorrect areas and provide the correct portion. Syntax is not important.
Additionally, provide a sequence of enqueues and dequeues which would get this TestQueue to exhibit unexpected behavior.

```
public class TestQueue{
    String[] data;
    int fp;
    int lp;
    public TestQueue(){
        data = new String[10];
        fp = 0;
        lp = 0;
    }
    public void enqueue(String toInput) {
        data[lp] = toInput;
        lp = (lp + 1)%data.length;
    }
    public String dequeue(){
        if(fp!=lp){
            String toRet = data[fp];
            fp = (fp + 1)%data.length;
            return toRet;
        }
        return null;
    }
    public String front(){
        return data[fp];
    }
}
```

5 Topological Sort

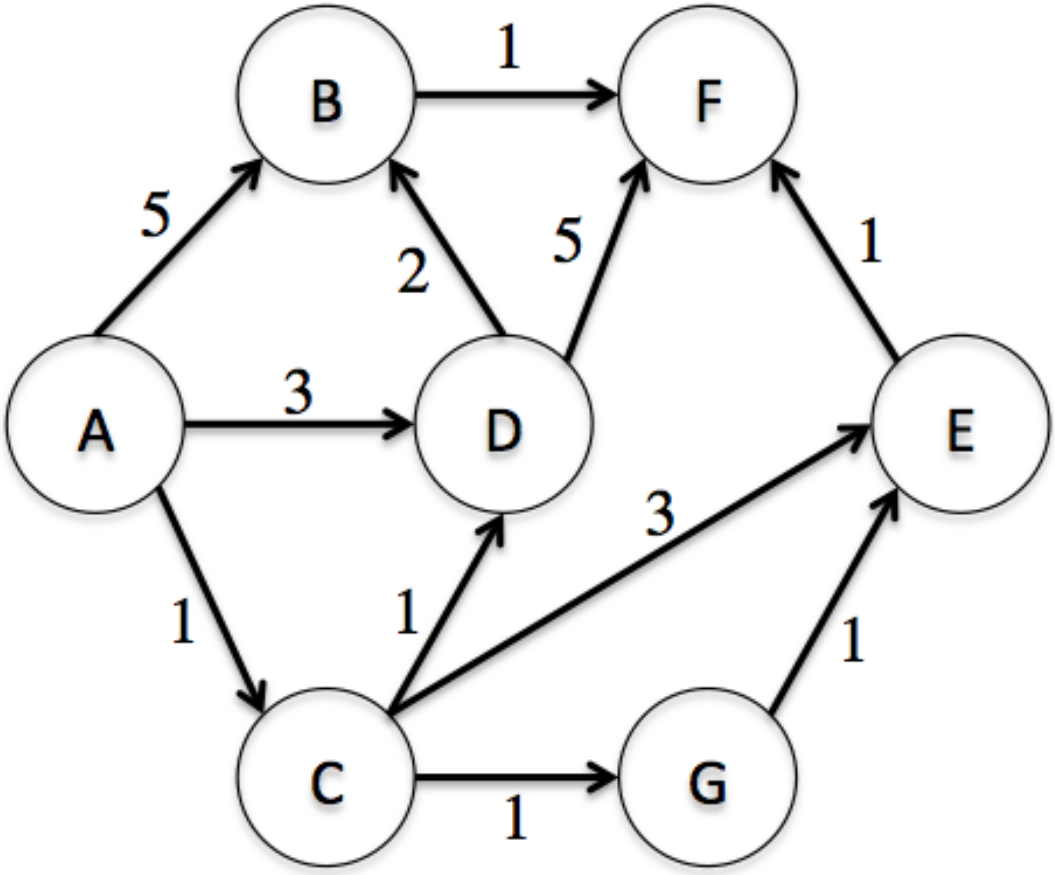
Provide two topological orderings for the following graph.



Explain how you could modify the topological sort algorithm to find cycles in directed graphs.

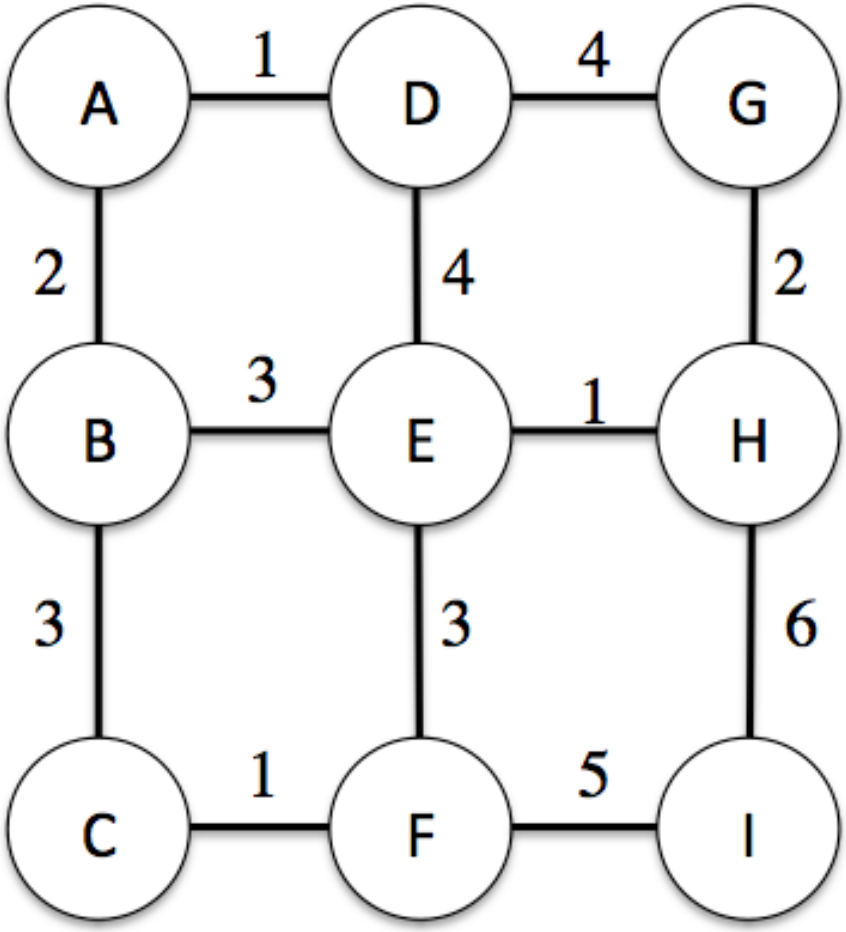
6 Dijkstra's Algorithm

Use Dijkstra's algorithm to provide the shortest path between nodes A and F. Show each of your steps. In your work, indicate in which order vertices are added to the known set.



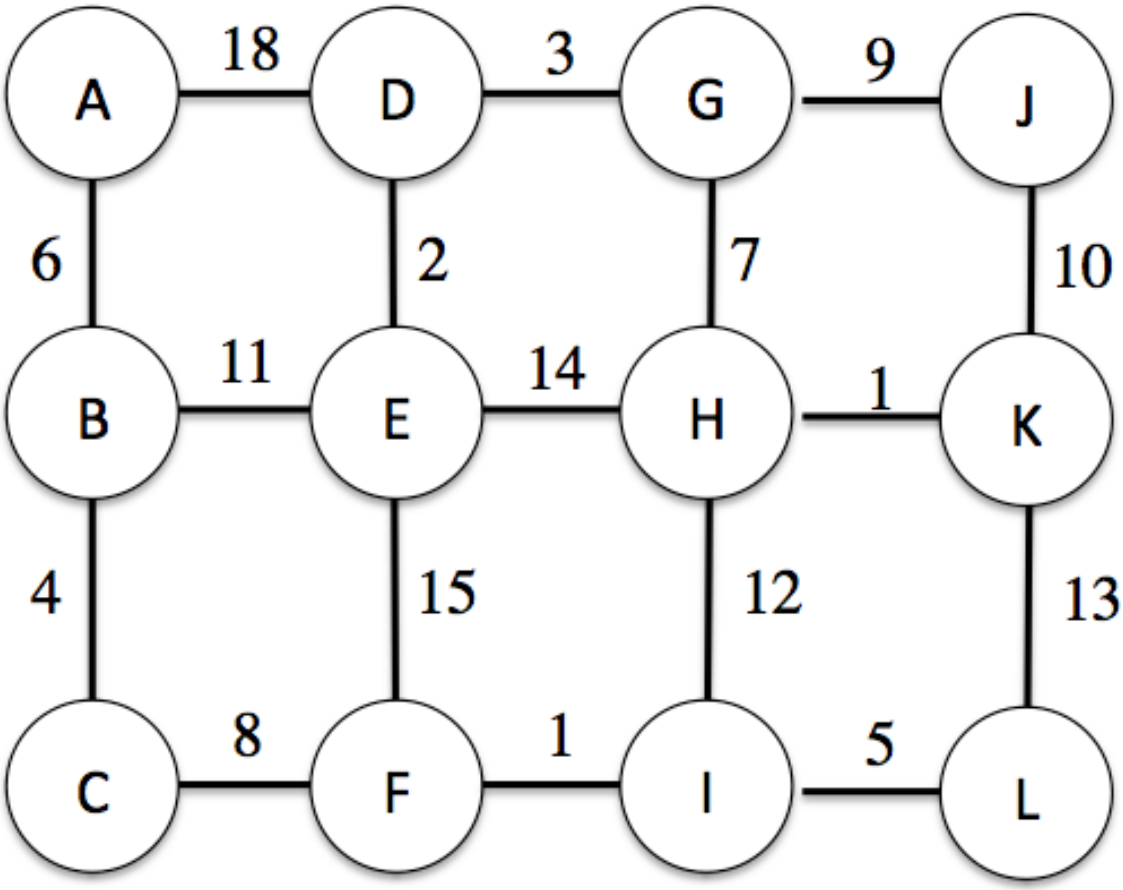
7 Prim's Algorithm

Find the minimum spanning tree of the following graph using Prim's algorithm. Show each of your steps and indicate which edge is added to the MST at each point.



8 Kruskal's Algorithm

Find the minimum spanning tree of the following graph using Kruskal's algorithm. Provide your ordering of edges. If an edge cannot be added to the MST, provide the cycle that would be created if that edge were added.



9 Union Find

Show a union find data structure that uses weighted unions and path compressions after the following operations. To start, there are 10 elements, named $[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]$ if there is a tie when performing a union, the first element should become the representative. Show the array at each step. Assume that a call to union will find the representative before performing the union.

```
union(1,4)

union(1,5)

union(2,6)

union(2,1)

find(6)

union(8,7)

union(0,9)

union(9,8)

union(3,1)

union(9,3)

find(3)
```

10 Algorithm Design

Design an algorithm, which given two integers (i, j) greater than two, determines whether they have any common factors. Recall that if two numbers are both prime and not equal to each other, then they have no common factors. Also, if a number k is composite (not-prime) then it must have one factor that is at most $\sqrt{k}$.

Provide pseudocode for this algorithm and then explain the runtime of your algorithm. Additionally, if it needs more than a constant amount of additional memory, explain this memory usage.

Continue your solution here, if necessary:

You may use this page as scratch paper, or as an extra page for a previous solution. If so, indicate on that page that your work continues here.