CS/ENGRD2110: Prelim 2

19th of April, 2011

NAME : ____

NETID: ________________

- The exam is **closed book and closed notes**. Do not begin until instructed. You have **90 minutes**. Good luck!
- Start by writing your name and Cornell netid on top! There are **11 numbered pages**. Check now that you have all the pages.
- Web, email, etc. may not be used. Calculator with programming capabilities are not permitted. This exam is **individual work**.
- We have **scrap paper** available, so you if you are the kind of programmer who does a lot of crossing out and rewriting, you might want to write code on scrap paper first and then copy it to the exam, just so that we can make sense of what you handed in!
- Write your answers in the space provided. Ambiguous answers will be considered incorrect. You should be able to **fit your answers easily into the space we provided**. Answers that are not concise might not receive full points. It you do need more space, use the back page of the exam.
- In some places, we have abbreviated or condensed code to reduce the number of pages that must be printed for the exam. In others, code has been obfuscated to make the problem more difficult. This does not mean that its good style.

POINTS:

Dictionaries and Hashtables	______	/ 12
Stacks/Queues	______	/ 9
Priority Queues and Heaps	______	/ 21
Graphs	______	/ 16
Graph Search	______	/ 13
Minimum Spanning Trees	______	/ 16
Graphical User Interfaces	______	/ 13
	===========	
Total	______	/100

1 Dictionaries and Hashtables

1. If you were given the job to write a hash function for storing names(strings), which one of the following hash functions is the BEST:

   2 pts.

   (a) multiply the integer value of each character in the string by their respective index in the string and return the sum of these products

   (b) return the int value of the middle character of the string

   (c) sum up the integer value of each character in the string

   (d) return the value of Math.Random (i.e. a random number) rounded to the closest integer

2. Mark all properties that are NOT TRUE for a hashtable with n elements?

   4 pts.

   (a) an ideal hash table using array doubling has amortized (over all n elements) time complexity of $O(1)$ for insert

   (b) an ideal hash table using array doubling has amortized (over all n elements) time complexity of $O(1)$ for lookup time

   (c) it can be implemented using two nested linked lists without loss in efficiency

   (d) can perform the array doubling operation in time $O(1)$, implying that every individual insert operation has time complexity $O(1)$.

   (e) it is possible to have different keys being hashed to the same position in the array

3. What properties must a hash function $h : Key \longrightarrow [0..m-1]$ have so that the JAVA data structure `Hashtable<Key,Integer>` works CORRECTLY?

   6 pts.

2 Stacks and Queues

1. Answer the following questions with either true of false. No explanation necessary.

   4 pts.

   - It is possible to implement a Queue so that both insertions and extractions can be done in time O(1).
   - It is possible to implement a Stack so that both insertions and extractions can be done in time O(1).

2. Using the following sequence of push (i.e. insert) and poll (i.e. extract) operations, demonstrate how two stacks can be used to implement a queue. In particular, show the content of the two stacks *AFTER* each operation is completed.

   5 pts.

• push(x): Stack A=	Stack B=
• push(y): Stack A=	Stack B=
• push(z): Stack A=	Stack B=
• poll(): Stack A=	Stack B=
• push(w): Stack A=	Stack B=
• poll(): Stack A=	Stack B=
• push(v): Stack A=	Stack B=
• poll(): Stack A=	Stack B=
• poll(): Stack A=	Stack B=
• poll(): Stack A=	Stack B=

3 Priority Queues and Heaps

1. In Java, the predefined class `PriorityQueue<E>` is implemented using which of the following data structures:

   3 pts.

   (a) Sorted list
   (b) Heap
   (c) Binary search Tree (BST)
   (d) Hash Table

2. For a binary heap with n elements, the time complexity of the push operation (i.e. insert) is:

   4 pts.

   (a) $O(log(n))$
   (b) $O(n)$
   (c) $O(nlog(n))$
   (d) $O(n^2)$

3. For a binary heap with n elements, the time complexity of the pop operation (i.e. extract) is:

   4 pts.

   (a) $O(log(n))$
   (b) $O(n)$
   (c) $O(nlog(n))$
   (d) $O(n^2)$

4. Construct a balanced binary max-heap (i.e. a heap that always returns the maximum element) using the following elements, pushing them onto the heap in the given order:

   3, 4, 2, 1, 5, 6

   Draw the heap after each completed insertion of an element.

   6 pts.

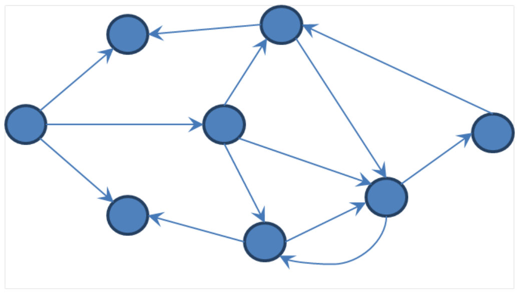
5. Now pop (i.e. extract) the two largest elements off the heap. Draw the heap after each such extraction.

   4 pts.

4 Graphs

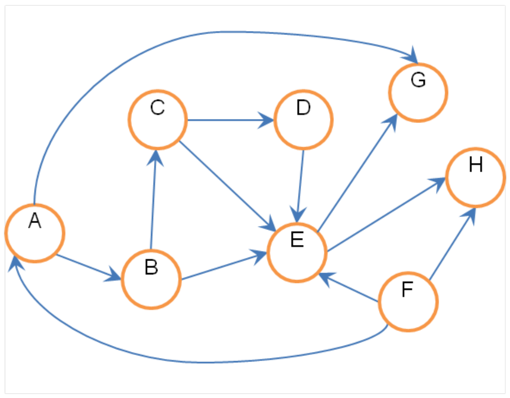
1. Remove the smallest possible number of edges from the following graph so that it becomes a Directed Acyclic Graph (DAG). Indicate the removed edges by crossing them out with an "X".

4 pts.



2. Perform topological sort on the graph below and write down the sorted list of nodes. If there is more than one valid topological sort order, write down all of them.

6 pts.



3. In the graph from the previous question, what is the node with the hightest indegree? What is its indegree?

3 pts.

4. In the same graph, what is the node with the hightest outdegree? What is its outdegree?

3 pts.

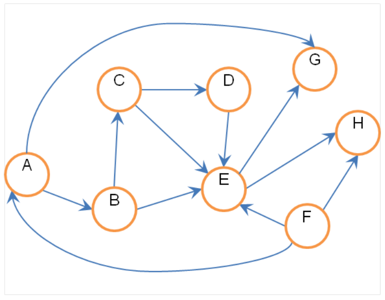
5 Graph Search

1. Give a pseudo-code implementation of a function `bfs_path(G,s,t)` that uses Breadth-First Search (BFS) to return true if an arbitrary unweighted graph G contains a path from s to t, otherwise false. Indicate which datastructures you are using. You can assume that standard datastructures are available and that $s \neq t$.

   6 pts.

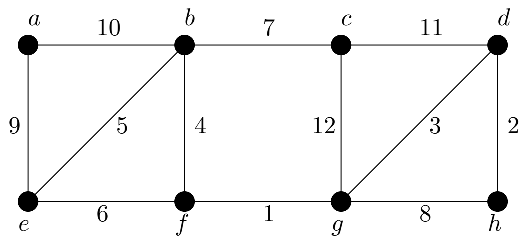
2. In the graph below, use your algorithm from above to compute whether there is a path from node A to node H. In particular, whenever BFS reaches a new node, show the content of the main datastructure that BFS maintains. Break ties between nodes by alphabetical order.

   7 pts.

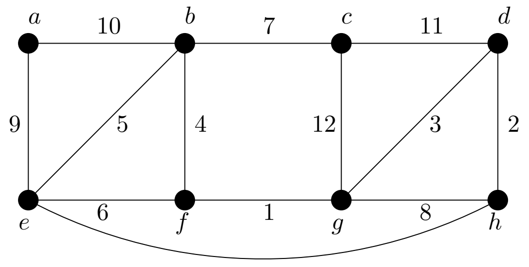


6 Minimum Spanning Trees

1. Which edges belong to the minimum spanning tree (MST) of this graph? 7 pts.



2. In the above graph we add an edge (e, h) with unknown (positive) weight as in the following figure. 3 pts.



Circle the correct answer. Compared to the previous graph, the weight of the MST for this graph

   (a) may become smaller or stay the same
   (b) may become larger or stay the same
   (c) may become either smaller or larger or stay the same

Note: the weight of the MST is the sum of the weights of its edges.

3. Describe in words an algorithm for updating the MST of a graph when a new edge (u, v) is added to the graph (i.e. just like in the previous question). Its time complexity must be better than running Prim's algorithm from scratch. What is the time complexity of your algorithm for a graph with E edges and V vertices? You can assume that all edge weights are distinct, and that your graph and your MST are represented using adjacency lists. 6 pts.

7 Graphical User Interfaces

For all questions in this section, refer to the following program:

```
import javax.swing.*; import java.awt.*; import java.awt.event.*;

public class QuestionFrame extends JFrame {
        JPanel qpnl,apnl,bpnl;
        JLabel q,a;
        JButton btn1,btn2,btn3;

    public QuestionFrame() {
        super("Hello, World!");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(400,400);

        qpnl = new JPanel(new FlowLayout(FlowLayout.CENTER));
        apnl = new JPanel(new FlowLayout(FlowLayout.CENTER));
        bpnl = new JPanel(new FlowLayout(FlowLayout.CENTER));

        q = new JLabel("Who is a better programmer?");
        a = new JLabel(" ");
        qpnl.add(q);
        apnl.add(a);

        btn1 = new JButton("Person on my Left");
        btn2 = new JButton("Me");
        btn3 = new JButton("Person on my Right");
        bpnl.add(btn1);
        bpnl.add(btn2);
        bpnl.add(btn3);

        add(qpnl, BorderLayout.NORTH);
        add(apnl, BorderLayout.CENTER);
        add(bpnl, BorderLayout.SOUTH);

        btn1.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                a.setText("Left");
                pack();
            }
        });
        btn2.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                a.setText("Me");
                pack();
            }
        });
        btn3.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                a.setText("Right");
                pack();
            }
        });

        setVisible(true);
    }

    public static void main(String[] args) {
        new QuestionFrame();
    }
}
```

1. Draw a sketch of what the GUI looks like before any interaction with the user. The sketch should include all features of the GUI and should put them roughly in the correct position.

7 pts.

2. Describe how the window changes after the user has clicked the button "btn3"?

6 pts.