CS/ENGRD2110: Prelim 2
SOLUTION

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

   SOLUTION:
   (a) is the best solution, since it best spreads out the hash keys. (b) takes only as many values as there are characters in the alphabet, so there are many collisions and only the first elements of the array will be used independent of the overall array size. (c) Not bad, but spreads hashkeys over a smaller interval than (a). (d) is not deterministic.
   END SOLUTION

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

   SOLUTION:
   (c) and (d) are false
   END SOLUTION

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.

   SOLUTION:
   (1) h must be deterministic during each execution (ie, for the same key, it has to return the same value each time it is invoked).
   (2) h must return the same value for 2 keys that are considered equal (If A.equals(B), then the hash function must return the same values for A and B).
   END SOLUTION

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).

   SOLUTION:
   true, true
   END SOLUTION

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=

   SOLUTION:

   - push(x): Stack A=[x] ; Stack B=[]
   - push(y): Stack A=[y,x] ; Stack B=[]
   - push(z): Stack A=[z,y,x] ; Stack B=[]
   - poll(): Stack A=[] ; Stack B=[y,z]
   - push(w): Stack A=[w] ; Stack B=[y,z]
   - poll(): Stack A=[w] ; Stack B=[z]
   - push(v): Stack A=[v,w] ; Stack B=[z]
   - poll(): Stack A=[v,w] ; Stack B=[]
   - poll(): Stack A=[] ; Stack B=[v]
   - poll(): Stack A=[] ; Stack B=[]

   The general algorithm is: Always add new elements to stack A, and always extract elements from stack B. If stack B becomes empty, repeatedly pop the next element off A and push on B until A is empty.
   END SOLUTION

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

   SOLUTION:
   (b)
   END SOLUTION

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) $O(n^2)$

   SOLUTION:
   (a)
   END SOLUTION

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) $O(n^2)$

   SOLUTION:
   (a)
   END SOLUTION

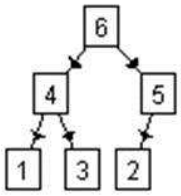
4. Construct a balanced binary max-heap (i.e. a heap that always returns the maximum element) using the following elements, pushing them onto the heap in the given order:

   3, 4, 2, 1, 5, 6

   Draw the heap after each completed insertion of an element.

   6 pts.

   SOLUTION:
   This is the correct final heap:

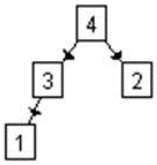


END SOLUTION

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

4 pts.

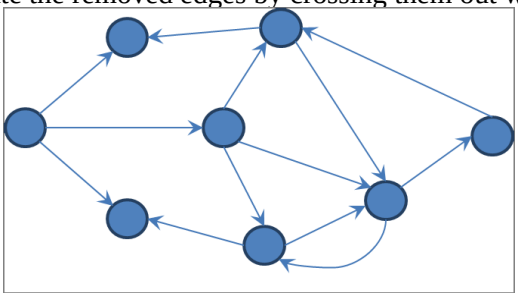
SOLUTION:
This is the correct final heap:



END SOLUTION

4 Graphs

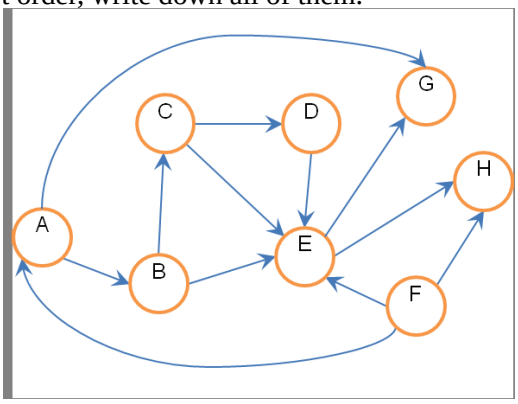
1. Remove the smallest possible number of edges from the following graph so that it becomes a Directed Acyclic Graph (DAG). Indicate the removed edges by crossing them out with an "X".



4 pts.

SOLUTION:
One can get a DAG by crossing out two edges. The one on the top right, and the curved one at the bottom is one solution.
END SOLUTION

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.

SOLUTION:
F,A,B,C,D,E,G,H or F,A,B,C,D,E,H,G
END SOLUTION

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

3 pts.

SOLUTION:
E has indegree 4
END SOLUTION

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

3 pts.

SOLUTION:
F has indegree 3
END SOLUTION

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.

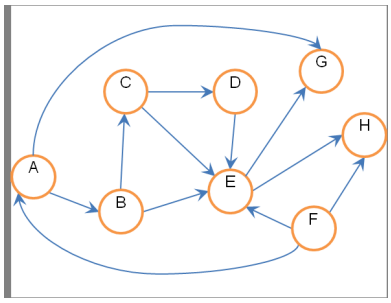
   SOLUTION:

   - Input: start node s, destination node t
   - Put start node s into queue and mark s as visited (use e.g. Hashset).
   - While queue not empty
     - Poll n off queue.
     - FOR all (unmarked) successors n' of n
       - IF n' equals t THEN return TRUE
       - Put n' into queue
       - Mark n' as visited.
   - return FALSE

   END SOLUTION

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.

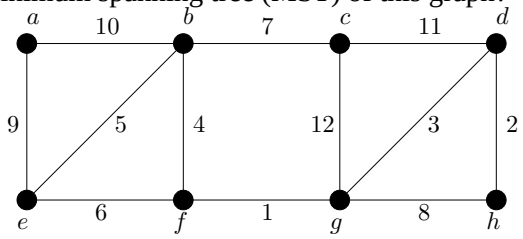
   7 pts.



   SOLUTION:
   The main datastructure is the Queue, but for completeness I also added which nodes are already marked as visited:
   (Queue: A; Visited: A)
   Queue: B,G; Visited: A,B,G
   Queue: G,C,E; Visited: A,B,G,C,E
   Queue: C,E; Visited: A,B,G,C,E
   Queue: E,D; Visited: A,B,G,C,D,E
   (Queue: D,H; Visited: A,B,G,C,D,E,H)
   END SOLUTION

6 Minimum Spanning Trees

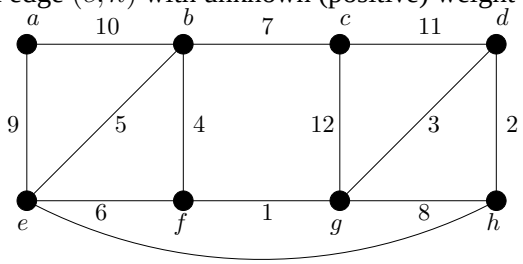
1. Which edges belong to the minimum spanning tree (MST) of this graph?



7 pts.

SOLUTION:
$(f, g), (d, h), (d, g), (b, f), (b, e), (b, c), (a, e)$
END SOLUTION

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.

SOLUTION:
(a)
END SOLUTION

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.

SOLUTION:
Find all the edges that belong to the path from u to v in the MST. This can be done with a simple DFS (or BFS) in time O(V), since an MST has $V - 1$ edges.

Let the edge with maximum weight on the path be (x, y). If the weight of (u, v) is greater than that of (x, y) do not update the tree. Otherwise add (u, v) in the tree and remove (x, y). The running time of the algorithm is dominated by the $O(V)$ time we need for DFS in the MST. The time complexity is $O(V)$ and not $O(E)$, since any MST has only $V - 1$ edges.
END SOLUTION

7 Graphical User Interfaces

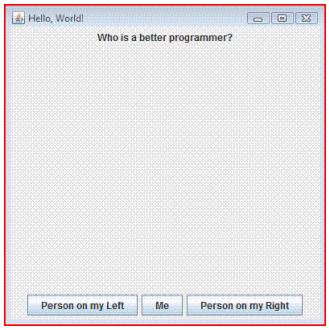
For all questions in this section, refer to the following program:

```
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.

SOLUTION:

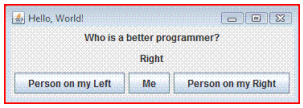


END SOLUTION

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

6 pts.

SOLUTION:



The window gets “packed”, which means the window is changed to a size so that all components fit without extra space. Further, in the center the text “Right” gets displayed.
END SOLUTION