CS 61B Fall 2014 Game Trees, Quad Trees and Heaps

1 Heaps of fun®

(a) Assume that we have a binary min-heap (smallest value on top) data structure called Heap that stores integers and has properly implemented insert and removeMin methods. Draw the heap and its corresponding array representation after each of the operations below:

```
Heap h = new Heap(5); //Creates a min-heap with 5 as the root
h.insert(7);
h.insert(3);
h.insert(1);
h.insert(2);
h.removeMin();
h.removeMin();
```

(b) Consider an array based min-heap with N elements. What is the worst case running time of each of the following operations if we ignore resizing? What is the worst case running time if we take into account resizing? What are the advantages of using an array based heap vs. using a BST-based heap?

```
Insert     _______
Find Min   _______
Remove Min _______
```

(c) Your friend Alyssa P. Hacker challenges you to quickly implement an integer max-heap data structure - "Hah! I'll just use my min-heap implementation as a template to write max-heap.java", you think to yourself. Unfortunately, your arch-nemesis Malicious Mallory deletes your min-heap.java file. You notice that you still have the min-heap.class file; could you use it to complete the challenge? – you can still use methods from min-heap but you cannot modify them. If so, describe your approach, do not write code. If not, explain why it is impossible.

2 Quad Tree

Let's say you were creating a photo album, and you wanted each photo in the album to be geotagged (latitude, longitude coordinates), so that you can quickly access these photos by location. With your 1337 61b skillz, you decide that you're going to use a Quad Tree to build this application. Recall that a node in a Quad Tree has four children, one corrresponding to all keys to the top left, top right, bottom right, and bottom left. There may be multiple photos with the same latitude and longitude. As in HW6, it'll be best to create an inner node class.

(a) Fill out some of the basic instance variables/inner classes you would need in your Quad Tree implementation below. While doing so, think about some design decisions you need to make based on the nature of the application that uses the QuadTree class:

```
/** A Quadtree is a tree data structure that is used to divide 2d
 *  space into 4 quadrants (recursively). When inserting data, it
 *  stores it in the appropriate region.
 */
public class QuadTree<_________________________> {
        _____ root;
        QuadTree(_____ lat,_____ lon, Image img) {
                //Fill me



        }

        //Other Instance variable or inner classes go below this line.






}
```

(b) Now go ahead and write an insert method for your QuadTree class:

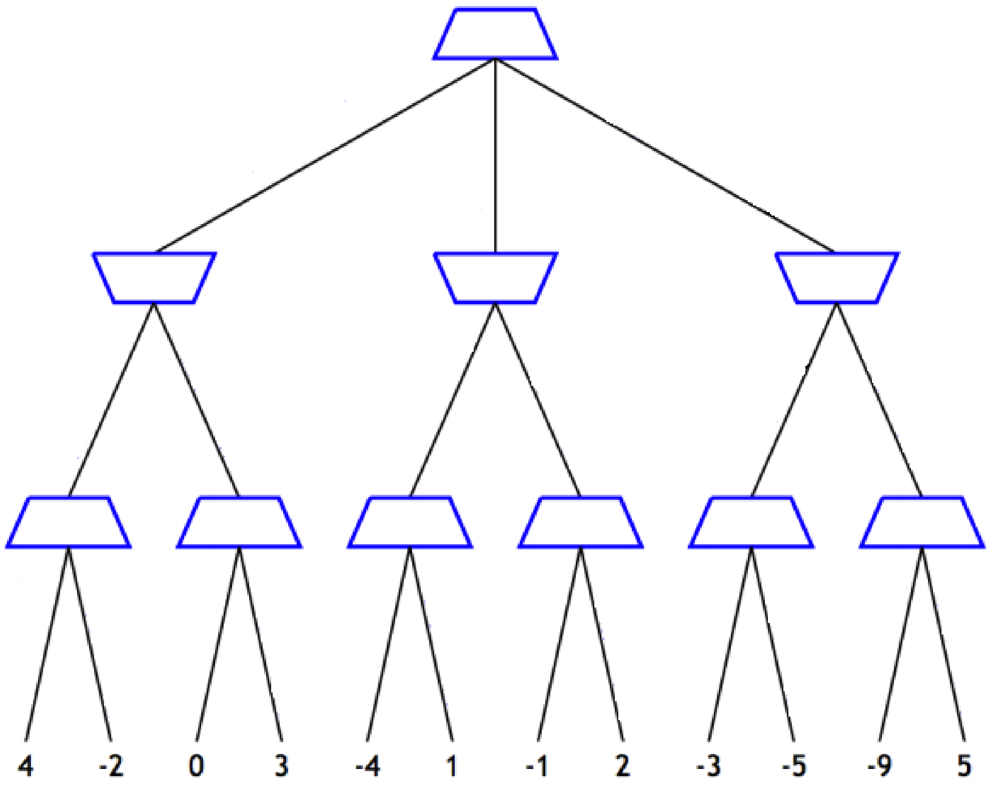
```
public void insert(_____ lat,_____ lon, Image img) {



}
```

3 Minimax

Consider the game tree shown below. Trapezoids that 'point' up, represent the player seeking to maximize the heuristic evaluation (this is you); trapezoids that 'point' down represent the player seeking to minimize the heuristic evaluation (your opponent).



(a) Fill out the values in the 'maximizer' and 'minimizer' node for the above game tree after applying the minimax algorithm to it.

(b) Cross out (with an X) all branches that would be pruned by a Minimax implementation that utilizes the pruning strategy discussed in class (alpha-beta pruning).

(c) According to the minimax algorithm, which move should we make for the above game? Say, your opponent was a 3-year old, would you still use the minimax algorithm?