CS 61B Disc 10: MT2 Review Fall 2014

1 Sometimes Sort of Sorted (from 61BL SU2014 Final)

Let n be some large integer, $k < \log_2 n$. For each of the input arrays described below, explain which sorting algorithm you would use to sort the input array the fastest and why you chose this sorting algorithm.

Make sure to state any assumptions you make about the implementation of your chosen sorting algorithm. Also specify the big-Oh running time for each of your algorithms in terms of k and n. Your big-Oh bounds should be simplified and as tight as possible.

(a) Input: An array of n `Comparable` objects in completely random order

(b) Input: An array of n `Comparable` objects that is sorted except for k randomly located elements that are out of place (that is, the list without these k elements would be completely sorted)

(c) Input: An array of n `Comparable` objects that is sorted except for k randomly located pairs of adjacent elements that have been swapped (each element is part of at most one pair).

(d) Input: An array of n elements where all of the elements are random `int`s between 0 and k

2 Up (from 61BL SU14 MT2)

```
import java.util.*;
public class Tree {
    private TreeNode root;
    public Tree(Object rootItem) {
        this.root = new TreeNode(rootItem);
    }
    private class TreeNode {
        private Object item;
        private ArrayList<TreeNode> children;
        public TreeNode(Object inputObj) {
            this.item = inputObj;
            children = new ArrayList<TreeNode>();
        }
        public void addChild(Object childObj) {
            children.add(new TreeNode(childObj));
        }
    }
}
```

(a) Write a method in the provided `Tree` class that returns the items of the tree in reverse BFS order. That is, it returns an `ArrayList` of items such that for all positive `i`, all items of nodes at depth `i + 1` come before all items of nodes at depth `i` in the returned `ArrayList`. Items of nodes at the same depth level can be in any order in the list. While not necessary, you are allowed to write up to one helper method.

```
public ArrayList<Object> reverseBFS() {









}
```

(b) What is the big-O running time of your algorithm from part (a)? Your answer should be as tight of a bound as possible.

3 HashMap Modification (from 61BL SU2010 MT2)

(a) When you modify a key that has been inserted into a HashMap will you be able to retrieve that entry again? Explain?

☐ Always ☐ Sometimes ☐ Never

(b) When you modify a value that has been inserted into a HashMap will you be able to retrieve that entry again? Explain?

☐ Always ☐ Sometimes ☐ Never

4 isMaxHeap? (based on 61B SP1996 MT3)

The `isMaxHeap` function determines whether a given array represents a valid max heap.

```
public static boolean isMaxHeap(int[] arr) {




}
```

```
assert isMaxHeap(new int[]{264, 38, 79, 2, 37, 77, -1, -2});
assert !isMaxHeap(new int[]{264, 266, 79, 2, 37, 77, -1, -2});
assert !isMaxHeap(new int[]{264, 38, 79, 2, 37, 77, -1, 3});
```