CS 61B Spring 2024

Midterm 2

Thursday, March 21, 2024

Intro: Welcome to CS61B Midterm 2!

Your Name: _____

Your SID: _____ Location: _____

SID of Person to your Left: _____ Right: _____

Formatting:

- ◯ indicates only one circle should be filled in. ☐ indicates more than one box may be filled in. **Please fill in the shape completely.** If you change your response, **erase as completely as possible**.
- Anything you write that you ~~cross out~~ will not be graded.
- You may not use ternary operators, lambdas, streams, or multiple assignment.

Tips:

- This midterm is worth **100 points**, and there are a lot of problems on this exam. Work through the ones with which you are comfortable first. Do not get overly captivated by interesting design issues or complex corner cases you're not sure about.
- Not all information provided in a problem may be useful, and you may not need all lines.
- **We will not give credit for solutions that go over the number of provided lines or that fail to follow any restrictions given in the problem statement.**
- There may be partial credit for incomplete answers. Write as much of the solution as you can, but bear in mind that we may deduct points if your answers are much more complicated than necessary.
- Unless otherwise stated, all given code on this exam compiles. All code has been compiled and executed before printing, but in the unlikely event that we do happen to catch any bugs in the exam, we'll announce a fix.

Write the statement below in the same handwriting you will use on the rest of the exam: "I have neither given nor received help on this exam (or quiz), and have rejected any attempt to cheat. If these answers are not my own work, I may be deducted 9,876,543,210 points on the exam."

_____

_____

_____

_____

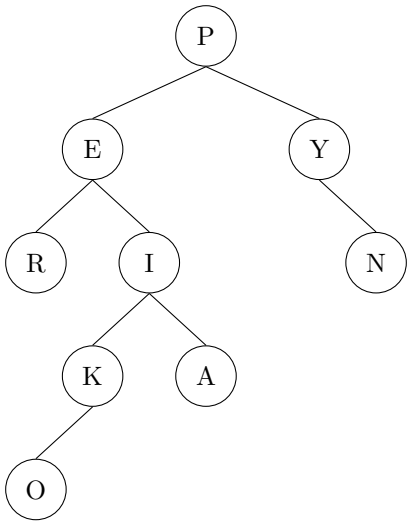
_____

Signature: _____

1 Potpourri

(22 Points)

For the first two subparts, consider the following BST of unique integer elements:



(a) How many elements must be greater than the element at node K?

(b) In what order will a post-order traversal, starting at node P, visit the nodes in the tree above?

- ◯ P, E, Y, R, I, N, K, A, O
- ◯ R, O, K, A, I, E, N, Y, P
- ◯ O, K, A, R, I, E, N, Y, P

(c) True or false: It is possible for a rooted tree with more than 1 node to be simultaneously a valid max-heap and a valid BST.

- ◯ True
- ◯ False

(d) How many positions within a min-heap with $2^N - 1$ elements (where the height of the heap is $N - 1$) could the maximum element potentially occupy? Assume there are no duplicate elements and that the heap is not empty.

- ◯ $\log(N)$
- ◯ N
- ◯ $2^{(N-1)}$
- ◯ 2^N

(e) True or false: Dijkstra's algorithm always works on both directed and undirected graphs with positive edge weights.

- ◯ True
- ◯ False

(f) True or false: A* always runs asymptotically faster than Dijkstra's algorithm.

◯ True

◯ False

(g) True or false: A* may not need to visit all nodes before completing.

◯ True

◯ False

(h) True or false: Kruskal's algorithm always works on graphs with negative edge weights.

◯ True

◯ False

(i) True or false: Prim's algorithm always works on graphs with negative edge weights.

◯ True

◯ False

(j) True or false: In a directed acyclic graph, any post-order traversal that visits all nodes is a valid topological sort.

◯ True

◯ False

(k) In a directed tree, which of the following traversals always results in a topological sort of the tree? Assume the traversal starts at the root of the tree. Select all that apply.

☐ Pre-order

☐ In-order

☐ Post-order

☐ Level-order

◯ None of the above

2 Awesometotics

(22 Points)

For parts (a) through (d), determine the asymptotic behavior, with respect to N, of each scenario. Answer in Θ notation.

Example: A square of side length N is constructed. What is the area of that square?

$\Theta(N^2)$

(a) An `ArrayList` is implemented with additive resizing, then N elements are added to it. What is the maximum number of unused indices in the underlying array?

(b) An `ArrayList` is implemented with multiplicative resizing, then N elements are added to it. What is the maximum number of unused indices in the underlying array?

(c) You create a weighted quick union **without** path compression, containing N elements. You then perform $\lfloor\sqrt{N}\rfloor$ calls to `union`. You look at the underlying array and take the sum of the absolute value of all elements. What is the smallest and largest possible sum?

For example, if the underlying array was $[-4, 2, 0, 0]$, the sum would be $|-4| + |2| + |0| + |0| = 6$.

Note: $\lfloor x \rfloor$ is the largest integer less than or equal to x.

Smallest sum:

Largest sum:

(d) A list of positive integers (not necessarily of length N) is created such that its elements sum to N^2. All elements are inserted, one at a time, into a min-heap. What are the best-case and worst-case runtimes for all insertions to complete?

Best case:

Worst case:

For parts (e) and (f), give the best-case and worst-case runtimes for the functions below.

(e)
```
public static void f1(int N) {
    for (int i = 0; i < N; i += 1) {
        for (int j = i; j < N; j += 1) {
            if (j % 2 == 0) {
                i += 1;
            }
            System.out.println("naming things is hard - Dylan");
        }
    }
}
```

Best case:

○ $\Theta(1)$ ○ $\Theta(\log(\log N))$ ○ $\Theta(\log N)$ ○ $\Theta((\log N)^2)$ ○ $\Theta(N)$ ○ $\Theta(N \log N)$
○ $\Theta(N^2)$ ○ $\Theta(N^2 \log N)$ ○ $\Theta(N^3)$ ○ $\Theta(N^3 \log N)$ ○ $\Theta(N^4)$ ○ $\Theta(N^4 \log N)$
○ Worse than $\Theta(N^4 \log N)$ ○ Never terminates (infinite loop) ○ None of the above

Worst case:

○ $\Theta(1)$ ○ $\Theta(\log(\log N))$ ○ $\Theta(\log N)$ ○ $\Theta((\log N)^2)$ ○ $\Theta(N)$ ○ $\Theta(N \log N)$
○ $\Theta(N^2)$ ○ $\Theta(N^2 \log N)$ ○ $\Theta(N^3)$ ○ $\Theta(N^3 \log N)$ ○ $\Theta(N^4)$ ○ $\Theta(N^4 \log N)$
○ Worse than $\Theta(N^4 \log N)$ ○ Never terminates (infinite loop) ○ None of the above

(f)
```
public static void f2(int N) {
    if (N < 2) {
        return;
    }
    f2(N/2);
    for (int i = 0; i < Math.sqrt(N); i += 1) {
        System.out.println("Aniruth Narayanan aka the Baller");
    }
    f2(N/2);
}
```

Best case:

○ $\Theta(1)$ ○ $\Theta(\log(\log N))$ ○ $\Theta(\log N)$ ○ $\Theta((\log N)^2)$ ○ $\Theta(N)$ ○ $\Theta(N \log N)$
○ $\Theta(N^2)$ ○ $\Theta(N^2 \log N)$ ○ $\Theta(N^3)$ ○ $\Theta(N^3 \log N)$ ○ $\Theta(N^4)$ ○ $\Theta(N^4 \log N)$
○ Worse than $\Theta(N^4 \log N)$ ○ Never terminates (infinite loop) ○ None of the above

Worst case:

○ $\Theta(1)$ ○ $\Theta(\log(\log N))$ ○ $\Theta(\log N)$ ○ $\Theta((\log N)^2)$ ○ $\Theta(N)$ ○ $\Theta(N \log N)$
○ $\Theta(N^2)$ ○ $\Theta(N^2 \log N)$ ○ $\Theta(N^3)$ ○ $\Theta(N^3 \log N)$ ○ $\Theta(N^4)$ ○ $\Theta(N^4 \log N)$
○ Worse than $\Theta(N^4 \log N)$ ○ Never terminates (infinite loop) ○ None of the above

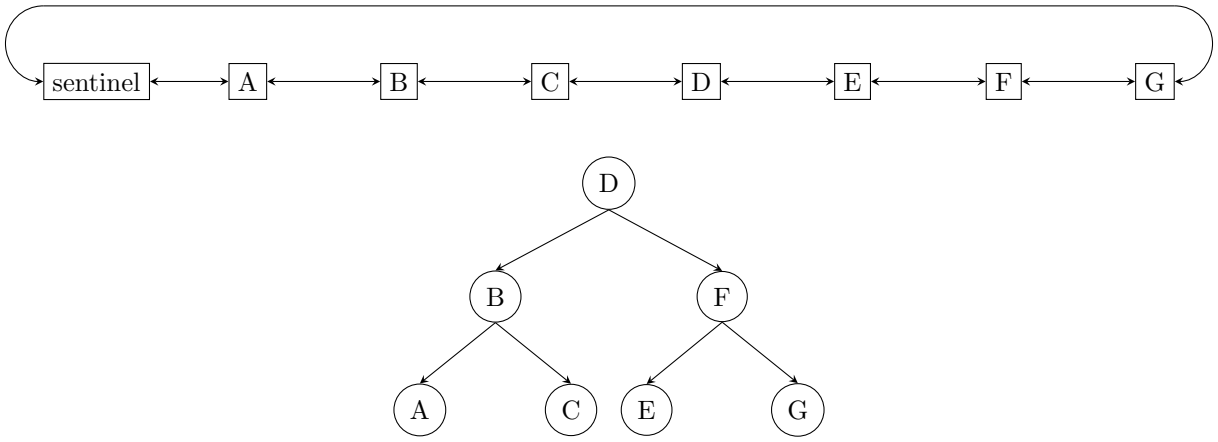
3 It's All a Blur

(18 Points)

Noah realizes that the internal nodes of DLLists and BSTs are remarkably similar: both contain an item, and two pointers to other nodes. Inspired by this, Noah creates a unified Node class that can act as the Node of either a DLList or a BST.

```
class Node {
    int item;
    Node prev;    // The previous node in a DLList, or the left child in a BST
    Node next;    // The next node in a DLList, or the right child in a BST
}
```

Noah hasn't built a wrapper class to handle DLList-specific or BST-specific operations (such as `size`). However, using these Nodes directly, Noah can connect the Nodes to create either a circular doubly-linked list with one sentinel, or a binary search tree, as in the example below.



Complete the method `treeify`, which behaves as follows:

- Input: The sentinel Node of a doubly-linked list.
- Behavior: Modifies the `next` and `prev` pointers of each Node, such that the Nodes are connected in the form of a complete binary search tree.
- Output: The root Node of the resulting tree.

In the example above, `treeify(sentinel)` should convert the top diagram to the bottom diagram, by changing `D.prev` from `C` to `B`, `D.next` from `E` to `F`, `C.next` from `D` to `null`, and so on. `treeify` should then return `D`.

You may assume:

- The input list is not empty (at least one non-sentinel Node exists).
- The number of non-sentinel items in the input list is one less than a power of two.
- Non-sentinel items in the input list are sorted in strictly increasing order.

(a) Complete `treeify` below. You may not create new Nodes. You may not use any methods of the DLList class or BST class.

```
public static Node treeify(Node sentinel) {
    return helper(sentinel.next, sentinel.prev);
}

private static Node helper(Node begin, Node end) {
    if (begin == end) {
        begin.prev = null;
        begin.next = null;
        return begin;
    }
    Node forward = _____1_____;

    Node backward = _____2_____;
    while (forward != backward) {

        _____3_____;

        _____4_____;
    }

    _____5_____;

    _____6_____;

    return _____7_____;
}
```

(b) What is the runtime of `treeify`, if run on a list of `N` nodes, where `N` is one less than a power of 2? Express your answer as a Θ bound.

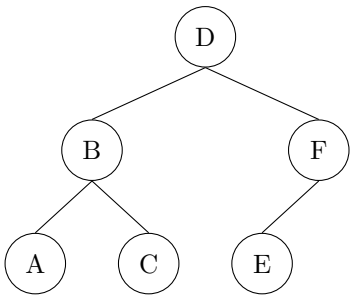
4 Lost in LLRBs

(18 Points)

After finishing your Deque from Midterm 1, you start working on an LLRB implementation. However, the same hacker comes along and paints all your links black! You may assume that the LLRB was valid before it got hacked.

Recall from lab that an LLRB node is red if the link to its parent is red, and black if either the node is the root, or the link to its parent is black. For example, if the edge E-F is meant to be red, then E is colored red.

(a) Consider the following hacked LLRB (all links painted black):



Select all nodes that must be red for this LLRB to be valid.

☐ A ☐ C ☐ F

☐ B ☐ E ◯ None of the above

(b) Write `fixLLRB`, which restores the correct colors to all nodes without otherwise modifying the tree.

Your code must be asymptotically optimal. You may not use any LLRB methods/attributes not listed below.

```
public class LLRB {
    private class Node {
        int value;
        Node left;
        Node right;
        boolean isRed;
    }

    private Node root;

    public void fixLLRB() {
        helper(root);
    }

    private _____ helper(Node n) {
              1

        if (_____________________________________________________________) {
                                        2

            return _____________________________________________________;
                                        3
        }
        int leftHeight = ____________________________________________________;
                                        4

        int rightHeight = ___________________________________________________;
                                        5

        if (_____________________________________________________________) {
                                        6
            n.left.isRed = true;
        }

        return _____________________________________________________;
                                        7
    }
}
```

5 I Love Hashbrowns

(12 Points)

In this question, consider these two `hashCodes`:

- The **default** `hashCode`: The implementation of `hashCode` provided by the `Object` class.
- The **trivial** `hashCode`: The `hashCode` that always returns 0.

Also, consider these two `equals` methods:

- The **default** `equals` method: The implementation of `equals` provided by the `Object` class.
- The **trivial** `equals` method: The `equals` method that always returns `true`.

Select whether each statement is true or false.

(a) The **default** `hashCode` is a valid `hashCode` for the **trivial** `equals` method.

◯ True

◯ False

(b) The **trivial** `hashCode` is a valid `hashCode` for the **trivial** `equals` method.

◯ True

◯ False

(c) The **trivial** `hashCode` is a valid `hashCode` for **all** `equals` methods.

◯ True

◯ False

(d) The **default** `hashCode` is a valid `hashCode` for **all** `equals` methods.

◯ True

◯ False

(e) The **only** valid `hashCode` (over all possible `hashCodes`) for the **default** `equals` method is the **default** `hashCode`.

◯ True

◯ False

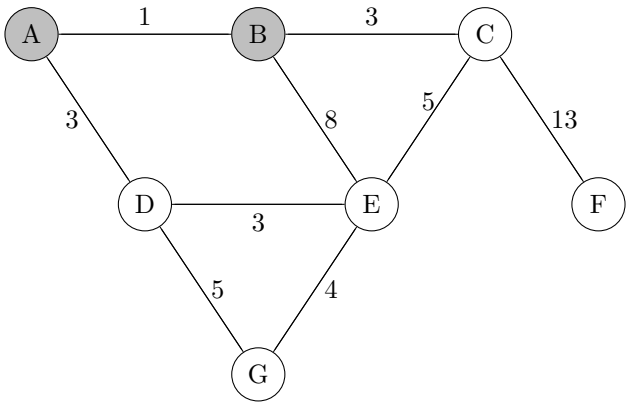
(f) The **only** valid `hashCode` (over all possible `hashCodes`) for the **trivial** `equals` method is the **trivial** `hashCode`.

◯ True

◯ False

6 Where's Prim?

(8 Points)



(a) Suppose we run Dijkstra's algorithm on the graph above, with source node A. After visiting A and B, the `distTo` array is as follows:

Node	A	B	C	D	E	F	G
`distTo`	0	1	4	3	9	∞	∞

Fill in the `distTo` array after visiting one more node and relaxing all of its outgoing edges. In each box, write either a number or ∞.

Node	A	B	C	D	E	F	G
`distTo`							

(b) What is the sum of the edge weights in the minimum spanning tree of the graph above?

(c) Which edge(s) are guaranteed to be in all (not necessarily minimum) spanning trees? Select all that apply.

- ☐ AB
- ☐ AD
- ☐ BC
- ☐ BE
- ☐ CE
- ☐ CF
- ☐ DE
- ☐ DG
- ☐ EG
- ◯ None of the above

(d) You add a new edge between F and G with weight x, and notice that the MST of the modified graph has a lower total weight (compared to the original MST). What is the maximum integer value that x could be?

Nothing on this page is worth any points.

---

7 The Dancing Men

(0 Points)

The Dancing Men Cipher was initially conceived by Sir Arthur Conan Doyle in the Sherlock Holmes story "The Adventure of the Dancing Men". The original alphabet was incomplete and only contained 18 letters, but today there is a consensus on all 26 English letters and all 10 digits.

What does the following ciphertext decode to?

8 Feedback

(0 Points)

Leave any feedback, comments, concerns, or more drawings below!