CSM 61B Fall 2020

Final Review

Mentoring 15: December 1, 2020

1 LLRB

1.1 Those darned Stanford Students are at it again! In an attempt to reverse-engineer your code, they stole your `get` and `put` methods! Fortunately, you understand red-black trees pretty well. Consider your now incomplete `RBTree` implementation.

```
public class RBTree<Key extends Comparable<Key>, Value> {
    private static final boolean RED   = true;
    private static final boolean BLACK = false;
    private Node root;     // root of the BST

    private class Node {
        private Key key;           // key
        private Value val;         // associated data
        private Node left, right;  // links to left and right subtrees
        private boolean color;     // color of parent link
        private int size;          // subtree count

        public Node(Key key, Value val, boolean color, int size) {
            this.key = key;
            this.val = val;
            this.color = color;
        }
    }

    public RBTree() {...}
    private Node rotateRight(Node h) {...}
    private Node rotateLeft(Node h){...}
    private void flipColors(Node h) {...}
    private boolean isRed(Node h) {...}  // returns false if node is black or null
    private int size(Node h) {...}

    public void put(Key key, Value val) {
        root = put(root, key, val);
        root.color = BLACK;
    }
```

```
private Value get(Key key) {
    Node runner = ________;
    while (____________) {
        int cmp = _______________;
        if (_______) {
            runner = __________;
        } else if (________) {
            runner = __________;
        } else {
            _________________;
        }
    }
    return null;
}
private Node put(Node h, Key key, Value val) {
    if (h == _____){
        return _____________;
    }
    int cmp = _____________;
    if  (_________) {
        h.left  = _____________;
    } else if (__________) {
        h.right = _____________;
    } else {
         h.val = _____;
    }
    if (_____________ && _____________)) {
         h = rotateLeft(h);
    }
    if (_____________  &&  _____________) {
        h = rotateRight(h);
    }
    if (_____________ && _____________) {
        _____________;
    }
    h.size = _________________;

    return h;
}
```

2 Sorting

2.1 We have a list of points on the plane. Find the K closest points to the origin (0, 0). (Here, the distance between two points on a plane is the Euclidean distance.)

You may return the answer in any order. The answer is guaranteed to be unique (except for the order that it is in.)

Example
Input: points = [[3,3],[5,-1],[-2,4]], K = 2
Output: [[3,3],[-2,4]]
(The answer [[-2,4],[3,3]] would also be accepted.)

This problem is taken from Leetcode: https://leetcode.com/problems/k-closest-points-to-origin/

3 Tries

3.1 Given a dictionary of n prefix strings and a sentence, describe a procedure to replace each word in the sentence with the shortest prefix string that can be found in the dictionary. If no such prefix string exists in the dictionary, leave the word as is.

For example, say we have the following prefix strings in our dictionary: `"bab"`, `"bat"`, `"rac"`, and `"racco"` and the sentence, `"the baboons battled the raccoon"`. Our output should be `"the bab bat the rac"`

4 Graphs

4.1 Briefly describe an efficient algorithm and the runtime for finding a minimum spanning tree in an undirected, connected graph $G = (V, E)$ when the edge weights satisfy:

(a) For all $e \in E$, $w_e = 1$. (All edge weights are 1.)

(b) For all $e \in E$, $w_e \in \{1, 2\}$. (All edge weights are either 1 or 2.)

5 Weighted QuickUnion

5.1 Describe how to construct a `WeightedQuickUnionUF` tree of maximum height.