Elle Simonds

Dr. Parra Rodriguez

Intro to Algor.

Apr 16, 2025

Assignment 6

```
1.    A.  7.9
             /    \
           0.5      8.2
             \        \
             1.0         9.9
                \
               6.5
               /  \
             1.2    7.0
               \
               2.4
                 \
                5.6
                /
              3.6
      Height = 8
      B. Petit Four
              /
          Cupcake
              \
             Donut
```

```
         \
        Eclair
            \
           Froyo
              \
          Gingerbread
                \
             Honeycomb
```

Height = 6

```
C.     32
      /   \
     5      94
       \   /
       10 87
          /
        85
          \
          47
         /
       25
         \
         29
```

Height = 5

```
D.       34
        /    \
      30        75
      /        \
    13            77
   /  \            \
 10    20            96
 / \    /
5 11 19
           \
            48
          /
        39
            \
            50
             \
             93
```

Height = 8

2.

```
a.          510
            /    \
         111         372
       /    \     /    \
    15       108     80      379
          /   \        /   \
        46     129       87      283
       /  \              /
```

```
36   37          93
                 \
                 97
```

b. No, because the updated node values no longer maintain the BST property, which is left < root < right.

c. No, because the balance factor property of AVL trees is not maintained after updating the nodes.

5. 1. initializeCandidates(LinkedList<String> candidates)

- Time: O(n) – iterates once through n candidates to add to the HashMap and ArrayList.
- Space: O(n) – stores each candidate in both the voteCount map and candidateList.

2. castVote(String candidate)

- Time: O(1) – direct access and update in HashMap.
- Space: O(1) – no additional space used.

3. castRandomVote()

- Time: O(1) – random access in a list and then castVote (which is O(1)).
- Space: O(1) – no extra space used.

4. rigElection(String candidate)

- Time: O(n) – resets all vote counts (n candidates) and sets one to maxVotes.
- Space: O(1) – no new structures; reuses existing.

5. getTopKCandidates(int k)

- Time: O(n log n) – building the priority queue from n candidates takes O(n), polling top k takes O(k log n).
- Space: O(n) – stores all entries in a priority queue.

6. auditElection()

- Time: O(n log n) – similar to getTopKCandidates, must sort or heapify all n candidates.
- Space: O(n) – creates a maxHeap copy of the entries.