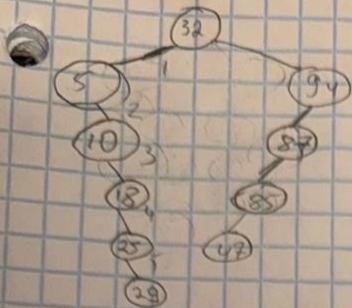


1. (Text) Type of tree

③ [32, 5, 94, 87, 10, 18, 85, 47, 25, 29]



① 32 → root

5 < 32 → left of the root

94 > 32 → right of the root

87 > 32; 87 < 94 → left of 94

10 < 32; 10 > 5 → right of 5

18 < 32; 18 > 5, 18 > 10 → right of 10

85 > 32; 85 < 94; 85 < 87 → left of 87

47 > 32; 47 < 94; 47 < 87; 47 < 85 → left of 85

25 < 32; 25 > 5; 25 > 10; 25 > 18; →

29 < 32; 29 > 5; 29 > 10; insert at the right of 18.

29 > 18; 29 > 25 → insert at the right of 25

the longest path =

= height =

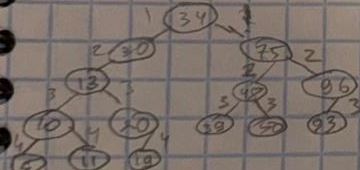
1 → [32 → 5 → 10 → 18 → 25 → 29]

= 5 → (the longest)

2 → [32 → 94 → 87 → 85 → 47] = 4.

Height = 5

④ [34, 30, 75, 77, 96, 48, 39, 50, 93, 13, 10, 5, 11, 20, 19]



① 34 → root

30 < 34 → insert at the left of 34

75 > 34 → insert at the right of 34

96 > 34; 96 > 75 → insert at the right of 75

48 > 34; 48 < 75 → insert at the left of 75

39 > 34; 39 < 75; 39 < 48 → insert at the left of 48

50 > 34; 50 < 75; 50 < 48 → insert at the right of 48

93 > 34; 93 > 75; 93 < 96 → insert at the left of 96

13 < 34; 13 < 30 → insert at the left of 30

10 < 34; 10 < 30; 10 < 13 → insert at the left of 13

5 < 34; 5 < 30; 5 < 13; 5 < 10 → insert at the left of 10

11 < 34; 11 < 30; 11 < 13; 11 > 10 → insert at the right of 10

20 < 34; 20 < 30; 20 > 13; insert at the right of 13

19 < 34; 19 < 30; 19 > 13; 19 < 20 → insert at the left of 20.

Height = the longest path

① [34 → 30 → 13 → 10 → 5] = 4

② [34 → 30 → 13 → 10 → 11] = 4

③ [34 → 30 → 13 → 20 → 19] = 4

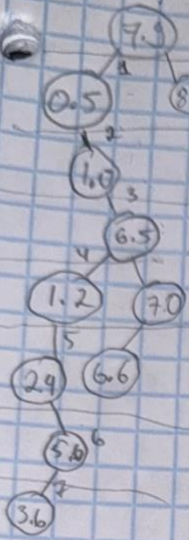
④ [34 → 75 → 48 → 39] = 3

⑤ [34 → 75 → 48 → 50] = 3

⑥ [34 → 75 → 96 → 93] = 3

Height = 4

1) [7.9, 0.5, 1.0, 6.5, 8.2, 7.0, 6.6, 9.9, 1.2, 2.4, 5.6, 3.6]



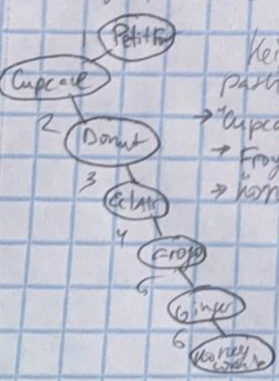
- 1) 7.9 - root
- 2) 0.5 < root \rightarrow left
- 3) 1.0 < root, 1.0 > 0.5 \rightarrow right of (0.5)
- 4) 6.5 < 7.9, 6.5 > 0.5, 6.5 > 1.0 \rightarrow right of (1.0)
- 5) 8.2 > root \rightarrow right of the root
- 6) 7.0 < root; 7.0 > 1.0; 7.0 > 6.5 \rightarrow right of (6.5)
- 7) 6.6 < root; 6.6 > 0.5; 6.6 > 1.0; 6.6 > 6.5; 6.6 < 7.0 \rightarrow left of (7.0)
- 8) 9.9 > root; 9.9 > 8.2 \rightarrow right of (8.2)
- 9) 1.2 < root; 1.2 > 0.5; 1.2 > 0.5; 1.2 < 6.5 \rightarrow insert at the left of (6.5)
- 10) 2.4 < root; 2.4 > 0.5; 2.4 > 1.0; 2.4 < 6.5, 2.4 > 1.2 \rightarrow insert at the right of (1.2)
- 11) 5.6 < root; 5.6 > 0.5, 5.6 > 1.0, 5.6 < 6.5, 5.6 > 1.2, 5.6 > 2.4 \rightarrow insert at the right of 2.4
- 12) 3.6 < root; 3.6 > 0.5, 3.6 > 1.0, 3.6 < 6.5, 3.6 > 1.2, 3.6 > 2.4, 3.6 < 5.6 \rightarrow insert at the left of (5.6)

The longest path \rightarrow height = [7.9 \rightarrow 0.5 \rightarrow 1.0 \rightarrow 6.5 \rightarrow 1.2 \rightarrow 2.9 \rightarrow 5.6 \rightarrow 3.6] = 7

Height = 7

height = [7.9 \rightarrow 8.2 \rightarrow 9.9] = 2 (not the longest)

2) Lexicograph order by: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
 ["Petit Four", "Cupcake", "Donut", "Eclair", "Froyo", "Gingerbread", "Honeycomb"]



Height = the longest path = ["Petit Four" \rightarrow "Cupcake" \rightarrow "Donut" \rightarrow "Eclair" \rightarrow "Froyo" \rightarrow "Gingerbread" \rightarrow "Honeycomb"] = 6

1. "Petit Four" - root
2. "Cupcake" < "Petit Four" \rightarrow left of the root
3. "Donut" < "Petit Four", "Donut" > "Cupcake" \rightarrow right of "Cupcake"
4. "Eclair" < "Petit Four", "Eclair" > "Cupcake", "Eclair" > "Donut" \rightarrow right of "Donut"
5. "Froyo" < "Petit Four", "Froyo" > "Cupcake", "Froyo" > "Donut", "Froyo" > "Eclair" \rightarrow insert at the right of "Eclair"
6. "Gingerbread" < "Petit Four", "Gingerbread" > "Cupcake", "Gingerbread" > "Donut", "Gingerbread" > "Eclair", "Gingerbread" > "Froyo" \rightarrow insert at the right of "Froyo"
7. "Honeycomb" < "Petit Four", "Honeycomb" > "Cupcake", "Honeycomb" > "Donut", "Honeycomb" > "Eclair", "Honeycomb" > "Froyo", "Honeycomb" > "Gingerbread" \rightarrow insert at the right of "Gingerbread"

Height = 6

2. (text) BST Traversal

② Preorder: visit \rightarrow left \rightarrow right
Inorder: left \rightarrow visit \rightarrow right

Preorder: [273, 144, 15, 218, 82, 110, 37, 189, 65, 366, 254, 87, 190, 93]

with root 68: $68 + 21 + 92 \cdot 2 = 273$

⑥8: left: $21 + 15 + 54 \cdot 2 = 144$

①5 = $15 + 0 + 0 \cdot 2 = 15$

⑤4 = $54 + 46 + 59 \cdot 2 = 218$

④6 = $46 + 36 + 0 \cdot 2 = 82$

③6 = $36 + 0 + 37 \cdot 2 = 110$

③7 = $37 + 0 + 0 \cdot 2 = 37$

⑤9 = $59 + 0 + 65 \cdot 2 = 189$

⑥5 = $65 + 0 + 0 \cdot 2 = 65$

⑤2 = $92 + 80 + 97 \cdot 2 = 366$

⑧0 = $80 + 0 + 87 \cdot 2 = 254$

⑧7 = $87 + 0 + 0 \cdot 2 = 87$

⑨2 = $92 + 80 + 97 \cdot 2 = 366$

⑨3 = $93 + 0 + 0 \cdot 2 = 93$

⑨7 = $97 + 0 + 0 \cdot 2 = 97$

⑨0 = $90 + 0 + 0 \cdot 2 = 90$

⑨7 = $97 + 0 + 0 \cdot 2 = 97$

⑨0 = $90 + 0 + 0 \cdot 2 = 90$

⑨7 = $97 + 0 + 0 \cdot 2 = 97$

⑨0 = $90 + 0 + 0 \cdot 2 = 90$

⑨7 = $97 + 0 + 0 \cdot 2 = 97$

⑨0 = $90 + 0 + 0 \cdot 2 = 90$

⑨7 = $97 + 0 + 0 \cdot 2 = 97$

⑨0 = $90 + 0 + 0 \cdot 2 = 90$

⑨7 = $97 + 0 + 0 \cdot 2 = 97$

⑨0 = $90 + 0 + 0 \cdot 2 = 90$

⑨7 = $97 + 0 + 0 \cdot 2 = 97$

⑨0 = $90 + 0 + 0 \cdot 2 = 90$

⑨7 = $97 + 0 + 0 \cdot 2 = 97$

⑨0 = $90 + 0 + 0 \cdot 2 = 90$

⑨7 = $97 + 0 + 0 \cdot 2 = 97$

⑨0 = $90 + 0 + 0 \cdot 2 = 90$

⑨7 = $97 + 0 + 0 \cdot 2 = 97$

⑨0 = $90 + 0 + 0 \cdot 2 = 90$

⑨7 = $97 + 0 + 0 \cdot 2 = 97$

⑨0 = $90 + 0 + 0 \cdot 2 = 90$

⑨7 = $97 + 0 + 0 \cdot 2 = 97$

⑨0 = $90 + 0 + 0 \cdot 2 = 90$

⑨7 = $97 + 0 + 0 \cdot 2 = 97$

⑨0 = $90 + 0 + 0 \cdot 2 = 90$

⑨7 = $97 + 0 + 0 \cdot 2 = 97$

⑨0 = $90 + 0 + 0 \cdot 2 = 90$

⑨7 = $97 + 0 + 0 \cdot 2 = 97$

⑨0 = $90 + 0 + 0 \cdot 2 = 90$

⑨7 = $97 + 0 + 0 \cdot 2 = 97$

⑨0 = $90 + 0 + 0 \cdot 2 = 90$

⑨7 = $97 + 0 + 0 \cdot 2 = 97$

⑨0 = $90 + 0 + 0 \cdot 2 = 90$

⑨7 = $97 + 0 + 0 \cdot 2 = 97$

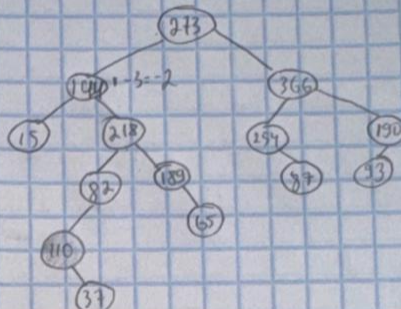
⑨0 = $90 + 0 + 0 \cdot 2 = 90$

⑨7 = $97 + 0 + 0 \cdot 2 = 97$

⑨0 = $90 + 0 + 0 \cdot 2 = 90$

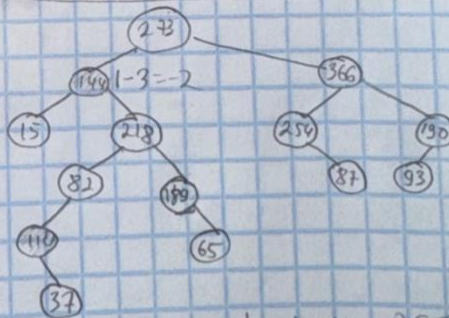
⑨7 = $97 + 0 + 0 \cdot 2 = 97$

⑨0 = $90 + 0 + 0 \cdot 2 = 90$



① With Preorder: node.data = left.data + right.data
 The BST property of left < node < right is broken at node 110, since its value is bigger than 82. No

② No, since at node 144 $H_L - H_R = -2$, which breaks the AVL property of balance factor being ± 1 or 0 .



③ No, since 110 breaks the BST property, b/c it's ~~more~~ more than 82.

④ No, since at node 144, -2 is the BF, BF of AVL can't be -2 .

5. Time complexity analysis for each method:
Time and Space Complexity Analysis (Worst Case)
 m = Number of candidates
 p = Total number of electorate votes

Election Class:

1.initializeCandidates (LinkedList<String> candidates)

Time: $O(m)$

Initializes a map with m candidates (loops through all candidates once).

Space: $O(m)$

Stores m candidates in the votes map and candidates list.

2. castVote(String candidate)

Time: $O(1)$ (average case) / $O(m)$ (worst case for hash collisions)

Checks and updates the candidate's vote count in the hash map.

Space: $O(1)$

3. castRandomVote ()

Time: $O(1)$

Randomly selects a candidate and casts a vote.

Space: $O(1)$

4.rigElection (String candidate)

Time: $O(m)$

Iterates over all m candidates twice:

Calculates the total votes of other candidates.

Redistributes votes to rig the election.

Space: $O(1)$

5.getTopKCandidates (int k)

Time: $O(m + k \log m)$

Heap construction: $O(m)$ (adding m candidates to a max-heap).

Extracting top k : $O(k \log m)$ (each extraction from the heap takes $O(\log m)$).

Space: $O(m)$ (stores all m candidates in the heap).

6.auditElection()

Time: $O(m \log m)$

Sorts all m candidates by votes (descending order).

Space: $O(m)$ (stores the sorted list of m candidates).

7.setElectorateVotes (int p)

Time: $O(1)$

Space: $O(1)$

ElectionSystem Class (Main Method)

1.Shuffling Candidates

Time: $O(m)$

Randomizes the order of m candidates using `Collections. shuffle`.

Space: $O(m)$ (stores the shuffled list).

2.Casting Votes

Time: $O(p)$

Casts p votes (each `castRandomVote` is $O(1)$).

Space: $O(1)$

3.Rigging the Election

Time: $O(m)$

Same as `rigElection` in the Election class.

Space: $O(1)$

4.Top-K Queries & Audit

Time: $O(m \log m)$

Dominated by sorting in `auditElection`.

Space: $O(m)$ (stores sorted results).

Overall Time : $O(m + p + m \log m)$

Overall Space : $O(m)$