

10. ALGORITHM computeRange (root)

// Input: root node

// Output: range (difference between the largest & smallest numbers in the tree)

node ← root

while (node.left ≠ null) do

node ← node.left

end while

smallest ← node.value

node ← root

while (node.right ≠ null) do

node ← node.right

end while

largest ← node.value

range ← largest - smallest

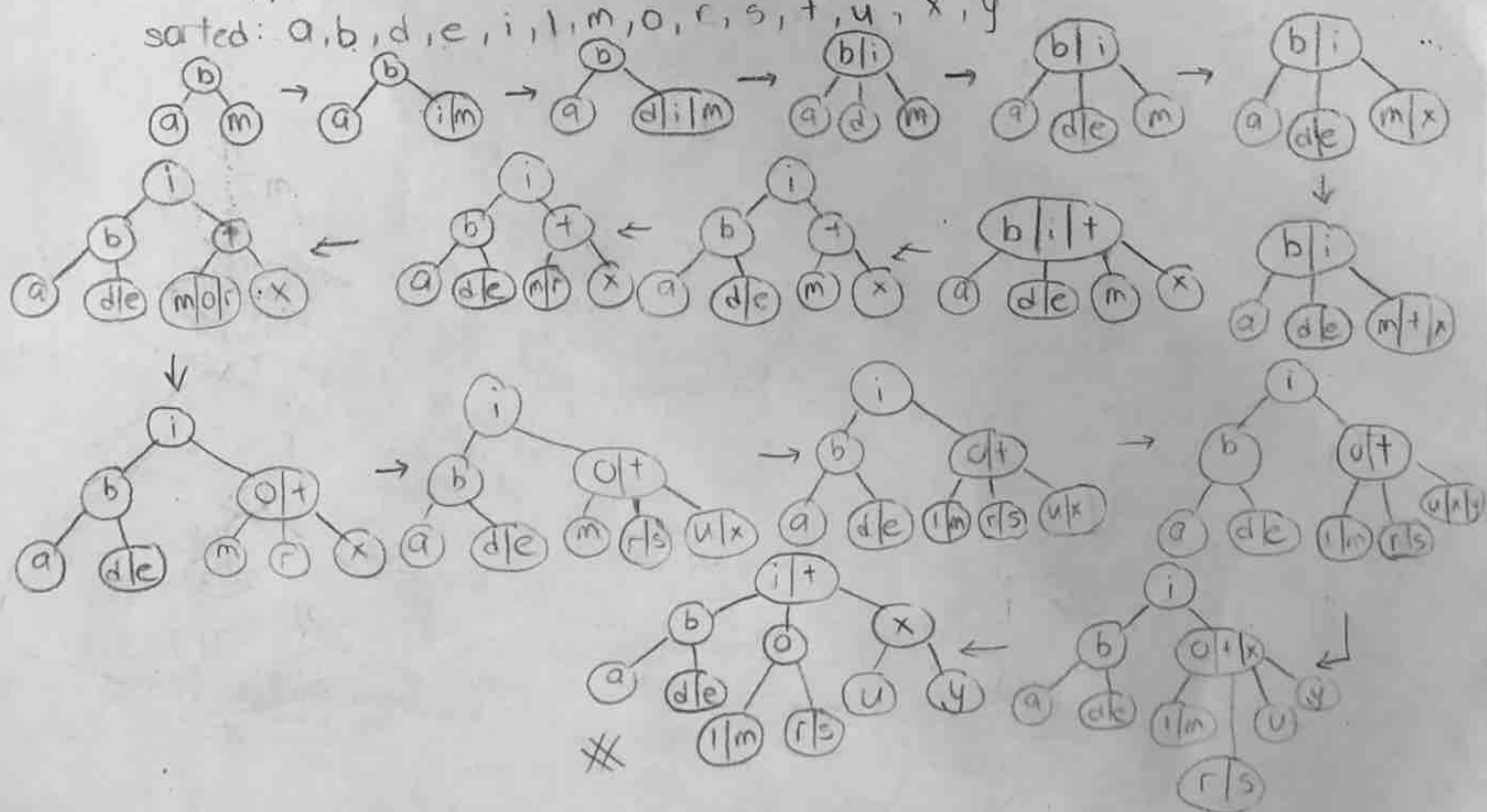
return range

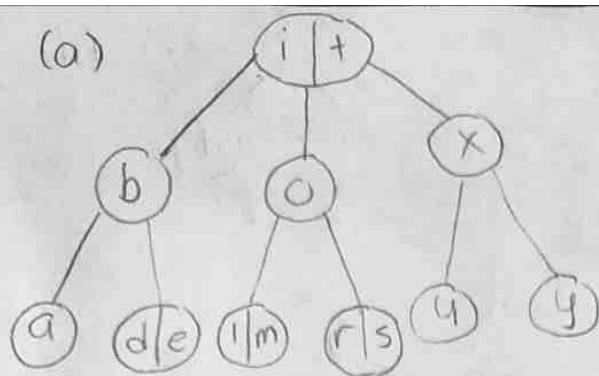
Worst case: $O(\log n) + O(\log n) \approx O(\log n)$

- b. The smallest key is always located at the leftmost node, while the largest key is always located at the rightmost node.

2a. list: a, m, b, i, d, e, x, t, r, o, u, s, l, y

sorted: a, b, d, e, i, l, m, o, r, s, t, u, x, y





(b) - There are 14 keys in the AVL tree.

- The largest number of key comparisons in a successful search will be in the searches for e, m, and s will be equal to 5.

- Average number of comparisons:

$$\begin{aligned}
 & \frac{1}{14} C(i) + \frac{1}{14} C(+) + \frac{1}{14} C(b) + \frac{1}{14} C(o) + \frac{1}{14} C(x) + \frac{1}{14} C(a) + \frac{1}{14} C(d) + \frac{1}{14} C(e) \\
 & + \frac{1}{14} C(l) + \frac{1}{14} C(m) + \frac{1}{14} C(r) + \frac{1}{14} C(s) + \frac{1}{14} C(u) + \frac{1}{14} C(y) \\
 & = (1 + 2 + 3 + 3 + 3 + 4 + 4 + 5 + 4 + 5 + 4 + 5 + 4 + 4) \frac{1}{14} \\
 & = \frac{51}{14}
 \end{aligned}$$

3 ALGORITHM computeRange (root)

// Input : root node

// Output : range (difference between the largest and smallest numbers in the tree)

node \leftarrow root

while (node.left \neq null) do

node \leftarrow node.left

end while

smallest \leftarrow node.value

node \leftarrow root

while (node.right \neq null) do

node \leftarrow node.right

end while

largest \leftarrow node.value

range \leftarrow largest - smallest

return range

Worst case efficiency: $O(\log n) + O(\log n) \approx O(\log n)$

4.

	Job 1	Job 2	Job 3	Job 4
Person 1	9	2	7	8
Person 2	6	4	3	7
Person 3	5	8	1	8
Person 4	7	6	9	4

- n people needed to be assigned to execute n jobs
- each person is assigned to exactly one job

ALGORITHM findMinCost (A[0..n-1, 0..n-1])

// Input: Matrix of job / person for cost

// Output: Minimum cost

minCost \leftarrow 0, k \leftarrow 0

selectedJob [0..n-1]

for i \leftarrow 0 to n-1 do

min \leftarrow ∞

for j \leftarrow 0 to n-1 do

if findValue(selectedJob[j], j) = true then
continue

end if

if A[i, j] < min

min \leftarrow A[i, j]

k \leftarrow j

end if

minCost \leftarrow minCost + min

insert(selectedJob[j], k)

return minCost