CS 225: Data Structures

Assignment 7

Group D1

Last Modified on April 13, 2022

Members	
Name	**Student ID**
Li Rong	3200110523
Zhong Tiantian	3200110643
Zhou Ruidi	3200111303
Jiang Wenhan	3200111016

Please turn over for our answers.

Exercise 4 is written by LI Rong.

Ex. 1 **Our Answer.** (This part is written by ZHONG Tiantian)

(i) Since in an AVL tree the left child is always smaller than its parent, and the right bigger, we can perform a DFS to do this job. Our algorithm is described in Function MINIMUM and MAXIMUM.

Function MINIMUM(node)

```
if leftChild = Null then
    return node
end
return MINIMUM(leftChild)
```

Function MAXIMUM(node)

```
if rightChild = Null then
    return node
end
return MAXIMUM(rightChild)
```

(ii) Define a function CheckAVL, which computes the depth of the tree with root `node`. See Function CheckAVL.

Function CheckAVL(node,depthOfParent,maxDepth)

Ex. 2 **Our Answer.** (This part is written by JIANG Wenhan, ZHONG Tiantian)

(i) Applying a search sequence from root to one leaf node (fixed directoin), then we return to the root and begin another search from root to another leaf node. Repeat such operation until all leaf nodes are traversed.

A structured description is provided in Algorithm 1.

Algorithm 1: Depth-Frist Search

```
ListOfElements ← [] /* Initialize an empty list */
Procedure DepthFirstSearch(node)
    if node is a leaf child then
        append key to ListOfElements
        exit subroutine
    end
    if node has a left child then
        append key to ListOfElements
        DepthFirstSearch(leftChild)
    end
    if node has a right child then
        append key to ListOfElements
        DepthFirstSearch(leftChild)
    end
/* Main */
begin
    call DepthFirstSearch(root)
    return ListOfElements
end
```

(ii) Applying a search sequence firstly among all child nodes of root node and add the children to the queue. The queue records the nodes to visit in the following steps.

Retrieve the top element (denoting it with e) in the queue, and **append** all the child nodes of e to the queue. Popping e, and repeat the operation until the queue is empty.

A structured description is provided in Algorithm 2.

Algorithm 2: Breath-First Search

```
append root to visit_queue
while visit_queue ≠ ∅ do
    node ← the front element in visit_queue
    pop visit_queue's front element
    append node's children (if any) to visit_queue
end
return result_list
```

(iii) Create a queue recording the element in AVL tree in an ascending sequence. We can achieve such recording through following method: ————————.Then we seek for the median value in the sequence and return it.

Algorithm 3: Find Median

(iv) hello world

hello

Ex. 3 **Our Answer.** (This part is written by ZHOU Ruidi)

(i) We consider the case for a single layer and the depth of a tree.

A Single Layer. For each level, we use binary search with $(b-1)$ nodes (since a root/vertex has at most b children); thus the comparison times for a single level is

$$\log(b-1) < \log b$$

Depth of the Tree. Because the minimum number of child nodes of a vertex is a, and all the leaf nodes lies in the same layer, the depth of the tree, H, is

$$H \le \log_a(n-1) \le \log_a(n-1) + (2 - \log_a 2)$$

Thus the total times is at most

$$\begin{aligned} T &\le \log(b-1) \times H \\ &\le \log(b-1) \times \log_a(n-1) \\ &\le \lceil \log b \rceil \Big((\log_a(n-1) + (2 - \log_a 2) \Big) \end{aligned}$$

Hence proved.

(ii) From Ex. 3i we have the total number of comparison is

$$T = \lceil \log b \rceil \Big((\log_a(n-1) + (2 - \log_a 2) \Big)$$

By the O-notation,

$$2 \log b \in O(\log b) \tag{1}$$

At the same time, we know that because a is a constant,

$$\log_a \frac{n-1}{2} \in O(\log n) \tag{2}$$

Plus b is a constant, it's obvious that

$$\log b \cdot \log_a \frac{n-1}{2} \in O(\log n)$$

Hence adding equation (1) and (2) will produce the result

$$O(\log b) + O(\log n)$$