CS225 Assignment 7

Group 18: Xu Ke, Wang Dingkun, Qian shuaicun, Hua Bingshen

April 2021

1 Assignment 7 Exercise 1

Please see the codes in the attached file.

2 Assignment 7 Exercise 2

Please see the codes in the attached file.

3 Assignment 7 Exercise 3

3.1

According to the definition from lecture slides, we know that an (a,b)-tree with a, b $\in$ N, 0 $<$ a $\leq$ b is a tree with nodes labelled by several elements e $\in$ T , where T is a totally ordered set satisfying the following conditions:

1) every non-leaf node except the root has between a and b children

2) if the number of nodes is $>$ 2, the root has between 2 and b children

3) each non-leaf node v with k children is labelled by e1 $< \cdots <$ e_k-1 $\in$ T such that

a) all nodes in the i'th successor tree of v (2 $\leq$ i $\leq$ k - 1) contain only elements e$\in$T with e_i-1 <e< e_i

b) all nodes in the first successor tree of v contain only elements e $\in$ T with e $<$ e_1

c) all nodes in the last successor tree of v contain only elements e $\in$ T with e_k $<$ e

4) each leaf node v is labelled by e_1 $<\cdots<$$e_k$ $\in$ T with a$\leq$k$\leq$b

Thus in order to prove that the total number of comparisons in a search in an (a,b)-tree with n nodes is bounded by $\lceil logb \rceil (2 + log_a((n-1))/2)$, we need to consider the extreme situation in which the most comparisons are needed. Thus, we may assume that the height and size should be as large as possible to achieve this bound, and let's discuss this step by step:

Step 1: Find the largest height.

For a (a,b)-tree, it is clearly that it should have a root and any child-node of this root also works as the root for the subtree. Assume there are totally n nodes, if we exclude the root with degree a, there are clearly n-1 nodes left with degree ranging from 2 to b. Assume h is the height of the tree without the root, a is the degree of node. Then, we know that the number of leaves will be at most $2d^{h-1}$. And, thus it must satisfy that $n - 1 \geq 2d^{h-1}$. Thus, h$\leq$ $1 + log_a(n-1)/2$. Thus, if the root added, the total height will be h+1, for which we have h+1 $\leq 1 + log_a(n-1)/2$.

Step 2: Find the largest size.

If we want to find the make comparison at a given height, the largest time consumed would be the same as the time of the search path for the node. And it is clearly that for a (a,b)-tree, there are at most b items, and thus at most $\lceil logb \rceil$ comparisons needed due to binary search.

Hence, the most complex condition is that we have largest height and size. And by combining Step 1 and Step 2 together, we will get the upper bound, that is $\lceil logb \rceil (2 + log_a((n-1))/2)$

3.2

According to Ex3.1, $\lceil logb \rceil (2 + log_a((n-1))/2)$.

And according to the assumption that b$\leq$ 2a, we will have:

O$\lceil logb \rceil (2 + log_a((n-1))/2) = 2\lceil logb \rceil + \lceil logb \rceil log_a((n-1))/2) \leq 2\lceil logb \rceil + \lceil logb \rceil log_{b/2}((n-1))/2) \leq 2\lceil logb \rceil + \lceil logb \rceil log_b n) \leq 2\lceil logb \rceil + 2logn)$ and it is clear that $2\lceil logb \rceil + 2logn) \in O(logb) + O(logn)$, hence proved.