NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

SC1007 Data Structures and Algorithms **2021/22 Semester 2**

Solution 3: Tree and Binary Search

School of Computer Science and Engineering *Nanyang Technological University*

Q1 A sequence, $x_1, x_2, \ldots, x_n$, is said to be cyclically sorted if the smallest number in the sequence is x_i for some i, and the sequence, $x_i, x_{i+1}, \ldots, x_n, x_1, x_2, \ldots, x_{i-1}$ is sorted in increasing order. Design an algorithm to find the minimal element in the sequence in $\mathcal{O}(\log n)$ time. What is the worst-case scenario?

S1 Let us see the following examples of cyclically sorted sequence:

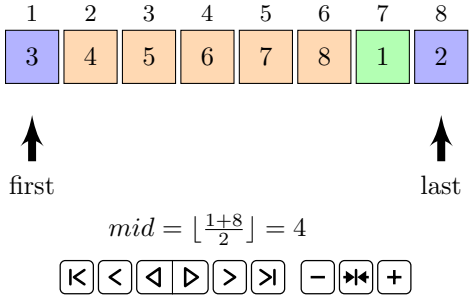
1	2	3	4	5	6	7	8
6	7	8	1	2	3	4	5

1	2	3	4	5	6	7	8
3	4	5	6	7	8	1	2

Given sequences above, $x'_1, x'_2, \ldots, x'_{j-1}, x'_j, x'_{j+1}, \ldots, x'_{m-1}, x'_m$, there exist an index, j that

$$x'_j < x'_{j+1} < \ldots < x'_m < x'_1 < \ldots < x'_{j-1}$$

How can we find out the index, j?
Let us select the middle element of the sequence, x'_{mid}. If the $x'_{mid} < x_m$ (the last element of the sequence), then the minimum definitely is in the first half ($x'_1, x'_2, \ldots, x'_{j-1}, x'_j$) (including itself). Otherwise, the minimum will be in the second half ($x'_{j+1}, \ldots, x'_{m-1}, x'_m$).



```
1  int minimum(int array[], int n, int m) \\ n and m are the indices of the 1st
     and last elements respectively
2  {
3    int mid;
4    if (m == n) return array[n];  \\ this is the min
5    else {
6     mid=(n+m)/2;
7     if (array[mid] < array[m])       \\ middle < last
8        return minimum(n, mid); \\ find min in 1st half
9     else
10       return minimum(mid + 1, m); \\ find min in 2nd half
11   }
12   }
```

Algorithm of Q1

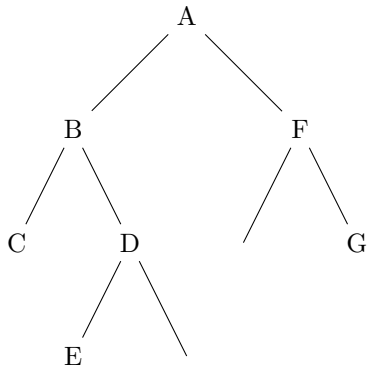
There is no difference. All cases have to run the same number of comparisons.

Q2 You are given a pre-order traversal and an inorder traversal of a binary tree. Draw the binary tree from the two traversal results.

pre-order: A, B, C, D, E, F, G

inorder: C, B, E, D, A, F, G

S2 The first visiting node of the pre-order traversal is the root. From the inorder traversal, you can find that C, B, E, D is in the left subtree and F, G is in the right subtree. The second visiting node of the pre-order traversal is the root of the left subtree if the left subtree is not empty. Otherwise, it will be the root of right subtree. From the inorder traversal, you can find that C is in the left subtree and E,D is in the right subtree. etc. The binary tree is



Q3 What is the upper bound of the height of an AVL Tree? Write it in asymptotic notation. The AVL tree is a binary tree in which the left and right subtrees of any node have heights that differ by at most 1.

S3 Let N_h denote the number of nodes of an AVL tree of height h. When $h = 0$, N_0 is 1. When $h = 1$, N_1 will be 2 or 3 nodes. We need at least 2 nodes to construct an AVL tree of height 1.

Since the maximum difference in height of both subtrees is 1, the minimum number of nodes to construct an AVL tree of height h is

$$N_h^{min} = N_{h-1}^{min} + N_{h-2}^{min} + 1$$

Similarly,

$$N_{h-1}^{min} = N_{h-2}^{min} + N_{h-3}^{min} + 1$$

By the method of the backward substitutions,

$$N_h^{min} = 2N_{h-2}^{min} + N_{h-3}^{min} + 2$$

we can approximate the minimum number of nodes,

$$\begin{aligned} N_h^{min} &> 2N_{h-2}^{min} \\ &> 2^2 N_{h-4}^{min} \\ &> 2^3 N_{h-6}^{min} \\ &\cdots \\ &> 2^k N_{h-2k}^{min} \end{aligned}$$

$$\begin{aligned} h - 2k &= 0 \\ k &= \frac{h}{2} \end{aligned}$$

Thus,

$$\begin{aligned} N_h^{min} &> 2^{\frac{h}{2}} \\ \log_2 N_h^{min} &> \log_2 2^{\frac{h}{2}} \\ 2\log_2 N_h^{min} &> h \\ h &= \mathcal{O}(\log N_h^{min}) \end{aligned}$$

If we use N_h^{min} nodes to construct an AVL of height h, the height cannot be more than $2\log N_h^{min}$. In other words, the upper bound of the height of an AVL tree is $\mathcal{O}(\log N_h^{min})$.