SOLUTIONS

10.1-7

Show how to implement a stack using two queues. Analyze the running time of the stack operations.

Solution

Stack can be implemented by always pushing the new element at front of Queue1, and when performing Pop operation, dequeue from Queue1. We use Queue2 to put every new element in front of Queue1.

Consider the below steps having two queues Q1 and Q2.
Push Operation

1.Enqueue(Q2,x)
2.Dequeue all the elements from Q1 and enqueue it to Q2
    for i in range 1 to Q1.length
    Enqueue(Q2,Dequeue(Q1))
3.Swap the names of Q1 and Q2

Pop Operation
Q1 will always contain only one element, which would be the last element inserted. Hence, a Dequeue operation on Q1 will behave as a pop operation in stack.

1. Dequeue(Q1)

The running time of Push operation will be $\theta(n)$ where n is the number of elements in stack and for the Pop operation, it is $\theta(1)$.

10-1

For each of the four types of lists in the following table, what is the asymptotic worst-case running time for each dynamic-set operation listed?

	unsorted, singly linked	sorted, singly linked	unsorted, doubly linked	sorted, doubly linked
Search(L, k)				
Insert(L, x)				
Delete(L, x)				
Successor(L, x)				
Predecessor(L, x)				
Minimum(L)				
Maximum(L)				

Solution

	unsorted, singly linked	sorted, singly linked	unsorted, doubly linked	sorted, doubly linked
Search(L, k)	$O(n)$	$O(n)$	$O(n)$	$O(n)$
Insert(L, x)	$O(1)$	$O(n)$	$O(1)$	$O(n)$
Delete(L, x)	$O(n)$	$O(n)$	$O(1)$	$O(1)$
Successor(L, x)	$O(n)$	$O(1)$	$O(n)$	$O(1)$
Predecessor(L, x)	$O(n)$	$O(n)$	$O(n)$	$O(1)$
Minimum(L)	$O(n)$	$O(1)$	$O(n)$	$O(1)$
Maximum(L)	$O(n)$	$O(n)$	$O(n)$	$O(n)$

Note :
1. Assuming that Successor is the next biggest element and Predecessor is the previous smallest element.
2. I have considered sorted lists are in ascending order.
3. Also, considering there is no tail pointer.

12.2-7

An alternative method of performing an inorder tree walk of an n-node binary search tree finds the minimum element in the tree by calling Tree-Minimum and then making $n - 1$ calls to Tree-Successor. Prove that this algorithm runs in $\Theta(n)$ time.

Solution

Best case : Tree will be a left stranded tree i.e The tree will only has left child, then it will first find the Tree-minimum in $\theta(n)$ time. Next it will traverse every edge only once by making (n-1) Tree-successor calls. So, best case time complexity will be $\Omega(n)$.

Worst case : It will first find Tree-minimum in $\theta(n)$ time. Then, it can traverse the edges at most twice, the time complexity will be $O(n)$.

Hence, the algorithm will run in $\theta(n)$ time.

12.3-1

Give a recursive version of the Tree-Insert procedure.

Solution

```
1  procedure Tree-Insert(x, z)
2  begin
3  if z.key < x.key and x.left != NIL
4      Tree-Insert(x.left , z)
5  else if z.key > x.key and x.right != NIL
6          Tree-Insert(x.right ,z)
7  z.parent = x
8  if z.key < x.key
9      x.left = z
10 else
11     x.right = z
12 end
```