Archit Agrawal
2020052307
Ankit Agrawal
Date
Page 1

CSI02 : Midsem Remote Exam

(1) When stack is implemented using an array so that the top element is always stored at index 0 of array, it has following dicadvantages :-

(i) The push() and pop() method becomes costlier as after inserting/removing the whole array will be shifted either to right or left which will take O(size) time, where size is the number of elements in the stack. Whereas, if we use last index of array as top element, push() and pop() methods are executed in O(1).

(ii) Array has a fixed size, so if we want to store more elements than its size, we cannot store them.

Advantages

There are no advantages as such because the isEmpty() and peek() method are executed in O(1) time for both the approaches (using 0 index as top and using index 'size' as top).

Archit Agrawal
2020052307
ArchitAgrawal

Date
Page 2

(2) ~~The~~ node

(2) Although the nodes in LinkedStack hold a reference for next stack element which increases the space needed to ~~the~~ store data, still the LinkedStack uses space more efficiently than an ArrayStack.
The reason for this is that ~~the~~ Array is not dynamic, i.e. it's size cannot change during the execution. Hence, there is an limit on the number of elements if we implement Stack using Arrays. However, LinkedList is dynamic, there is no limit in its size and hence any number of elements can be stored in a LinkedStack.

Hence, the given ~~claim~~ fact doesn't invalidate the claim that LinkedStack uses space more efficiently than an ArrayStack.

- Even if we use arraylist for implementing stack, we need to double the ~~size~~ capacity each time the arraylist is full.
But in linkedlist, capacity ~~is increased by any number~~ can be increased by 1 without any time wastage.
While, if we increase the capacity of array by 1 each time, it will take

O(size) time every time.

3. Create three new nodes named prevNode and currNode and nextNode.

prevNode → reference of previous Node of current Node

currNode → reference of current node

nextNode → reference of next node to current node.

- Initialise
  prevNode = null
  currNode = head
  nextNode = null

- Now, iterate through linked list using while loop and node currNode.
  In each iteration,
  → store the nextNode of currNode i.e. currNode.next in nextNode. Then, we store the reference of prevNode in the next of currNode ie.

  currNode.next = prevNode

  Now, we store the reference of currNode in prevNode.

- Finally, currNode = nextNode so as to continue traversing the linked

list till last element.

∴ In short, in each iteration of while loop, the four steps take place.

```
nextNode = curNode.next
curNode.next = prevNode
prevNode = curNode
curNode = nextNode
```

The time complexity of this algorithm is O(n).

(4). A stack can be implemented using two queues. Let stack to be implemented be 's' and queues used to implement be 'q1' and 'q2'

Stack 's' can be implemented in two ways:-

(i) By making push method costly
(ii) By making pop method costly

By making pop method costly

Process:-

- In push operation, new element is always enqueued to q1.
- In pop operation, if q2 is empty then all the elements except the last, are moved to q2. Finally, the last element is dequeued from q1 and returned.

- push (s, x) operation:
  → enqueue x to q1

- pop (s) operation:
  → dequeue everything except last element from q_1 and enqueue to q_2
  → dequeue last item of q_1, the dequeued item is result, store it,
  → Swap names of q_1 and q_2
  → Return the element stored in step 2.

for example:-

→ Push '1' in stack

queue q_1 → 1
queue q_2

→ push '2' in stack

queue q_1 → 1, 2
queue q_2 →

→ push '3' in stack

queue q_1 → 1, 2, 3
queue q_2 →

→ pop() from stack

- dequeuing every element from q_1 and enqueuing to q_2

1st time

q_1 → 2, 3
q_2 → 1

2nd time

q_1 → 3
q_2 → 1, 2

Now, the size of q_1 is 1. Hence, dequeued the element from q_1 and return this value (do not enqueue this to q_2)

Archit Agrawal
2020052307
Archit Agrawal

Date
Page 8

∴ $q_1 \rightarrow$
$q_2 \rightarrow 1, 2$

- rename q_1 and q_2

∴ $q_1 \rightarrow 1, 2$
$q_2 \rightarrow$