Assignment 2

**C-2.1 Describe, in pseudo-code, a link-hopping method for finding the middle node of a doubly linked list with header and trailer sentinels and an odd number of real nodes between them. (Note: This method can only use link-hopping; it cannot use a counter.) What is the running time of this method?**

Answer:

|  |  |
| --- | --- |
| Algorithm findMiddle (L)  Input: List with an odd number of nodes  Output: Middle position of List  p:= L.first()  q:= L.last()  while p != q do  p = L.after(p)  q= L.before(q)  return p | O(1)  O(1)  O(n)  O(n)  O(n)  O(1)  Running Time T(n) = O(n) |

**C-2.2 Describe, in pseudo-code, how to implement the queue ADT using two stacks. What is the running time of the enqueue() and dequeue() methods in this case?**

Answer:

S1 := Empty Stack

S2 := Empty Stack

|  |  |
| --- | --- |
| Algorithm enqueue (o)  Input: Object to push  Output:  If size () = n - 1 then  Throw FullQueueException  S1.push(o)  Algorithm dequeue ()  Input:  Output: Object removed  If S2.isEmpty() then  White !S1.isEmpty() do  S2.push(S1.pop())  If s2.isEmpty() then  Throw EmptyQueueException  Return S2.pop() | O(1)  O(1)  O(1)  Running Time T(n) = O(1)  O(1)  O(n)  O(n)  O(1)  O(1)  O(1)  Running Time T(n) = O(n) |

C-2.3 Describe how to implement the **stack** ADT using **two queues**. What is the running time of the push() and pop() methods in this case?

1. Design a pseudo code algorithm to take a Sequence and remove all duplicate elements from the Sequence. Is the algorithm the same for both a List or a Sequence? Explain. Analyze your algorithm twice, once assuming it is a Sequence and once assuming it is a List. Which ADT is a better choice for this problem, i.e., does one version have a better running time over the other?
2. Describe a recursive algorithm for enumerating all subsets of the numbers {1,2,...,n}, i.e., the powerset of the elements of a Sequence; the result should be a Sequence containing Sequences. What is the running time of your method?

Answer:

R-2.1 Describe, using pseudo-code, implementations of the methods insertBefore(*p,e*) , insertFirst(*e*), and insertLast(*e*) of the List ADT, assuming the list is implemented using a doubly-linked list.

**Optional** if you have time:

C-2-5 Describe the structure and pseudo-code for an array-based implementation of the vector ADT that achieves *O*(1) time for insertions and removals at rank 0, as well as insertions and removals at the end of the vector. Your implementation should also provide for a constant-time elemAtRank method.