Homework lecture : Complextity analyses

1.

1. O(4nlogn + 2n) = O(nlogn)
2. O(3n + 100logn) = O(n)
3. O(n2 + 10n) = O(n2 )
4. O(210 )
5. O(4n) = O(n)
6. O(n3 )
7. O(2logn ) = O(n)
8. O(2n )
9. O(nlogn)

=> O(O() = O(3n + 100logn) = O(4n) < O(nlogn) = O(4nlogn + 2n) < O( + 10n) < O( < O(

2. Two different algorithms in pseudo-codes to calculate

- Algo 1: Iterative Approach

Func powerTwoN(n) :

res = 1

for I from 1 to n:

res \*= 2

return res

Complexity: O(n)

- Algo2: Recursive Approach

Func powerTwoN(n):

if n < 0 : return 1 / powerTwoN(-n)

else:

if n = 0 return 1

else if n = 1 return 2

else if n % 2 = 0 return powerTwoN(n/2) \* powerTwoN(n/2)

else return 2 \* powerTwoN(n/2) \* powerTwoN(n/2)

Complexity: O(logn)

3. Operations of queue data structure in pseudo-codes using an array

enqueue(x): add x to the back of the queue

dequeue: remove the front element of the queue

class Queue:

array arr

interger index = 0

Method enqueue(x) :

arr[index] = x

index++

Method dequeue:

if index = 0. print “queue is empty”

else:

for i from 0 to index-1:

arr[i] = arr[i+1]

index = index – 1

Method print:

for i from 0 to index-1: print arr[i]

- The complexity of enqueue: O(1)

- The complexity of dequeue: O(N)

- The complexity of print: O(N)

4. Operations of queue data structure in pseudo-codes using a linked list

Class Node:

data

Node next

Class LinkedList:

Node head

Node tail

Class Queue:

LinkedList ll

Method enqueue(x):

Node newNode

newNode.setData(x)

if ll.head = ll.tail = null:

ll.head = ll.tail = newNode

else:

ll.tail.next = newNode

ll.tail = newNode

Metnod dequeue:

if ll.head = null:

return null

else:

Node removeNode = ll.head

ll.head = ll.head.next

return removeNode

Method printLinkedList:

Node cur = head

while cur != null:

print cur.data

cur = cur.next

Complexity: enqueue(x): O(1)

dequeue: O(1)

printLinkedList: O(N)

5. Operations of stack data structure in pseudo-codes using an array

Class Stack:

array arr

integer index

Method push(x):

arr[index] = x

index++

Method pop :

index = index - 1

Method print:

for i from index-1 to 0: print arr[i]

The complexity: push(x): O(1)

pop: O(1)

print: O(N)

6. Operations of stack data structure in pseudo-codes using a linked list

Class Node:

data

Node next

Class LinkedList:

Node head

Node tail

class Stack:

LinkedList ll

Method push(x):

Node newNode;

newNode.data = x

if ll.head = ll.tail = null:

ll.head = ll.tail = newNode

else:

newNode.next = head

newNode = head

Method pop():

if ll.head = null:

return null

else:

ll.head = ll.head.next

Method printStack:

Node cur = ll.head

while cur != null:

print cur.data

cur = cur.next

Complexity: push(x): O(1)

- pop(): O(1)

- printStack: O(N)