

Week 3: PYQs

06 June 2024 09:50

Stacks and Queues:

1. Jan 2024, End Term

Question Label : Multiple Choice Question

The following sequence of stack operations is performed on a Stack:

1	Push(10)
2	Push(20)
3	Pop
4	Push(20)
5	Push(10)
6	Pop
7	Push(10)
8	Pop
9	Pop
10	Pop

The sequence of values popped from the Stack is:

Options :

 10, 10, 20, 10, 20

 20, 10, 10, 20, 10

 20, 20, 10, 10, 10

 10, 20, 10, 20, 10

Screen clipping taken: 16-06-2024 12:02

2. Jan 24, Quiz 1

Assume `s` is a stack and `q` is a queue. `push` and `pop` operations are usual stack operations, `enqueue` and `dequeue` are usual queue operations, and `isEmpty()` is a method that returns true if either the stack or the queue is empty. Assume that stack `s` and Queue `q` are empty initially.

```

1 for i in range(5,0,-1):
2     s.Push(i)
3     q.Enqueue(i)
4
5 while not q.isEmpty():
6     s.Push(q.Dequeue())
7
8 while not s.isEmpty():
9     q.Enqueue(s.Pop())
10
11 while not q.isEmpty():
12     print (q.Dequeue(),end = " ")

```

What is the output of the given code snippet?

Options :

- ☐ 1 2 3 4 5 5 4 3 2 1
- ☐ 5 4 3 2 1 1 2 3 4 5
- ☐ 5 4 3 2 1 5 4 3 2 1
- ☐ 1 2 3 4 5 1 2 3 4 5

Screen clipping taken: 15-06-2024 20:32

3. Sep 23, End Term

Consider the function `dosomething` given below:

```

1 def dosomething(Q,R):
2     if (not Q.isEmpty()):
3         v = Q.dequeue()
4         dosomething(Q,R)
5         R.enqueue(v)

```

Assume `Q` and `R` are two queues. `enqueue` and `dequeue` are usual queue operations and `isEmpty()` is a method that returns `True` if the queue is empty, otherwise `False`.

Let `Q = [1, 2, 3, 4, 5]` (where 5 is the last inserted element in the queue) and `R = []` initially. What is the state of `Q` and `R` after the above `dosomething(Q, R)` is executed?

Options :

- ☐ `Q = [1, 2, 3, 4, 5]` and `R = [1, 2, 3, 4, 5]`
- ☐ `Q = []` and `R = [1, 2, 3, 4, 5]`

`Q = [] and R = [5, 4, 3, 2, 1]`

`Q = [1, 2, 3, 4, 5] and R = [5, 4, 3, 2, 1]`

Screen clipping taken: 16-06-2024 12:06

4. Sep 23, Quiz 1

Question Label : Multiple Choice Question

Let `s` be a stack and `q` be a queue supporting the following operations:

Stack operation:

- `Push(d)` : Insert element `d` in stack
- `Pop()` : Remove the element from the stack and return the removed element

Queue Operation:

- `Enqueue(d)` : Insert element `d` in queue
- `Dequeue()` : Remove the element from the queue and return the removed element

Consider the following function:

```
1 def fun(s,q):
2     if (not s.isempty()):
3         q.Enqueue(s.Pop())
4         fun(s,q)
5         s.Push(q.Dequeue())
```

What operation is performed by the above function `fun(s,q)` ? Suppose initially stack `s` has `n` elements and queue `q` is empty.

Options :

☐ Leaves the stack `S` unchanged

☐ Reverses the order of the elements in the stack `S`

☐ Swap the top and bottom element of the stack `S`, keeping the other elements in the same order

☐ Empties the stack `S`

Screen clipping taken: 15-06-2024 20:36

Hashing :

1. Jan 2024, End Term

Question Label : Multiple Choice Question

A hash table of size 9 (index 0 to 8) uses open addressing with hash function $h(k) = k \bmod 9$, and linear probing. The following elements are added into the hash table, which was initially empty.

18, 21, 90, 31, 45 and 55

The key value 55 is stored at which index of the hash table?

Options :

 2

 3

 4

 5

Screen clipping taken: 16-06-2024 12:03

2. Jan 24, Quiz 1

Question Label : Short Answer Question

Linear probing is an open addressing scheme in computer programming for resolving hash collisions in hash tables. Linear probing takes the original hash index and increments the value by 1 until a free slot is found.

A hash table contains 8 buckets indexed from 0 to 7 and uses linear probing to resolve collisions. The key values are integers and the hash function used is $key \bmod 8$. If key values 25, 87, 48, 64, 11 are inserted in to the table in the given order, at what index would the key value 120 be inserted after them?

Response Type : Numeric

Evaluation Required For SA : Yes

Show Word Count : Yes

Answers Type : Equal

Text Areas : PlainText

Possible Answers :



Screen clipping taken: 15-06-2024 20:33

3. Sep 23, End Term

Question Label : Multiple Choice Question

Linear probing is an open addressing scheme in computer programming for resolving hash collisions in hash tables. Linear probing operates by taking the original hash index and adding successive values linearly until a free slot is found.

A hash table of size **10** (indexed from 0 to 9) initialized with **None**, uses linear probing to resolve collisions. The key values are integers and the hash function used is **key mod 10**. Values **43, 65, 62, 23, 42, and 54** are stored in the given order in the hash table.

What is the sequence of elements (from index 0 to 9) in the hash table?

Options :

☐ None, None, 62, 23, 43, 65, 42, 54, None, None

☐ None, None, 62, 43, 23, 65, 42, 54, None, None

☐ None, None, 62, 43, 23, 54, 42, 65, None, None

☐ None, None, 62, 43, 23, 42, 54, 65, None, None

Screen clipping taken: 16-06-2024 12:07

4. Sep 23, Quiz 1

Question Label : Short Answer Question

Linear probing is an open addressing scheme in computer programming for resolving hash collisions in hash tables. Linear probing takes the original hash index and increments the value by 1 until a free slot is found.

A hash table contains 13 buckets(indexed from 0 to 12) and uses linear probing to resolve collisions. The key values are integers and the hash function used is **key mod 13**. If key values 14, 55, 144, 83, 122, 131 are inserted into the table, in what index would the key value 131 be inserted?

Response Type : Numeric

Evaluation Required For SA : Yes

Show Word Count : Yes

Answers Type : Equal

Text Areas : PlainText

Possible Answers :



Screen clipping taken: 15-06-2024 20:36

Linked Lists and Arrays:

1. May 22, Quiz 1

CORRECT MARKS : 5

Question Label : Multiple Choice Question

According to the conventional definition, an array is a fixed size data structure whose values are contiguously located in memory, whereas a linked list is a dynamic collection (size can change) of values that are not contiguously located in memory.

Considering the above definitions, if we use a Binary search algorithm to find a value from a linked list, then what would be the worst-case time complexity of Binary search?

Options :

☐ $O(n \log n)$

☐ $O(n^2)$

☐ $O(n)$

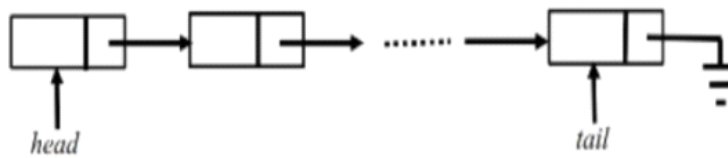
☐ $O(\log n)$

Screen clipping taken: 16-06-2024 12:00

2. Sep 23, Quiz 1

```
1 class Node:
2     def __init__(self,data):
3         self.data = data
4         self.next = None
```

Consider the following linked list structure, where each node is an object of class `Node` and it has a `head` pointer that points to the first node of the linked list and a `tail` pointer that points to the last node of the linked list.



Which of the following operations on a given linked list requires traversal of the entire list?

Options :

- ☐ Insert a new node at the beginning
- ☐ Delete a node from the end
- ☐ Insert a new node at the end
- ☐ Delete a node from the beginning

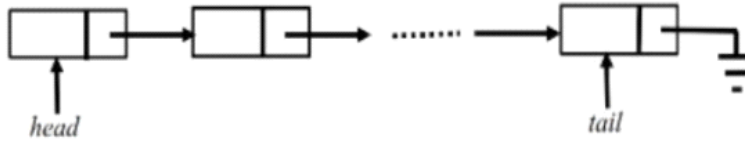
Screen clipping taken: 15-06-2024 20:34

3. May 22 (Quiz 1)

Question Label : Multiple Choice Question

```
1 class Node:
2     def __init__(self,data):
3         self.data = data
4         self.next = None
```

Consider an implementation of a singly linked list, where each node is created using the given class `Node`. Suppose it has a `head` pointer that points to the first node of the linked list and a `tail` pointer that points to the last element of the linked list.



Suppose we want to perform the following operations on the given linked list:-

1. Insertion of the new node at the front of the linked list.
2. Insertion of the new node at the end of the linked list.
3. Deletion of the first node of the linked list.
4. Deletion of the last node of the linked list.

Which of the following option represents the correct complexity for each operation?

Options :

☐ 1 – $O(1)$, 2 – $O(n)$, 3 – $O(1)$, 4 – $O(1)$

☐ 1 – $O(1)$, 2 – $O(1)$, 3 – $O(1)$, 4 – $O(1)$

☐ 1 – $O(1)$, 2 – $O(n)$, 3 – $O(1)$, 4 – $O(n)$

☐ 1 – $O(1)$, 2 – $O(1)$, 3 – $O(1)$, 4 – $O(n)$

Screen clipping taken: 06-06-2024 09:52

Question Label : Multiple Choice Question

Consider a linked list made up of nodes whose structure is defined by the following class

```
1 class Node:
2     def __init__(self, value):
3         self.value = value
4         self.next = None
```

Assume we have a class `LinkedList` in which `head` refers to the first node of the linked list. A method `fun(self, curr_node, prev_node)` is defined in the `LinkedList` class, as given below:

```
1 def fun(self, curr_node, prev_node):
2     if curr_node.next is None:
3         self.head = curr_node
4         curr_node.next = prev_node
5         return
6
7     temp = curr_node.next
8     curr_node.next = prev_node
9
10    self.fun(temp, curr_node)
```

The initial state of the linked list before calling `fun` was: 34, 12, 67, 9, 12, 4

What would be the state of the linked list after calling `fun(l.head, None)`, where `l` is the `LinkedList` object?

Options :

- ☐ 12, 34, 9, 67, 4, 12
- ☐ 4, 12, 9, 67, 12, 34
- ☐ 4, 12, 67, 9, 12, 34
- ☐ None of these

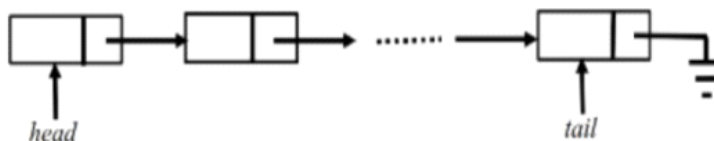
Screen clipping taken: 06-06-2024 09:54

5. Sep 22, Quiz 1

Question Label : Multiple Choice Question

```
1 class Node:
2     def __init__(self, data):
3         self.data = data
4         self.next = None
```

Consider an implementation of a singly linked list, where each node is created using the given class `Node`. Suppose it has a `head` variable that points to the first node of the linked list and a `tail` variable that points to the last element of the linked list.



Suppose we want to perform the following operations on the given linked list:-

1. Insertion of the new node at the first position of the linked list.
2. Insertion of the new node at the last position of the linked list.
3. Deletion of the first node of the linked list.
4. Deletion of the last node of the linked list.

Which of the above operation can be performed in **constant time** $O(1)$?

Options :

Which of the above operation can be performed in **constant time** $O(1)$?

Options :

☐ 1, 2 and 4

☐ 1, 2 and 3

☐ 2, 3 and 4

☐ 1, 2, 3 and 4

Screen clipping taken: 06-06-2024 10:02

Quick Sort and Other Sorting Algorithms:

1. Jan 23, Quiz 1

Question Label : Multiple Choice Question

Consider that **Quick sort** is applied on a list of size n which is already sorted. What will be the asymptotic running time of Quick sort if the pivot is taken to be

I) Middle element II) Last element

Choose the correct option corresponding to the correct pair of complexities for both pivots.

Options :

☐ I : $O(n^2)$ and II : $O(n \log n)$

☐ I : $O(n^2)$ and II : $O(n)$

☐ I : $O(n \log n)$ and II : $O(n \log n)$

☐ I : $O(n \log n)$ and II : $O(n^2)$

Screen clipping taken: 06-06-2024 10:07

2. Jan 2024, Quiz 1

Consider the below **Quick sort** algorithm to sort elements in ascending order using first element as pivot.

```
1 def partition(L, lower, upper):
2     # Select first element as a pivot
3     pivot = L[lower]
4     i = lower
5     for j in range(lower+1, upper+1):
6         if L[j] <= pivot:
7             i += 1
8             L[i], L[j] = L[j], L[i]
9     L[lower], L[i] = L[i], L[lower]
10    # Return the position of pivot
11    return i
12 def quicksort(L, lower, upper):
13    if (lower < upper):
14        pivot_pos = partition(L, lower, upper);
15        # Call the quick sort on leftside part of pivot
16        quicksort(L, lower, pivot_pos-1)
17        # Call the quick sort on rightside part of pivot
18        quicksort(L, pivot_pos+1, upper)
19    return L
```

Which of the below input sequence will require the **maximum** number of comparisons?

Options :

☐ [22, 25, 56, 67, 89]

☐ [52, 25, 76, 67, 89]

☐ [22, 25, 76, 67, 50]

☐ [52, 25, 89, 67, 76]

Screen clipping taken: 15-06-2024 20:31

3. Sep 22, Quiz 1

Question Label : Multiple Choice Question

Consider an input list L of n distinct elements, and the aim is to sort it in increasing order. Which of the following statement(s) is/are **true**?

1. Input in increasing order is the worst case for Insertion sort, but not for Quick sort.
2. Input in increasing order is the worst case for Quick sort, but not for Insertion sort.
3. Input in decreasing order is the worst case for both Quick sort and Insertion sort.

Options :

☐ 1 and 2

☐ 1 and 3

☐ 2 and 3

☐ 1, 2 and 3

Screen clipping taken: 06-06-2024 09:57


4. May 22, Quiz 1

Correct marks : 0

Question Label : Multiple Choice Question

What is the recurrence and time complexity for the worst case behaviour of **Quick Sort** ?

Options :

 Recurrence is $T(n) = 2T(n - 1) + O(n)$ and time complexity is $O(n^2)$

6406531153132. ✓ Recurrence is $T(n) = T(n - 1) + O(n)$ and time complexity is $O(n^2)$

6406531153133. ✗ Recurrence is $T(n) = T(n - 1) + O(1)$ and time complexity is $O(n)$

6406531153134. ✗ Recurrence is $T(n) = 2T(n/2) + O(n)$ and time complexity is $O(n \log n)$

Screen clipping taken: 16-06-2024 11:58

5. May 22, Quiz 1

Question Label : Multiple Choice Question

Consider the following `partition` function, which uses the last element of the list as the pivot.

```
1 def partition(L, low, high):
2     i = low - 1
3     pivot = L[high]
4     for j in range(low, high):
5         if L[j] <= pivot:
6             i = i + 1
7             L[i], L[j] = L[j], L[i]
8     #putting the pivot element in its appropriate position
9     L[i+1], L[high] = L[high], L[i+1]
10
11 L = [1, 4, 8, 2, 9, 3, 6]
12 partition(L, 0, len(L)-1)
```

What will be the state of the list `L` after the `partition` function terminates?

Options :

6406531153135. ✖ [1, 4, 2, 3, 6, 9, 8]

6406531153136. ✖ [1, 4, 2, 8, 9, 3, 6]

6406531153137. ✔ [1, 4, 2, 3, 6, 8, 9]

6406531153138. ✖ [1, 2, 3, 4, 6, 8, 9]

Screen clipping taken: 16-06-2024 12:01