Big O Notation

Question1: What is the O(nlog n + 3n+1) equal to ?

1) nLogn
2) 1
3) 3n
4) n

Question2: is this true or false : $O(2^n) > O(n!)$?

1)True
2) False

Question3: in which case the number of execution steps remains the same regardless of the size of the input?

1)Exponential Time
2) Linear Time
3) Polynomial Time
4) Constant Time

Question4: What is the Big O of this code?

fib(n)=fib(n-1)+fib(n-2)

```
 int fibonacci(int num) {
    if (num <= 1) return num;
     return fibonacci(num - 2) + fibonacci(num - 1);
}
```

1)n^2
2) 2^n
3)n!
4)nlogn

Question4: What is the Big O of this code?

```
def my_func( arr) {

    for  i in range(len(arr)):
          print(arr[i])

   for  i in range(len(arr)):
          print(arr[i])

   print('done') //
}
```

1)n^2
2) 2n
3)2n+1
4)n

Question5: What is the computational Complexity of the python code below in terms of Big O Notation? (// operator in python is integer division)

```
k=0
for i in range(n//2,n):  n/2
    j=1
    while (j<=n): logn
     j = j * 2
            k=k+j
```

1. O(n)
2. O(nLogn)
3. O(n^2Logn)
4. O(n^2)

Arrays, Stacks, Queues

Question1: Which one of the following is not in the Stack ADT?
1)push
2)pop
3)peek
4)enqueue

Question2: Which of the following uses the FIFO method?
1)Queue
2)stack
3)linked list
4)Heaps

Question3: Consider the following operation performed on a stack and a queue:

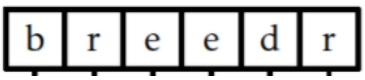
```
stack.Push(1);
stack.Pop();
stack.Push(2);
stack.Push(3);
queue.enqueue(stack.Pop())
queue.enqueue(stack.peek())
```

stack.Push(4)

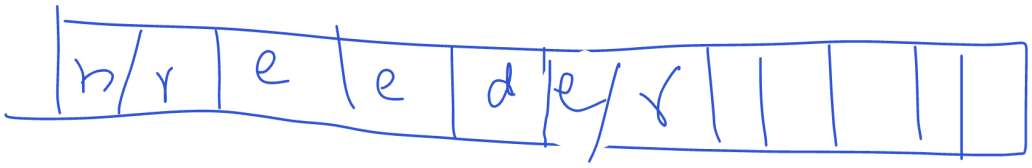
After the completion of all operations, what elements are stored on stack and queue?
1)stack :4, queue: 3,2
2)stack :4, queue: 2,3
3)stack: (empty), queue: 3,2
4)stack :4,2, queue: 3,2

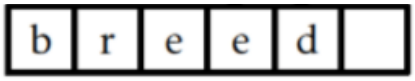
Question6: Which method is called in a POP() operation for stacks?
1)IsEmpty()
2)IsFull()
3)Both

Question7: How will the array below look like after we call add(5,e) arrayStack operation?

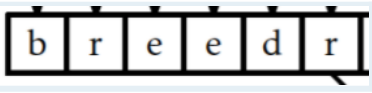


1)





2)

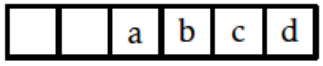
3) 

Question8: How do we implement push and pop in ArrayStack using the add and remove functions?

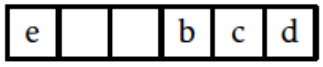
1) push(x): add(n,x) pop(): remove(n - 1)
2) push(x): add(n-1,x) pop(): remove(n)
3) push(x): add(0,x) pop(): remove(n-1)

Question9) how will this circular arrayqueue look like after these operations?
add(e)
remove()

$j = 2, n = 4$



$j = 3, n = 4$



Question10) in a circular arrayqueue which expression returns the element at location i? (a is the array and j is the variable which keeps track of the front of the queue)

1) a[(j + i) mod length(a)]
2) a[i mod length(a)]
3) a[(n + i) mod length(a)]

What is the computational complexity of add(i,x) in ArrayDeque?

1) in O(1 + min{i,n-i})
2)O(logn)
3)O(n)
4)O(n-i)

Linked Lists:

Question1: The following code shows the linked SLList implementation for the stack push operation, fill the missing line after the *then* command?

1)head<--u
2)tail<--u
3)tail<--nil

```
push(x)
    u ← new_node(x)
    u.next ← head
    head ← u
    if n = 0 then

    n ← n + 1
    return x
```

Question2: What does this function do in a doubly linked list?

unknown_function(i):

```
    if i < n/2 then
        p ← dummy.next
        repeat i times
            p ← p.next
    else
        p ← dummy
        repeat n − i times
            p ← p.prev
    return p
```

1) get_node(i)
2) add(i)
3) Resize

Question3: What does this code in DLList do?

$$w.prev.next \leftarrow w.next$$
$$w.next.prev \leftarrow w.prev$$

1) Adds a new node after w
2) Removes node w
3) Does not modify the list

Question4: What is the functionality of the following Java code?

```
public void function(Node node)
{
      if(size == 0)
            head = node;
      else
      {
            Node temp,cur;
            for(cur = head; (temp = cur.next)!=null; cur = temp);
            cur.Next=node;
      }
      size++;
}
```

a) Inserting a node at the beginning of the list
b) Deleting a node at the beginning of the list
c) Inserting a node at the end of the list
d) Deleting a node at the end of the list

Question4: What is the functionality of the following code?

```
public int function(Node head)
{
      int m = 0;
      Node cur = head;
      while( (cur!=null) && (cur.getNext()!=null))
      {
          m++;
          cur = cur.getNext().getNext();
      }
      return m;
}
```

1)returns the size of the list
2)returns the size of the list divided by 2
3)returns the number of non null elements in the list

Question5: which one of the following operations depends on the length of the list in an SLList?
1)Deleting the first element
2)deleting the last element
3)inserting at the beginning

Question6: which of the following is easier to do in a DLLlist compared to an SLList?

1) Deleting a given node
2) Searching for an element if it exists in the list
3) Process all nodes in the list

Hashtables

Question1: The keys 22, 38, 13, 2, 3, 23, 5 and 15 are inserted into an initially empty hash table of length 10 using open addressing with linear probing with hash function h(k) = k mod 10. Which of the following will be the resulting hashtable?

A)

0	
1	
2	22
3	13
4	2
5	3
6	23
7	5
8	38
9	15

B)

0	
1	
2	22,2
3	13,3,23
4	
5	5,15
6	
7	
8	38
9	

C)

0	
1	
2	2
3	23
4	
5	15
6	
7	
8	38
9	

1. A (Correct Answer)
2. B
3. C
4. None of the Above
5.

Question2: Assume that we have a hash table with ten buckets as shown in the figure below and that we have inserted the keys a1 to a6 into this hashtable using an open addressing with linear probing hashing function. What is the maximum number of comparisons needed in order to search for an item that is not present ?

0	a4
1	a5
2	a3
4	a1
5	
6	
7	
8	a2
9	a6

1. 1
2. 2
3. 3
4. 6 (Correct Answer)

Question3 :(from geeksforgeeks): A hash table of length 10 uses open addressing with hash function h(k)=k mod 10, and linear probing. After inserting 6 values into an empty hash table, the table is as shown below.

0	
1	
2	42
3	23
4	34
5	52
6	46
7	33
8	
9	

Which one of the following choices gives a possible order in which the key values could have been inserted in the table?
1)46, 42, 34, 52, 23, 33
2)34, 42, 23, 52, 33, 46
3)46, 34, 42, 23, 52, 33
4)42, 46, 33, 23, 34, 52

Question4: Which one of the following hash functions on integers will distribute keys most uniformly over 10 buckets numbered 0 to 9 for i ranging from 0 to 2020?

1)h(i) =i^2 mod 10
2)h(i) =i^3 mod 10
3)h(i) = (11 ∗ i^2) mod 10
4)h(i) = (12 ∗ i) mod 10

Solution:
Since mod 10 is used, the last digit matters. If you do cube all numbers from 0 to 9, you get following

Number	Cube	Last Digit in Cube
0	0	0
1	1	1
2	8	8
3	27	7
4	64	4
5	125	5
6	216	6

7	343	3
8	512	2
9	729	9

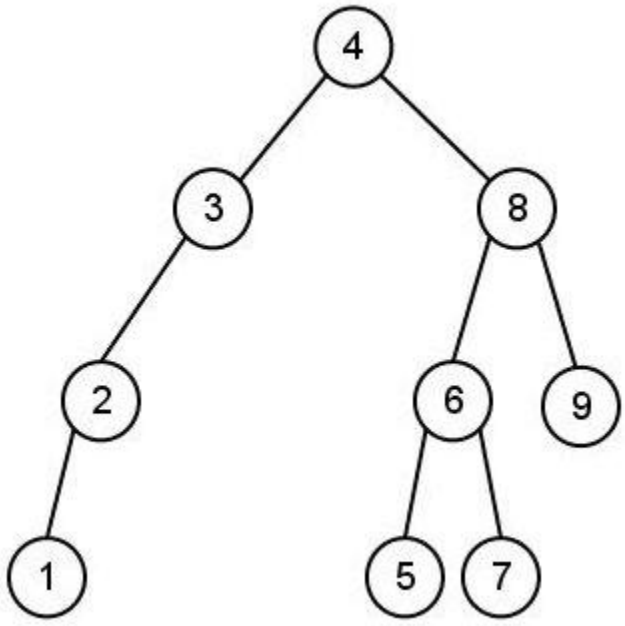
Therefore all numbers from 0 to 2020 are equally divided in 10 buckets.

Binary Trees

Question1: Which of the traversals in binary search tree gives sorted list

A. Post-order
B. Pre-order
C. In-order

Question2: Consider the following tree, What is the pre-order traversal of the tree?



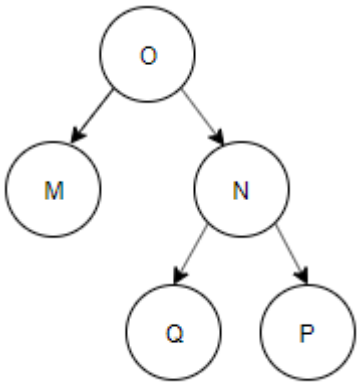
Question3:. The number of edges from the node to the deepest leaf is called _____ of the tree.

a) Height
b) Depth
c) Length
d) Width

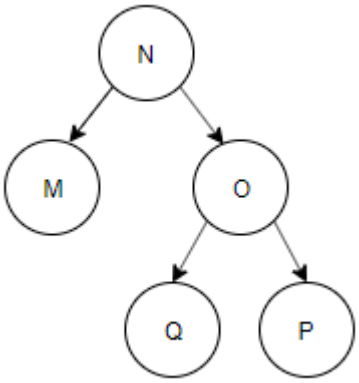
Question4: What is the average case time complexity for finding the height of the binary tree?
a) h = O(loglogn)
b) h = O(nlogn)
c) h = O(n)
d) h = O(log n)

Question5 (https://www.sanfoundry.com/data-structure-questions-answers-binary-tree-properties/): Construct a binary tree by using postorder and inorder sequences given below.
Inorder: N, M, P, O, Q
Postorder: N, P, Q, O, M

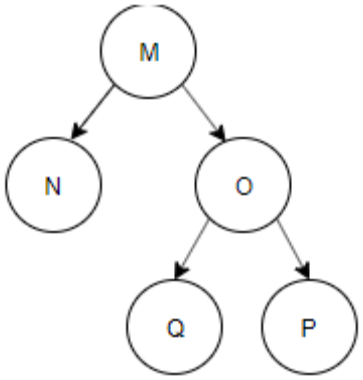
a)



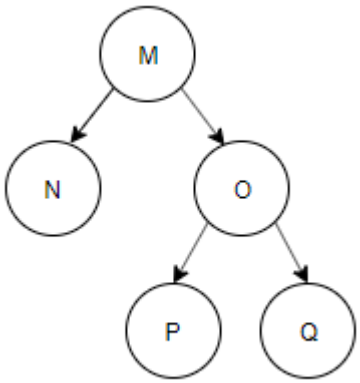
b)



c)



d)



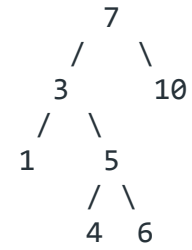
Question 6: WHat does this function do in a Binary search tree?

Function(x):

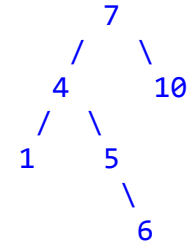
```
w ← r
while w ≠ nil do
    if x < w.x then
        w ← w.left
    else if x > w.x
        w ← w.right
    else
        return w.x
return nil
```

Answer: find_eq(x)

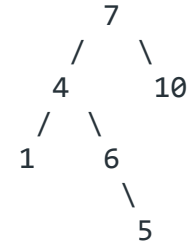
Question 7: **Question4: How will this BST tree look like after deleting node 3**



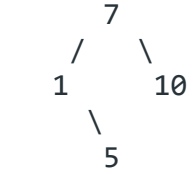
1) (Correct Answer)



2)



3)



4)