MCQs

1. **What is the complexity of adding an element to the heap? (Easy) (crs-be-programming)**
    a. O(n^2)
    b. O(1)
    c. O(log n) (correct)
    d. O(n)

2. **Which notation do we consider while calculating the average case time complexity of an algorithm? (Easy) (crs-be-programming)**
    a. Big Omega
    b. Big O
    c. Big Theta (correct)
    d. None of them

3. **What is the time complexity of the below snippet? (Medium) (crs-be-programming)**

```
for(let i = 0; i < n; i++){
        for(let j; j < n; j *= 2){
                //
        }
}
```

    a. O(n^2)
    b. O(logn)
    c. O(n)
    d. O(nlogn) (correct)

4. **What is the worst case time complexity of the KMP matching algorithm? (Medium) (crs-be-programming)**
    a. O(n) (correct)
    b. O(n^2)
    c. O(nlogn)
    d. O(2^n)

5. **Which of the following is a valid type of heap?(easy) (crs-be-programming)**
    a. Min heap
    b. Max heap
    c. Both a and b (correct)
    d. None of the above

6. **What happens when the backtracking algorithm reaches a complete solution?(Medium) (crs-be-programming)**
    a. It backtracks to the root
    b. It either stops or continue searching for other possible solutions (correct)
    c. It traverses from a different route

   d. Recursively traverses through the same route

7. **In which data structure element is inserted at one end called Rear and deleted at other end called Front.(easy) (crs-be-programming)**
   a. Stack
   b. Queue (correct)
   c. Both of the above
   d. Binary Tree

8. **Recursion uses more memory space than iteration because (Medium) (crs-be-programming)**
   a. It uses a stack for recursive calls.
   b. every recursive call has to be stored.
   c. It uses a queue for recursive calls
   d. Only a and b (correct)

9. **Which sorting algorithm sort uses counting sort as a subroutine to sort an array of numbers?(Difficult) (crs-be-programming)**
   a. Insertion Sort
   b. Radix Sort (correct)
   c. Selection Sort
   d. Bubble Sort

10. **Which of the data structures can be used to implement the stack data structure?(Medium) (crs-be-programming)**
    a. Array
    b. Linkedlist
    c. Both a and b (correct)
    d. None of these

11. **Which of the following points is/are true about Singly Linked List data structure when it is compared with array?(Medium) (crs-be-programming)**
    a. Random access is faster in LinkedList as compared to Arrays.
    b. Insertion at the tail can be done in O(1)
    c. The size of the array has to be pre-decided, linked lists can change their size any time. (correct)
    d. All of these

12. **What is the output of the following function for `node` pointing to the first node of the following linked list? 1->2->3->4->5->6 (Difficult) (crs-be-programming)**

```
function fun(node)
{
  console.log(node.data);

  if(node.next != NULL )
    fun(node.next.next);
  console.log(node.data);
}
```

a. 1 4 6 6 4 1
b. 1 3 5 1 3 5
c. Runtime error (correct)
d. 1 3 5 5 3 1

13. To implement a Stack using Queue, how many queues would be required? (Difficult) (crs-be-programming)

a. 1
b. 2 (correct)
c. 3
d. 4

14. Which of the following LinkedList stores previous and next node addresses without forming circle? (easy) (crs-be-programming)

a. Singly LinkedList
b. Doubly LinkedList (correct)
c. Circular LinkedList
d. None of the above

15. Which of the following is a technique in which we define two variables typically pointing to the first and last element(or second element) and is typically used for searching pairs in a sorted array and reduces the time complexity from O(n^2) to O(n)?(easy) (crs-be-programming)

a. Binary Search
b. Two pointers (correct)
c. Greedy Algorithm
d. None of these

16. Suppose we declare a 2d array of size n*m, what would be the considered time complexity of accessing an element in this context? (easy) (crs-be-programming)

a. $O(n^2)$
b. O(1) (correct)
c. O(nlogn)
d. O(n)

17. Which of the following operations is O(1) for an array of sorted data. You may assume that array elements are distinct. (Medium) (crs-be-programming)

a. Find the i^{th} largest element.
b. Delete an element.
c. Find the i^{th} smallest element.
d. Only a and c (correct)

18. Suppose we have a max heap and 10 elements in it. We print each element present in heap by popping them, in what order will the element be printed? (Medium) (crs-be-programming)

a. In increasing to decreasing order
b. In random order
c. Same as inserted order
d. In decreasing to increasing order (correct)

19. What will be the time complexity for the below function? (Medium) (crs-be-programming)

```
function doSomething(a, b)
{
        if (b==1)
                return a;
        else
                return a + doSomething(a,b/2);
}
```

a. O(logb) (correct)
b. O(b)
c. O(a+b)
d. O(a * b)

20. Suppose we are given a maze and an entry point, which of the following algorithms would help us come out? (Medium) (crs-be-programming)

a. Binary Search
b. Backtracking (correct)
c. Normal Recursion
d. Two pointers

21. What will be the output of the below code snippet? (Medium) (crs-be-programming)

```
function print(int n)
{
  if(n == 0) return;
  console.log(n/2);
  print(n/2);
}
print(10);
```

a. 5 2 1 0
b. 10 5 2 1
c. Stack overflow error
d. None of these (correct)

22. What is the worst case time complexity for search, insert and delete operations in a general Binary Search Tree?

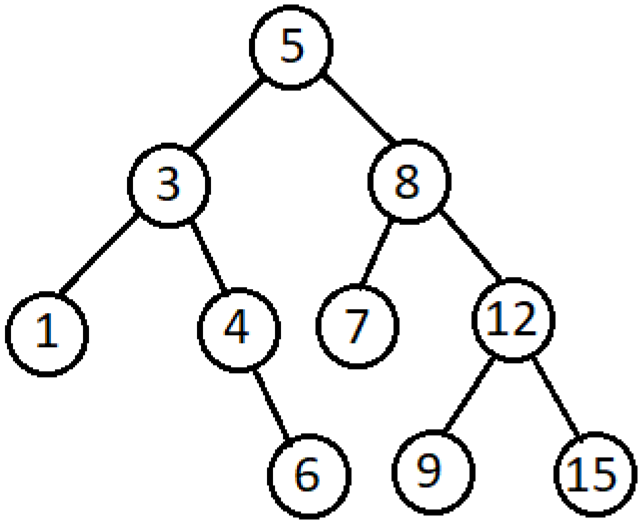
a. O(n) for all (correct)
b. O(Logn) for all

c. O(Logn) for search and insert, and O(n) for delete
d. O(Logn) for search, and O(n) for insert and delete

23. Which of the following traversal follows Left, Right, Root? (easy) (crs-be-programming)

a. Inorder Traversal
b. Preorder Traversal
c. Postorder Traversal (correct)
d. All of these

24. In the preorder traversal of the tree below, what will be printed after 8?(Difficult) (crs-be-programming)



a. 12
b. 7 (correct)
c. 9
d. 15

25. What is the time complexity of adding an element into BST? (Difficult) (crs-be-programming)

a. O(height)
b. O(number of nodes)
c. Both a and b (correct)
d. None of the above

26. If the elements “A”, “B”, “C” and “D” are placed in a stack and are deleted one at a time, in what order will they be removed? (medium) (crs-be-programming)

a. ABCD
b. DCBA (correct)
c. DCAB
d. ABDC

27. What is the term for deleting from a empty queue known as? (medium) (crs-be-programming)

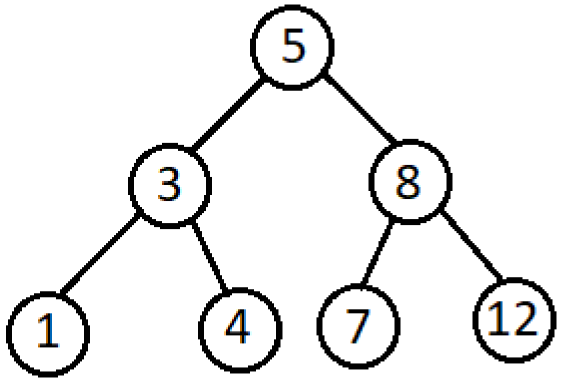
a. Overflow

b. Underflow (correct)
c. Null Pointer Exception
d. program won't be compiled

28. The number of edges from the root to the node is called ___________ of the tree.(Easy) (crs-be-programming)

a. Height
b. Depth (correct)
c. Length
d. Width

29. In the inorder traversal of the below tree, 8 will be printed after?(easy) (crs-be-programming)



a. 7 (correct)
b. 5
c. 12
d. 3

30. Which of the problems can be solved using a two pointers approach? (Medium) (crs-be-programming)

a. Two Sum problem
b. Reverse a String
c. Check Palindrome
d. All of these (correct)

Round 2

1. Max Product

Problem Statement
Given an array of integers nums, you have to choose two different indices i and j of that array. Return the maximum value of (nums[i])*(nums[j])

Constraint
- **2 <= nums.length <= 500**
- **1 <= nums[i] <= 10^3**

Input Format
- Space separated integers

Output Format
- Return top 2 max element product

Sample Input 1
9 5 12 7 8
Sample Output 1
108
Explanation of Sample 1
12 and 9 are top 2 max element and their product is 108

Sample Input 2
3 2
Sample Output 2
6
Explanation of Sample 2
3*2 = 6

Sample Input 3
1 2 3 4
Sample Output 3
12
Explanation of Sample 3
3 * 4 = 12

Solution:
```
process.stdin.resume();
process.stdin.setEncoding('utf8');

let inputString = '';
let currentLine = 0;

process.stdin.on('data', inputStdin => {
    inputString += inputStdin;
});

process.stdin.on('end', _ => {
    input = inputString.trim().split(" ").map(string => {
        return parseInt(string.trim());
    });

    console.log(maxProduct(input));
});

function maxProduct(input) {
  input.sort((a, b) => a-b);

  return input[input.length-1] * input[input.length-2];
}
```

Template

```
process.stdin.resume();
process.stdin.setEncoding('utf8');

let inputString = '';
let currentLine = 0;

process.stdin.on('data', inputStdin => {
    inputString += inputStdin;
});

process.stdin.on('end', _ => {
    input = inputString.trim().split(" ").map(string => {
        return parseInt(string.trim());
    });

    console.log(maxProduct(input));
});

function maxProduct(input) {
            //write your logic here
}
```

2. Merge two Sorted Array

Problem Statement

You will be given two arrays as input which are already sorted, merge them into a single array sorted in non-decreasing order.

Constraints

- 1 <= m, n <= 200 , m and n are length of two arrays nums1 and nums2
- 2 <= m + n <= 200
- -10^9 <= nums1[i], nums2[j] <= 10^9

Input Format

- Two lines, each line containing space separated integers

Output Format

- Print the new array

Sample Input 1

```
1 3 5 7
2 4 6 8
```

Sample Output 1

```
1 2 3 4 5 6 7 8
```

Explanation of Sample 1

If we merge both the arrays it will be become as the above output i.e. 1 2 3 4 5 6 7 8 , as we need to maintain the sorted order while merging both the arrays

Sample Input 2

```
1 1 1
2 3 4
```

Sample Output 2

```
1 1 1 2 3 4
```

Sample Input 3

```
1
1
```

Sample Output 3

```
1 1
```

Explanation of Sample 3

The unique elements are [1,2,3,4,5], and the sum is 15

Solution:

```
process.stdin.resume();
process.stdin.setEncoding('utf8');

let inputString = '';
let currentLine = 0;
```

```
process.stdin.on('data', inputStdin => {
    inputString += inputStdin;
});

process.stdin.on('end', _ => {
    inputString = inputString.trim().split("\n").map(string => {
        return string.trim().split(" ").map(x => parseInt(x));
    });

    console.log(mergeSortedArrays(inputString[0], inputString[1]));
});

function mergeSortedArrays(arr1, arr2) {
  let result = [];

  let i = 0, j = 0;
  while(i < arr1.length && j < arr2.length){
      if(arr1[i] <= arr2[j]){
          result.push(arr1[i++]);
      } else {
          result.push(arr2[j++]);
      }
  }
  while(j < arr2.length){
    result.push(arr2[j++]);
  }
  while(i < arr1.length){
    result.push(arr1[i++]);
  }
  return result.join(" ");
}
```

Template

```
process.stdin.resume();
process.stdin.setEncoding('utf8');

let inputString = '';
let currentLine = 0;

process.stdin.on('data', inputStdin => {
    inputString += inputStdin;
});
```

```
process.stdin.on('end', _ => {
    inputString = inputString.trim().split("\n").map(string => {
        return string.trim().split(" ").map(x => parseInt(x));
    });

    console.log(mergeSortedArrays(inputString[0], inputString[1]));
});

function mergeSortedArrays(arr1, arr2) {
  let result = [];
  //write your logic here

  return result.join(" "); // return space separated integers
}
```

3. Same Tree

Problem Statement

You will be given the roots of two binary trees, write a function to check if they are the same or not. They are considered the same if they are structurally identical, and the nodes have the same value.

Constraints

- The number of nodes in both trees is in the range [0, 100].
- $-10^4 <= node.data <= 10^4$

Input Format

- Two lines, each line containing space separated integers

Output Format

- Print the new array

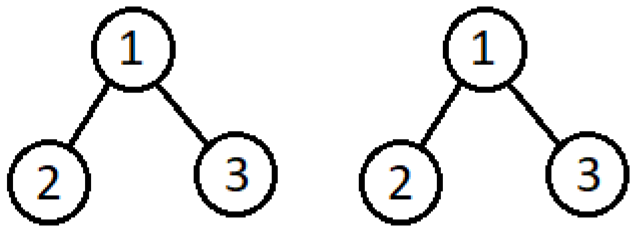
Sample Input 1

1 2 3 -1 -1 -1 -1
1 2 3 -1 -1 -1 -1

Sample Output 1

true

Explanation of Sample 1



Both the trees are identical, 1 is root in both trees, 2 is the left child and 3 is the right child of 1.

Sample Input 2

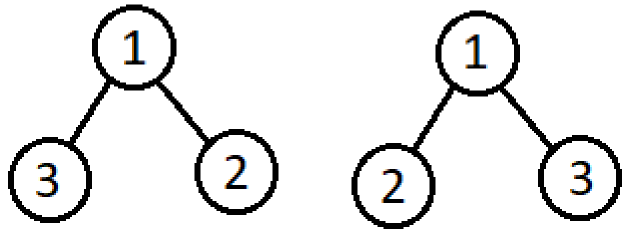
1 3 2 -1 -1 -1 -1

1 2 3 -1 -1 -1 -1

Sample Output 2

false

Explanation of Sample2



Trees are not identical

Sample Input 3

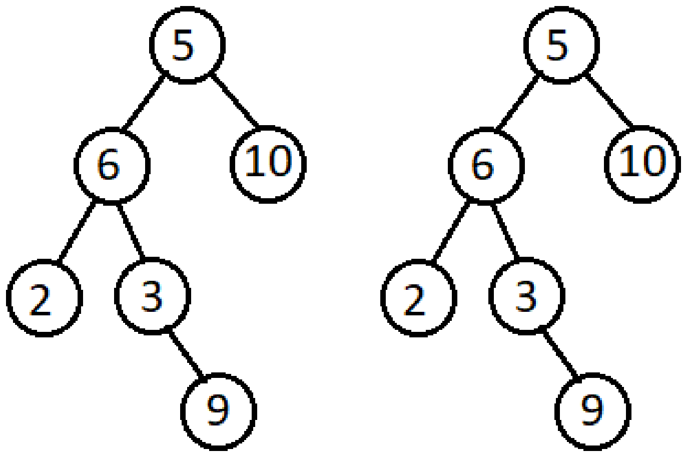
5 6 10 2 3 -1 -1 -1 -1 -1 9 -1 -1

5 6 10 2 3 -1 -1 -1 -1 -1 9 -1 -1

Sample Output 3

true

Explanation of Sample 3



Both the trees are identical, hence return true

Solution:

```
process.stdin.resume();
process.stdin.setEncoding('utf8');

let inputString = '';
let currentLine = 0;

process.stdin.on('data', inputStdin => {
    inputString += inputStdin;
});

process.stdin.on('end', _ => {
    inputString = inputString.trim().split("\n").map(string => {
        return string.trim().split(" ").map(x => parseInt(x));
    });

    main(inputString[0], inputString[1]);
});

class BinaryTreeNode{
    constructor(data){
        this.data = data;
        this.left = null;
        this.right = null;
    }
}

function takeInput(nodes){
    let index = 0;
    if(nodes[index] == -1) return null;
    let root = new BinaryTreeNode(nodes[index]);

    let queue = [];
    queue.push(root);

    while(queue.length > 0){
        let currNode = queue.shift();

        if(nodes[++index] != -1){
            let leftNode = new BinaryTreeNode(nodes[index]);
            currNode.left = leftNode;
            queue.push(leftNode);
        }
        if(nodes[++index] != -1){
            let rightNode = new BinaryTreeNode(nodes[index]);
            currNode.right = rightNode;
            queue.push(rightNode);
        }
    }
    return root;
}

function isIdentical(root1, root2) {
  //write your logic here
  if (root1 == null && root2 == null) {
    return true;
  }
  if(root1 == null || root2 == null || root1.data != root2.data){
      return false;
  }
  return isIdentical(root1.left, root2.left) && isIdentical(root1.right, root2.right);
}

function main(input1, input2){
    let tree1 = takeInput(input1);
    let tree2 = takeInput(input2)
    console.log(isIdentical(tree1, tree2));
}
```

Template

```
process.stdin.resume();
process.stdin.setEncoding('utf8');

let inputString = '';
let currentLine = 0;

process.stdin.on('data', inputStdin => {
    inputString += inputStdin;
});

process.stdin.on('end', _ => {
    inputString = inputString.trim().split("\n").map(string => {
        return string.trim().split(" ").map(x => parseInt(x));
    });

    main(inputString[0], inputString[1]);
});

class BinaryTreeNode{
    constructor(data){
        this.data = data;
        this.left = null;
        this.right = null;
    }
}

function takeInput(nodes){
    let index = 0;
    if(nodes[index] == -1) return null;
    let root = new BinaryTreeNode(nodes[index]);

    let queue = [];
    queue.push(root);

    while(queue.length > 0){
        let currNode = queue.shift();

        if(nodes[++index] != -1){
            let leftNode = new BinaryTreeNode(nodes[index]);
            currNode.left = leftNode;
            queue.push(leftNode);
        }
        if(nodes[++index] != -1){
            let rightNode = new BinaryTreeNode(nodes[index]);
            currNode.right = rightNode;
            queue.push(rightNode);
        }
    }
    return root;
}

function isIdentical(root1, root2) {
  //write your logic here
}

function main(input1, input2){
    let tree1 = takeInput(input1);
    let tree2 = takeInput(input2)
    console.log(isIdentical(tree1, tree2));
}
```

Feature:

1. **Design Twitter**

Design a simplified version of Twitter. Users can login, post tweets, follow/unfollow other users
POST /login -> signup is not required(optional), on login you can create user if not exist
POST /tweet -> post a tweet
POST /follow -> pass follow/unfollow status along with userId and other userId to which user wants to follow/unfollow

2. **Enhancement**

User A should be able to fetch the list of users followed by A i.e. A is following
Users should be able to see the news feed. Feeds should be ordered by date posted.
GET /newsfeed

3. **Optimization**

Users should be able to see the only 10 most recent tweets in their news feed. Also think why are we restricting it to 10. You will have to implement pagination for the news feed (in real system there is infinite scrolling on the UI when user scrolls we call api which is paginated and we keep loading feeds until the list is exhausted)

Boilerplate code is available :- https://github.com/VJ28/boilerplate_code

ROUND 3 Solution:
https://github.com/Ojhaakshat/relevel-twitter-backend-nodejs-development