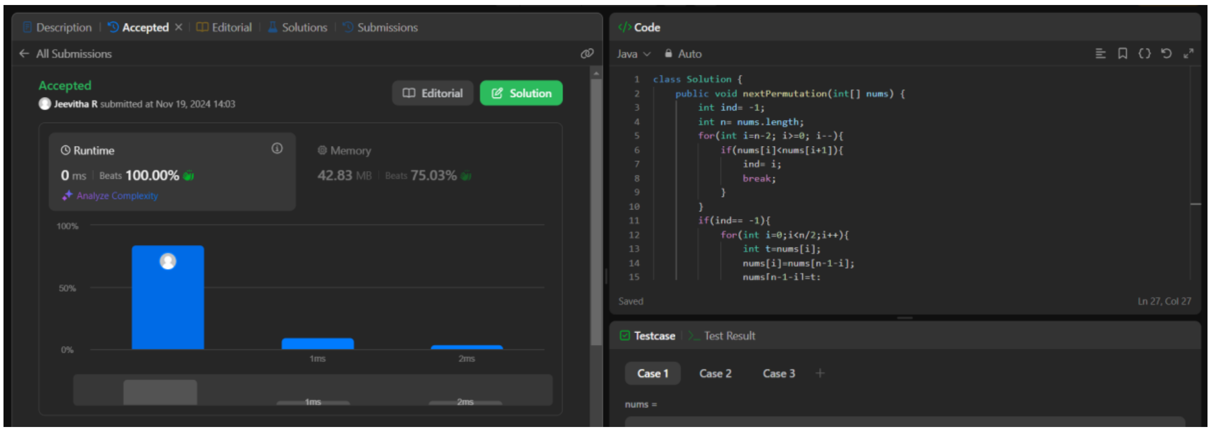
DSA PRACTICE QUESTIONS- DAY 7

Name: Jeevitha R
Reg No: 22IT040
Date: 19/11/2024

1.Next Permutation

```
class Solution {
   public void nextPermutation(int[] nums) {
      int ind= -1;
      int n= nums.length;
      for(int i=n-2; i>=0; i--){
         if(nums[i]<nums[i+1]){
            ind= i;
            break;
         }
      }
      if(ind== -1){
         for(int i=0;i<n/2;i++){
            int t=nums[i];
            nums[i]=nums[n-1-i];
            nums[n-1-i]=t;
         }
         return;
      }
      for(int i=n-1; i>=0; i--){
          if(nums[i]> nums[ind]){
            int temp= nums[ind];
            nums[ind]= nums[i];
            nums[i]= temp;
            break;
          }
      }
      int s=ind+1,l=n-1;
      while(s<l){
         int t=nums[s];
         nums[s]=nums[l];
         nums[l]=t;
         s++;
         l--;
      }
   }
}
```

Output:



Accepted
Jeevitha R submitted at Nov 19, 2024 14:03

Runtime
0 ms | Beats 100.00%

Memory
42.83 MB | Beats 75.03%

```java
class Solution {
    public void nextPermutation(int[] nums) {
        int ind= -1;
        int n= nums.length;
        for(int i=n-2; i>=0; i--){
            if(nums[i]<nums[i+1]){
                ind= i;
                break;
            }
        }
        if(ind== -1){
            for(int i=0;i<n/2;i++){
                int t=nums[i];
                nums[i]=nums[n-1-i];
                nums[n-1-i]=t;
```

Saved
Ln 27, Col 27

Time complexity: O(n)
Space complexity: O(1)

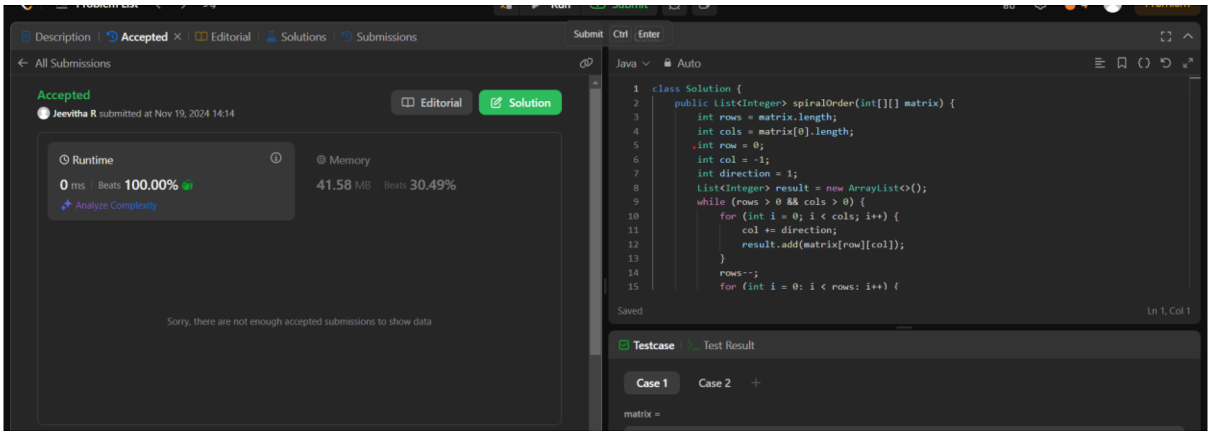
2. Spiral Matrix

```java
class Solution {
public ArrayList<Integer> spirallyTraverse(int mat[][]) {
    int rows = mat.length;
    int cols = mat[0].length;
    int row = 0;
    int col = -1;
    int direction = 1;
    ArrayList<Integer> result = new ArrayList<>();
    while(rows>0 && cols>0) {
        for(int i=0; i<cols; i++) {
            col += direction;
            result.add(mat[row][col]);
        }
        rows--;
        for(int i=0; i<rows; i++) {
            row += direction;
            result.add(mat[row][col]);
        }
        cols--;

        direction *= -1;
    }

    return result;
  }
}
```

Output:



Accepted
Jeevitha R submitted at Nov 19, 2024 14:14

Runtime
0 ms | Beats 100.00%

Memory
41.58 MB | Beats 30.49%

Sorry, there are not enough accepted submissions to show data

```
class Solution {
    public List<Integer> spiralOrder(int[][] matrix) {
        int rows = matrix.length;
        int cols = matrix[0].length;
        int row = 0;
        int col = -1;
        int direction = 1;
        List<Integer> result = new ArrayList<>();
        while (rows > 0 && cols > 0) {
            for (int i = 0; i < cols; i++) {
                col += direction;
                result.add(matrix[row][col]);
            }
            rows--;
            for (int i = 0; i < rows; i++) {
```

Saved
Ln 1, Col 1

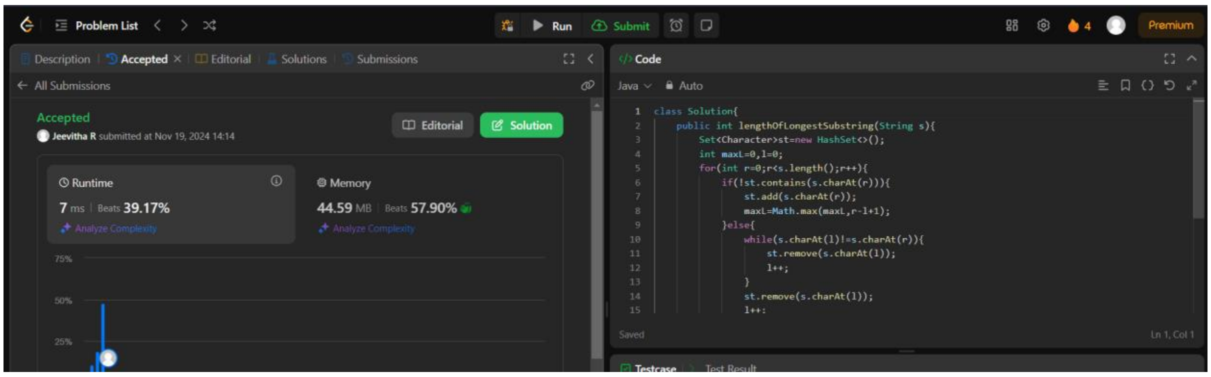
matrix =

Time complexity: O(RXC)
Space complexity: O(RXC)

3. Longest substring without repeating characters

```
class Solution{
    int longestUniqueSubsttr(String S){
        Set<Character>st= new HashSet<>();
        int maxL= 0,l= 0;
        for(int r= 0; r<S.length(); r++){
            if(!st.contains(S.charAt(r))){
                st.add(S.charAt(r));
                maxL=Math.max(maxL, r-l+1);
            }else{
                while(S.charAt(l)!= S.charAt(r)){
                    st.remove(S.charAt(l));
                    l++;
                }
                st.remove(S.charAt(l));
                l++;
                st.add(S.charAt(r));
            }
        }
        return maxL;
    }
}
```

Output:



Accepted
Jeevitha R submitted at Nov 19, 2024 14:14

Runtime
7 ms | Beats 39.17%

Memory
44.59 MB | Beats 57.90%

```
class Solution{
    public int lengthOfLongestSubstring(String s){
        Set<Character>st=new HashSet<>();
        int maxL=0,l=0;
        for(int r=0;r<s.length();r++){
            if(!st.contains(s.charAt(r))){
                st.add(s.charAt(r));
                maxL=Math.max(maxL,r-l+1);
            }else{
                while(s.charAt(l)!=s.charAt(r)){
                    st.remove(s.charAt(l));
                    l++;
                }
                st.remove(s.charAt(l));
                l++:
```

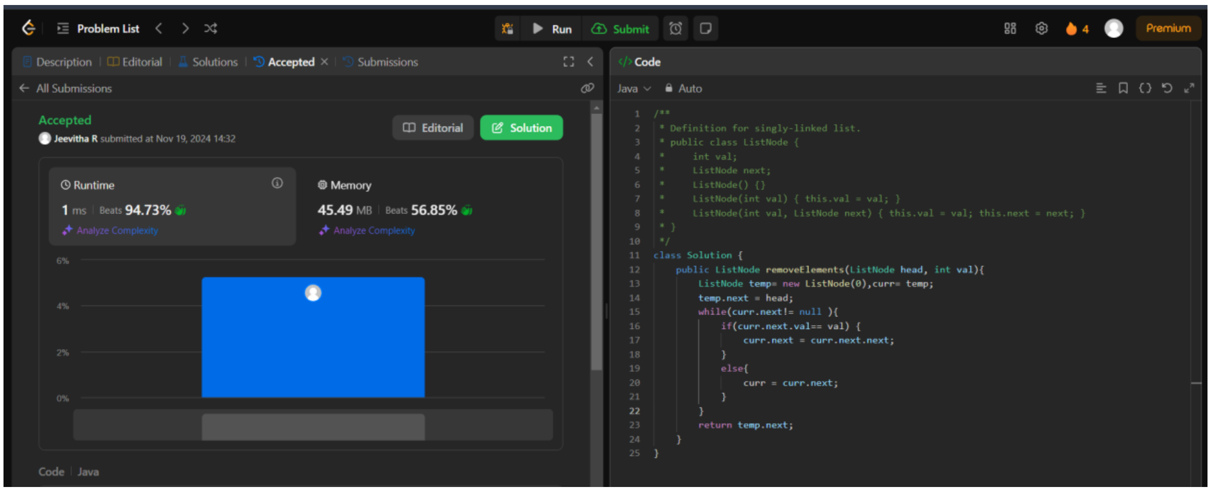
Saved

Time complexity: O(n)
Space complexity: O(n)

4. Remove Linked List Elements

```
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode() {}
 *     ListNode(int val) { this.val = val; }
 *     ListNode(int val, ListNode next) { this.val = val; this.next = next; }
 * }
 */
class Solution {
    public ListNode removeElements(ListNode head, int val){
        ListNode temp= new ListNode(0),curr= temp;
        temp.next = head;
        while(curr.next!= null ){
            if(curr.next.val== val) {
                curr.next = curr.next.next;
            }
            else{
                curr = curr.next;
            }
        }
        return temp.next;
    }
}
```

Output:



Accepted
Jeevitha R submitted at Nov 19, 2024 14:32

Runtime: 1 ms | Beats 94.73%

Memory: 45.49 MB | Beats 56.85%

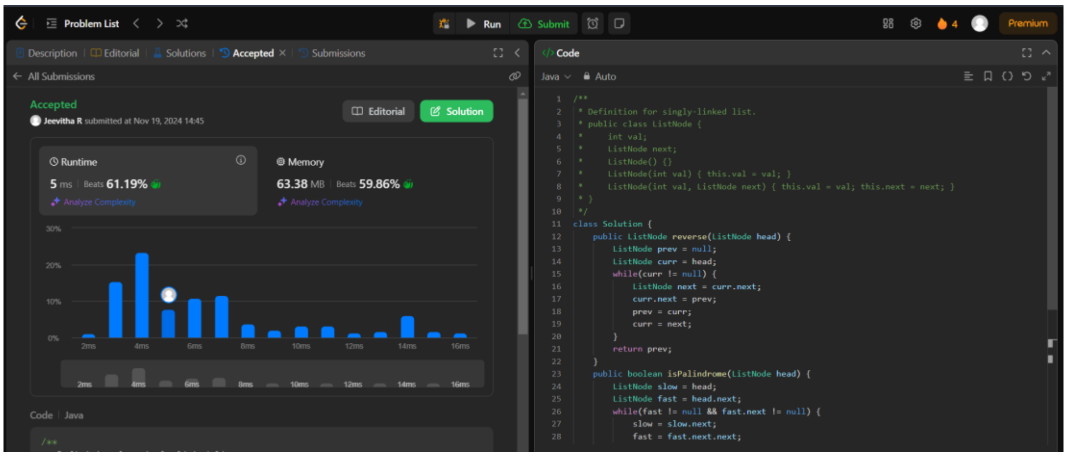
```java
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode() {}
 *     ListNode(int val) { this.val = val; }
 *     ListNode(int val, ListNode next) { this.val = val; this.next = next; }
 * }
 */
class Solution {
    public ListNode removeElements(ListNode head, int val){
        ListNode temp= new ListNode(0),curr= temp;
        temp.next = head;
        while(curr.next!= null ){
            if(curr.next.val== val) {
                curr.next = curr.next.next;
            }
            else{
                curr = curr.next;
            }
        }
        return temp.next;
    }
}
```

Time complexity: O(n)
Space complexity: O(1)

5. Palindrome Linked List

```
class Solution {
    public ListNode reverse(ListNode head) {
        ListNode prev = null;
        ListNode curr = head;
        while(curr != null) {
            ListNode next = curr.next;
            curr.next = prev;
            prev = curr;
            curr = next;
        }
        return prev;
    }
    public boolean isPalindrome(ListNode head) {
        ListNode slow = head;
        ListNode fast = head.next;
        while(fast != null && fast.next != null) {
            slow = slow.next;
            fast = fast.next.next;
        }
        ListNode rev = reverse(slow.next); // reverse second list
        slow.next = null;
        while(rev != null) {
            if(head.val != rev.val) {
                return false;
            }
            head = head.next;
            rev = rev.next;
        }
        return true;
    }
}
```

Output:



Accepted

Jeevitha R submitted at Nov 19, 2024 14:45

Runtime

5 ms | Beats 61.19%

Memory

63.38 MB | Beats 59.86%

Code | Java

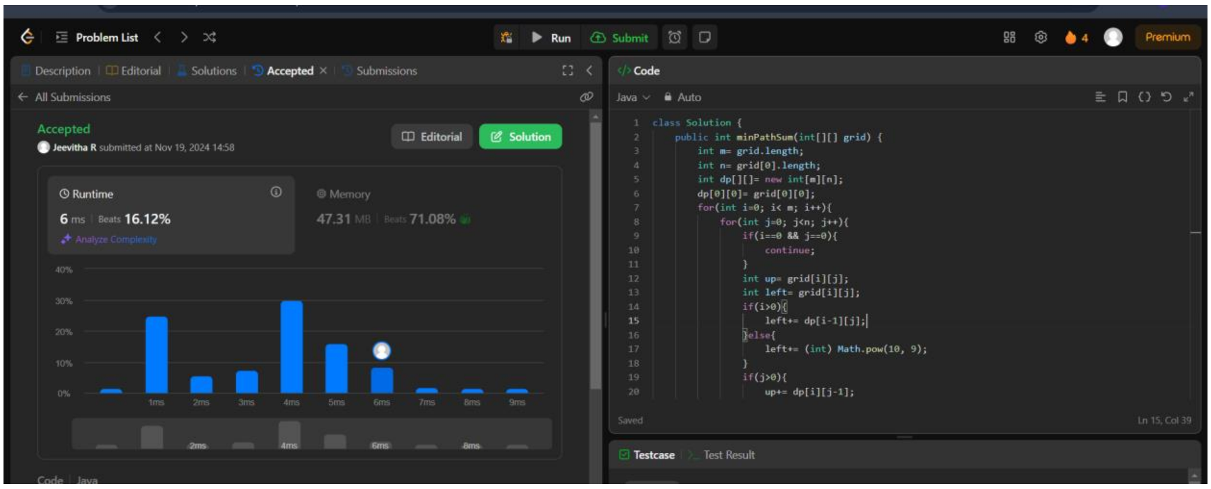
```java
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode() {}
 *     ListNode(int val) { this.val = val; }
 *     ListNode(int val, ListNode next) { this.val = val; this.next = next; }
 * }
 */
class Solution {
    public ListNode reverse(ListNode head) {
        ListNode prev = null;
        ListNode curr = head;
        while(curr != null) {
            ListNode next = curr.next;
            curr.next = prev;
            prev = curr;
            curr = next;
        }
        return prev;
    }
    public boolean isPalindrome(ListNode head) {
        ListNode slow = head;
        ListNode fast = head.next;
        while(fast != null && fast.next != null) {
            slow = slow.next;
            fast = fast.next.next;
```

Time complexity: O(n)
Space complexity: O(1)

6. Minimum path sum

```java
  class Solution {
  public int minPathSum(int[][] grid) {
      int m= grid.length;
      int n= grid[0].length;
      int dp[][]= new int[m][n];
      dp[0][0]= grid[0][0];
      for(int i=0; i< m; i++){
          for(int j=0; j<n; j++){
              if(i==0 && j==0){
                  continue;
              }
              int up= grid[i][j];
              int left= grid[i][j];
              if(i>0){
                  left+= dp[i-1][j];
              }else{
                  left+= (int) Math.pow(10, 9);
              }
              if(j>0){
                  up+= dp[i][j-1];
              }else{
                  up+= (int) Math.pow(10, 9);
              }
              dp[i][j]= Math.min(up , left);
          }
      }
      return dp[m-1][n-1];
    }
}
```

Output:



Accepted
Jeevitha R submitted at Nov 19, 2024 14:58

Runtime
6 ms | Beats 16.12%

Memory
47.31 MB | Beats 71.08%

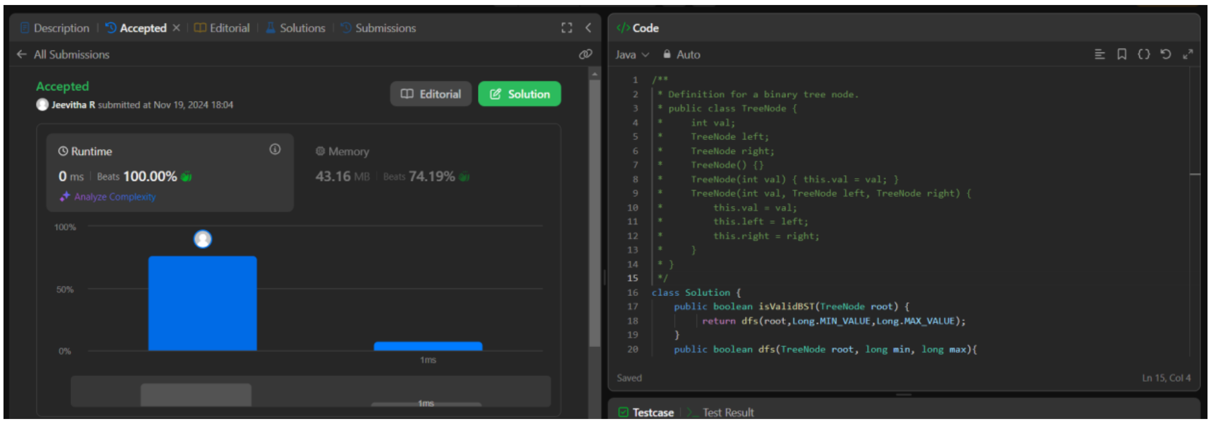
```java
class Solution {
    public int minPathSum(int[][] grid) {
        int m= grid.length;
        int n= grid[0].length;
        int dp[][]= new int[m][n];
        dp[0][0]= grid[0][0];
        for(int i=0; i< m; i++){
            for(int j=0; j<n; j++){
                if(i==0 && j==0){
                    continue;
                }
                int up= grid[i][j];
                int left= grid[i][j];
                if(i>0){
                    left+= dp[i-1][j];
                }else{
                    left+= (int) Math.pow(10, 9);
                }
                if(j>0){
                    up+= dp[i][j-1];
```

Time complexity: O(mxn)
Space complexity: O(mxn)

7. Validate binary search tree

```java
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode() {}
 *     TreeNode(int val) { this.val = val; }
 *     TreeNode(int val, TreeNode left, TreeNode right) {
 *         this.val = val;
 *         this.left = left;
 *         this.right = right;
 *     }
 * }
 */
class Solution {
    public boolean isValidBST(TreeNode root) {
        return dfs(root,Long.MIN_VALUE,Long.MAX_VALUE);
    }
    public boolean dfs(TreeNode root, long min, long max){
        if(root==null){
            return true;
        }
        if(min>=root.val || root.val>=max){
            return false;
        }
        boolean isleftvalid=dfs(root.left,min,root.val);
        boolean isrightvalid=dfs(root.right,root.val,max);
        if(isleftvalid && isrightvalid){
            return true;
        }
        return false;
    }
}
```

Output:



Accepted
Jeevitha R submitted at Nov 19, 2024 18:04

Runtime
0 ms | Beats 100.00%

Memory
43.16 MB | Beats 74.19%

Java Auto

```java
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode() {}
 *     TreeNode(int val) { this.val = val; }
 *     TreeNode(int val, TreeNode left, TreeNode right) {
 *         this.val = val;
 *         this.left = left;
 *         this.right = right;
 *     }
 * }
 */
class Solution {
    public boolean isValidBST(TreeNode root) {
        return dfs(root,Long.MIN_VALUE,Long.MAX_VALUE);
    }
    public boolean dfs(TreeNode root, long min, long max){
```

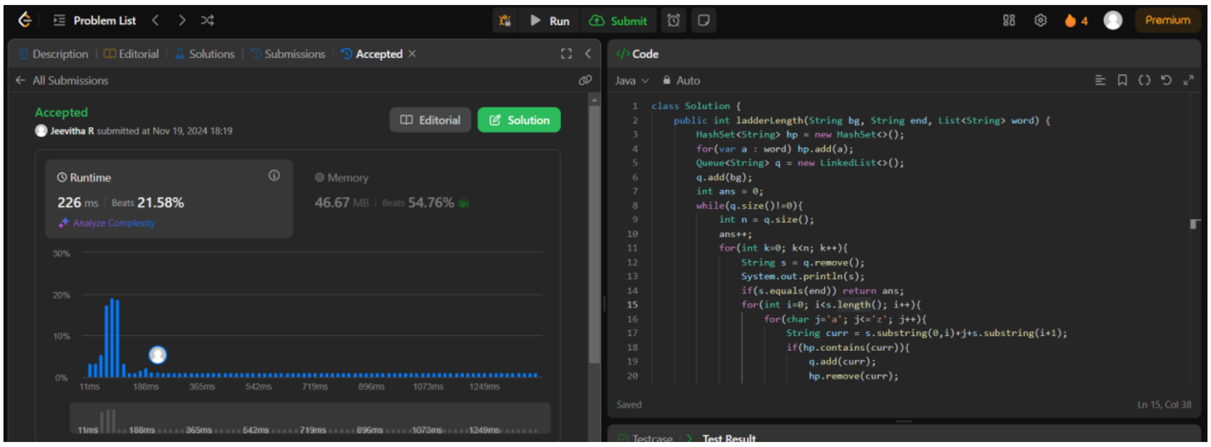
Saved   Ln 15, Col 4

Time complexity: O(n)
Space complexity: O(h)

8. Word Ladder

```java
class Solution {
    public int ladderLength(String bg, String end, List<String> word) {
        HashSet<String> hp = new HashSet<>();
        for(var a : word) hp.add(a);
        Queue<String> q = new LinkedList<>();
        q.add(bg);
        int ans = 0;
        while(q.size()!=0){
            int n = q.size();
            ans++;
            for(int k=0; k<n; k++){
                String s = q.remove();
                System.out.println(s);
                if(s.equals(end)) return ans;
                for(int i=0; i<s.length(); i++){
                    for(char j='a'; j<='z'; j++){
                        String curr = s.substring(0,i)+j+s.substring(i+1);
                        if(hp.contains(curr)){
                            q.add(curr);
                            hp.remove(curr);
                        }
                    }
                }
            }
        }
        return 0;
    }
}
```

Output



Accepted
Jeevitha R submitted at Nov 19, 2024 18:19

Runtime
226 ms | Beats 21.58%

Memory
46.67 MB | Beats 54.76%

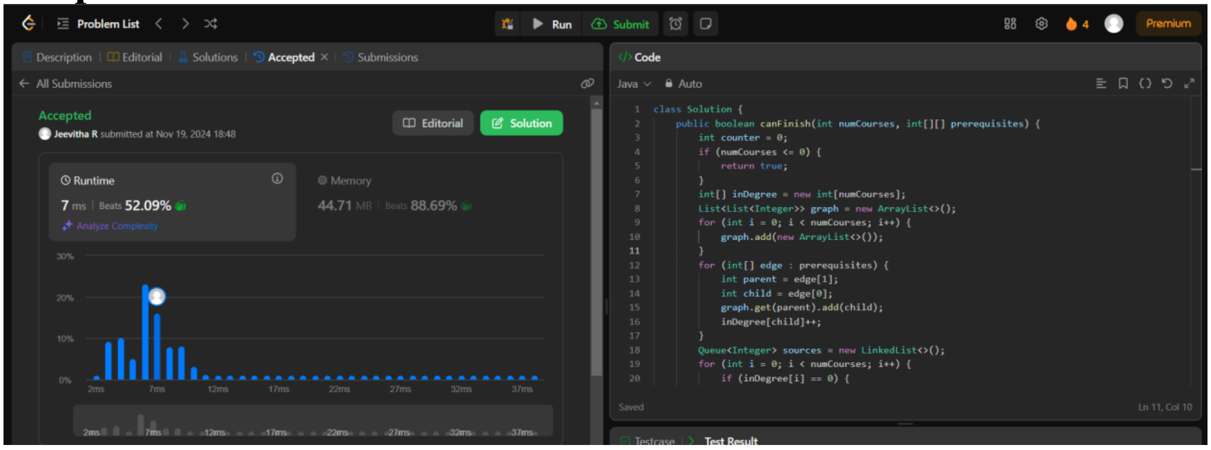
```java
class Solution {
    public int ladderLength(String bg, String end, List<String> word) {
        HashSet<String> hp = new HashSet<>();
        for(var a : word) hp.add(a);
        Queue<String> q = new LinkedList<>();
        q.add(bg);
        int ans = 0;
        while(q.size()!=0){
            int n = q.size();
            ans++;
            for(int k=0; k<n; k++){
                String s = q.remove();
                System.out.println(s);
                if(s.equals(end)) return ans;
                for(int i=0; i<s.length(); i++){
                    for(char j='a'; j<='z'; j++){
                        String curr = s.substring(0,i)+j+s.substring(i+1);
                        if(hp.contains(curr)){
                            q.add(curr);
                            hp.remove(curr);
```

Time complexity: O(M*M*N)
Space complexity: O(M*N)

10. Course Schedule

```java
class Solution {
    public boolean canFinish(int numCourses, int[][] prerequisites) {
        int counter = 0;
        if (numCourses <= 0) {
            return true;
        }
        int[] inDegree = new int[numCourses];
        List<List<Integer>> graph = new ArrayList<>();
        for (int i = 0; i < numCourses; i++) {
            graph.add(new ArrayList<>());
        }
        for (int[] edge : prerequisites) {
            int parent = edge[1];
            int child = edge[0];
            graph.get(parent).add(child);
            inDegree[child]++;
        }
        Queue<Integer> sources = new LinkedList<>();
        for (int i = 0; i < numCourses; i++) {
            if (inDegree[i] == 0) {
                sources.offer(i);
            }
        }
        while (!sources.isEmpty()) {
            int course = sources.poll();
            counter++;
            for (int child : graph.get(course)) {
                inDegree[child]--;
                if (inDegree[child] == 0) {
                    sources.offer(child);
                }
            }
        }
        return counter == numCourses;
    }
}
```

Output



Accepted
Jeevitha R submitted at Nov 19, 2024 18:48

Runtime
7 ms | Beats 52.09%

Memory
44.71 MB | Beats 88.69%

```java
class Solution {
    public boolean canFinish(int numCourses, int[][] prerequisites) {
        int counter = 0;
        if (numCourses <= 0) {
            return true;
        }
        int[] inDegree = new int[numCourses];
        List<List<Integer>> graph = new ArrayList<>();
        for (int i = 0; i < numCourses; i++) {
            graph.add(new ArrayList<>());
        }
        for (int[] edge : prerequisites) {
            int parent = edge[1];
            int child = edge[0];
            graph.get(parent).add(child);
            inDegree[child]++;
        }
        Queue<Integer> sources = new LinkedList<>();
        for (int i = 0; i < numCourses; i++) {
            if (inDegree[i] == 0) {
```

Time complexity: O(V+E)
Space complexity: O(V+E)

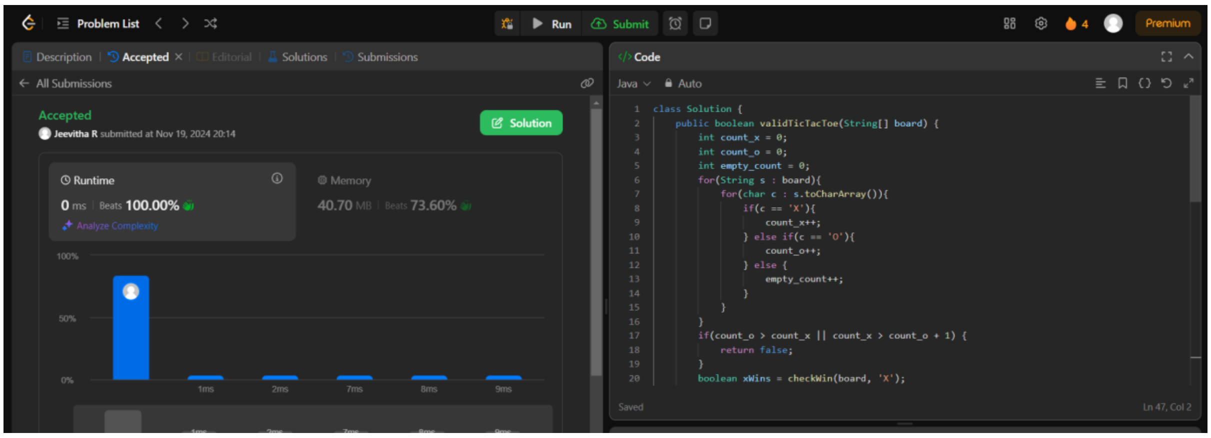
11. Design tic tac toe

```java
class Solution {
    public boolean validTicTacToe(String[] board) {
        if(board.length == 0){
            return true;
        }
        int count_x = 0;
        int count_o = 0;
        int empty_count = 0;
        for(String s : board){
            for(char c : s.toCharArray()){
                if(c == 'X'){
                    count_x++;
                }else if(c == 'O'){
                    count_o++;
                }else{
                    empty_count++;
                }
            }
        }
        if(count_x!=0){
            if(count_x==count_o || count_o+1==count_x || count_o==count_x+1){
                return true;
            }
            boolean X= valid(board, 'X');
            boolean O= valid(board, 'O');
            if(X && O)

        }
       return false;
    }
    private boolean valid(String[] board, char p){
            for(int i = 0; i < 3; i++) {
```

```
            if((board[i].charAt(0) == p && board[i].charAt(1) == p && board[i].charAt(2) == p) ||
               (board[0].charAt(i) == p && board[1].charAt(i) == p && board[2].charAt(i) == p)) {
                return true;
            }
        }
            if((board[0].charAt(0) == p && board[1].charAt(1) == p && board[2].charAt(2) == p) ||
            (board[0].charAt(2) == p && board[1].charAt(1) == p && board[2].charAt(0) == p)) {
                return true;
            }
            return false;
    }
}
```

Output:



Accepted
Jeevitha R submitted at Nov 19, 2024 20:14

Runtime
0 ms | Beats 100.00%

Memory
40.70 MB | Beats 73.60%

```
class Solution {
    public boolean validTicTacToe(String[] board) {
        int count_x = 0;
        int count_o = 0;
        int empty_count = 0;
        for(String s : board){
            for(char c : s.toCharArray()){
                if(c == 'X'){
                    count_x++;
                } else if(c == 'O'){
                    count_o++;
                } else {
                    empty_count++;
                }
            }
        }
        if(count_o > count_x || count_x > count_o + 1) {
            return false;
        }
        boolean xWins = checkWin(board, 'X');
```

Saved
Ln 47, Col 2

Time complexity: O(1)
Space complexity: O(1)