DSA PRACTICE PROBLEMS – SET 8 [LEETCODE QUESTIONS]

Madhulekha R – AIML - [20/11/2024]

1. Question: 3sum closest

Code:

```
class Solution {
    public int threeSumClosest(int[] nums, int target) {
        Arrays.sort(nums);
        int closest = nums[0] + nums[1] + nums[2];

        for (int i = 0; i < nums.length - 2; i++) {
            int left = i + 1, right = nums.length - 1;
            while (left < right) {
                int currSum = nums[i] + nums[left] + nums[right];

                if (currSum == target) {
                    return target;
                }
                if (Math.abs(currSum - target) < Math.abs(closest - target)) {
                    closest = currSum;
                }
                if (currSum < target) {
                    left++;
                } else {
                    right--;
                }
            }
        }
        return closest;
    }
}
```

Output:

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

nums =
[-1,2,1,-4]

target =
1

Output

2

Expected

2

Time complexity: O(n^2)
Space complexity: O(1)

2. **Question: Jump Game II**

Code:

```java
class Solution {
    public int jump(int[] nums) {
        int jumps = 0, currentEnd = 0, furthest = 0, n = nums.length;
        for (int i = 0; i < n - 1; i++) {
            furthest = Math.max(furthest, i + nums[i]);
            if (i == currentEnd) {
                jumps++;
                currentEnd = furthest;
            }
        }
        return jumps;
    }
}
```

Output:

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

nums =

[2,3,1,1,4]

Output

2

Expected

2

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

3. **Question: Group anagrams**

Code:

```java
class Solution {
    public List<List<String>> groupAnagrams(String[] strs) {
        Map<String, List<String>> map = new HashMap<>();

        for (String word : strs) {
            char[] chars = word.toCharArray();
            Arrays.sort(chars);
            String sortedWord = new String(chars);

            if (!map.containsKey(sortedWord)) {
                map.put(sortedWord, new ArrayList<>());
            }

            map.get(sortedWord).add(word);
        }

        return new ArrayList<>(map.values());
    }
}
```

Output:

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

strs =
["eat","tea","tan","ate","nat","bat"]

Output

[["eat","tea","ate"],["bat"],["tan","nat"]]

Expected

[["bat"],["nat","tan"],["ate","eat","tea"]]

Time complexity: O (n * k log k)
Space complexity: O (n*k)

4. **Question: Decode ways**

Code:

```java
class Solution {
    public int numDecodings(String s) {
        int n=s.length() ;
        int[] dp=new int[n+1] ;
        dp[n]=1 ;
```

```
        for(int i = n-1 ; i >= 0 ; i--)
            if(s.charAt(i)!='0')
            {
                dp[i] = dp[i+1] ;
                if(i < n-1 && (s.charAt(i)=='1' || s.charAt(i)=='2' &&
s.charAt(i+1)<'7'))
                    dp[i]+=dp[i+2];
            }
        return dp[0];
    }
}
```

Output:

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

s =
"12"

Output

2

Expected

2

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

5. **Question: Best time to buy and sell stock II**

Code:

```
class Solution {
    public int maxProfit(int[] prices) {
        int n=prices.length;
        if(prices==null || n<2) return 0;
        int totalProfit=0;
        int minNo=Integer.MAX_VALUE;
        for(int i=0;i<n;i++){
            if(prices[i]<minNo){
                minNo=prices[i];
            }
            else{
                totalProfit+=prices[i]-minNo;
                minNo=prices[i];
            }
        }
        return totalProfit;
    }
}
```

Output:

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

prices =

[7,1,5,3,6,4]

Output

7

Expected

7

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

6. **Question: Number of islands**

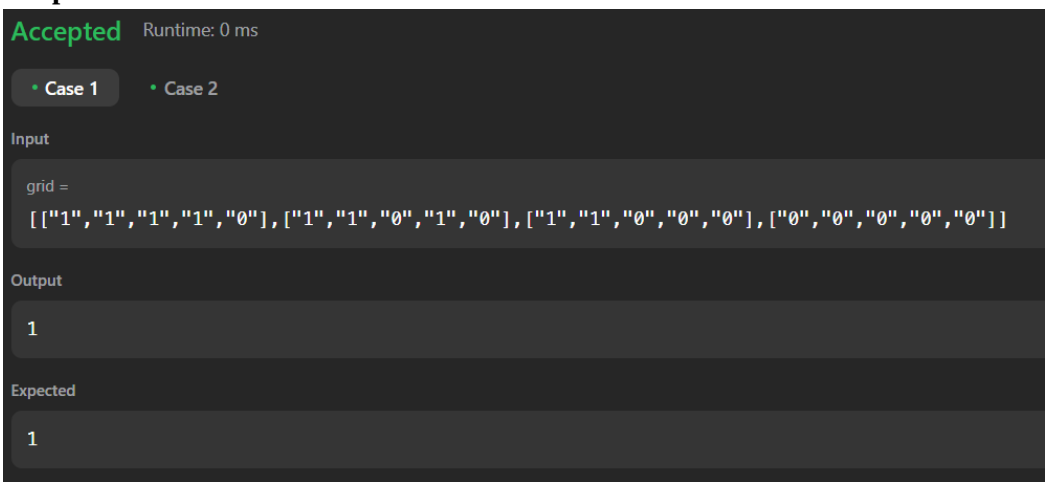
Code:

```
class Solution {
    int row=0;
    int col=0;
    public int numIslands(char[][] grid) {
        row=grid.length;
        col=grid[0].length;
        int count=0;

        for(int i=0;i<row;i++){
            for(int j=0;j<col;j++){
                if(grid[i][j]=='1'){
                    dfs(grid,i,j);
                    count++;}
            } }
        return count;}

    public void dfs(char[][] grid,int i,int j){
        if (i < 0 || i >= row || j < 0 || j >= col || grid[i][j] != '1') {
            return;
        }
        grid[i][j] = '2';
        dfs(grid, i, j - 1);
        dfs(grid, i - 1, j);
        dfs(grid, i, j + 1);
        dfs(grid, i + 1, j);
    }}
```

Output:



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

7. **Question: Quick sort**

Code:

```
import java.util.Arrays;

class Main5{
    public static void main(String[] args) {
        int[] arr = {10, 7, 8, 9, 1, 5};
        int n = arr.length;
        quickSort(arr, 0, n - 1);
        for (int val : arr) {
            System.out.print(val + " ");
        }
    }

    static int partition(int[] arr, int low, int high){
        int pivot = arr[high];
        int i = low - 1;
        for (int j = low; j <= high - 1; j++) {
            if (arr[j] < pivot) {
                i++;
                swap(arr, i, j);
            }
        }
        swap(arr, i + 1, high);
```

```
        return i + 1;
    }

    static void swap(int[] arr, int i, int j) {
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }

    static void quickSort(int[] arr, int low, int high) {
        if (low < high) {
            int pi = partition(arr, low, high);
            quickSort(arr, low, pi - 1);
            quickSort(arr, pi + 1, high);
        }
    }
}
```

Output:

```
C:\Users\HP\Documents>javac Main5.java

C:\Users\HP\Documents>java Main5
1 5 7 8 9 10
```

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

8. **Question: Merge sort**

Code:

```
class Main5{
    static void merge(int arr[], int l, int m, int r){
        int n1 = m - l + 1;
        int n2 = r - m;
        int L[] = new int[n1];
        int R[] = new int[n2];

        for (int i = 0; i < n1; ++i)
            L[i] = arr[l + i];
        for (int j = 0; j < n2; ++j)
            R[j] = arr[m + 1 + j];
```

```
    int i = 0, j = 0;
    int k = l;
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            i++;
        }
        else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }

    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }

    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}

static void sort(int arr[], int l, int r){
    if (l < r){
        int m = l + (r - l) / 2;
        sort(arr, l, m);
        sort(arr, m + 1, r);
        merge(arr, l, m, r);
    }
}

static void printArray(int arr[]){
    int n = arr.length;
    for (int i = 0; i < n; ++i)
        System.out.print(arr[i] + " ");
    System.out.println();
}

public static void main(String args[]){
```

```
        int arr[] = { 12, 11, 13, 5, 6, 7 };
        System.out.println("Given array is");
        printArray(arr);
        sort(arr, 0, arr.length - 1);
        System.out.println("\nSorted array is");
        printArray(arr);
    }
}
```

Output:

```
C:\Users\HP\Documents\SDE DSA Practice Problems\Program execution>javac Main5.java

C:\Users\HP\Documents\SDE DSA Practice Problems\Program execution>java Main5
Given array is
12 11 13 5 6 7

Sorted array is
5 6 7 11 12 13
```

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

9. **Question: Ternary search**

Code:

```
class Main5{
    public static void main(String args[]){
        int l, r, p, key;
        int ar[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
        l = 0;
        r = 9;
        key = 5;

        p = ternarySearch(l, r, key, ar);
        System.out.println("Index of " + key + " is " + p);
        key = 50;
        p = ternarySearch(l, r, key, ar);
        System.out.println("Index of " + key + " is " + p);
    }

    static int ternarySearch(int l, int r, int key, int ar[]){
        if (r >= l){
            int mid1 = l + (r - l) / 3;
            int mid2 = r - (r - l) / 3;
```

```
            if (ar[mid1] == key){
                return mid1;
            }
            if (ar[mid2] == key){
                return mid2;
            }
            if (key < ar[mid1]){
                return ternarySearch(l, mid1 - 1, key, ar);
            }
            else if (key > ar[mid2]){
                return ternarySearch(mid2 + 1, r, key, ar);
            }
            else{
                return ternarySearch(mid1 + 1, mid2 - 1, key, ar);
            }
        }
        return -1;
    }
}
```

Output:

```
C:\Users\HP\Documents\SDE DSA Practice Problems\Program execution>javac Main5.java

C:\Users\HP\Documents\SDE DSA Practice Problems\Program execution>java Main5
Index of 5 is 4
Index of 50 is -1
```

Time complexity: O(2*log3n)
Space complexity: O(log3n)

10. **Question: Interpolation search**

Code:

```
import java.util.*;

class Main5{
  public static int interpolationSearch(int arr[], int lo, int hi, int x){
    int pos;
      if (lo <= hi && x >= arr[lo] && x <= arr[hi]){
            pos = lo + (((hi - lo) / (arr[hi] - arr[lo])) * (x - arr[lo]));

            if (arr[pos] == x){
              return pos;
          }
```

```
            if (arr[pos] < x){
              return interpolationSearch(arr, pos + 1, hi,x);
            }
            if (arr[pos] > x){
                    return interpolationSearch(arr, lo, pos - 1, x);
          }
           }
           return -1;
  }

  public static void main(String[] args){
          int arr[] = { 10, 12, 13, 16, 18, 19, 20, 21, 22, 23, 24, 33, 35, 42, 47 };
      int n = arr.length;
          int x = 18;
          int index = interpolationSearch(arr, 0, n - 1, x);

          if (index != -1){
            System.out.println("Element found at index " + index);
      }
          else{
            System.out.println("Element not found.");
      }
  }
}
```

Output:

```
C:\Users\HP\Documents\SDE DSA Practice Problems\Program execution>javac Main5.java

C:\Users\HP\Documents\SDE DSA Practice Problems\Program execution>java Main5
Element found at index 4
```

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