DSA PRACTICE-8-20/11/24

1.3SUM CLOSEST

```
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 l=i+1;
            int r=nums.length-1;
            while(l<r){
                int cursum=nums[i]+nums[l]+nums[r];
                if(cursum==target){
                    return target;
                }
                if(Math.abs(cursum-target)<Math.abs(closest-target)){
                    closest=cursum;
                }
                if(cursum<target){
                    l++;
                }
                else{
                    r--;
                }

            }
        }
        return closest;
    }
}
```

2.Jump Game II

```
class Solution {
    public int jump(int[] nums) {
        int reach = 0, count = 0, last = 0;
        for (int i = 0; i < nums.length - 1; i++) {
            reach = Math.max(reach, i + nums[i]);
            if (i == last) {
                last = reach;
                count++;
            }
        }
        return count;
    }
}
```

3. **GROUP ANAGRAMS**

```
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());
    }
}
```

4. Decode Ways

```
class Solution {
    public int numDecodings(String s) {
        if (s == null || s.length() == 0 || s.charAt(0) == '0') {
            return 0;
        }
        int n = s.length();
        int[] dp = new int[n + 1];
        dp[0] = 1;
        dp[1] = 1;
        for (int i = 2; i <= n; i++) {
            int oneDigit = Integer.parseInt(s.substring(i - 1, i));
            int twoDigits = Integer.parseInt(s.substring(i - 2, i));
            if (oneDigit >= 1 && oneDigit <= 9) {
                dp[i] += dp[i - 1];
            }
            if (twoDigits >= 10 && twoDigits <= 26) {
                dp[i] += dp[i - 2];
            }
        }
        return dp[n];
    }
}
```

5. **BEST TIME TO BUY AND SELL STOCK II**

```
class Solution {
    public int maxProfit(int[] prices) {
        int profit = 0;

        for (int i = 1; i < prices.length; i++) {
            if (prices[i] > prices[i - 1]) {
                profit += prices[i] - prices[i - 1];
            }
        }

        return profit;
    }
}
```

6. Number of Islands

```
class Solution {
    public void dfs(int i ,int j,char[][] grid){
        if(i<0 || j<0 || i>=grid.length || j>=grid[0].length || grid[i][j]=='0'){
            return ;
        }

        if( grid[i][j] =='x') {
                return ;
        }

        grid[i][j]='x';
        dfs(i-1,j,grid);
        dfs(i+1,j,grid);
        dfs(i,j-1,grid);
        dfs(i,j+1,grid);
```

```
    }
    public int numIslands(char[][] grid) {

        int noOfIslands = 0;
        int m = grid.length;
        int n = grid[0].length;
        for(int i = 0;i<m;i++){
            for(int j =0;j<n;j++){
                if(grid[i][j]=='1'){
                    dfs(i,j,grid);
                    noOfIslands++;
                }
            }
        }

        return noOfIslands;
    }
}
```