

DSA CODING QUESTIONS PRACTICE – 8

1. 3 Closest Sum

The screenshot shows a coding editor interface for a problem titled "3 Closest Sum". The solution is marked as "Accepted" and was submitted at Nov. The runtime is 12 ms, beating 97.44% of other solutions. The memory usage is 42.98 MB, beating 77.04% of other solutions. The code is written in Java and implements a three-pointer approach to find the closest sum of three numbers to a target.

```
Java  
1 public int threeSumClosest(int[] nums, int target) {  
2     Arrays.sort(nums);  
3     int ans = 0;  
4     int dif = Integer.MAX_VALUE;  
5     for(int i=0; i<nums.length-2; i++){  
6         int l = i+1;  
7         int r = nums.length-1;  
8         while(l<r){  
9             int currSum = nums[i] + nums[l] + nums[r];  
10            if(Math.abs(currSum-target)<dif){  
11                dif = Math.abs(currSum-target);  
12                ans = currSum;  
13            }  
14            if(currSum<target){  
15                l++;  
16            }else if(currSum > target){  
17                r--;  
18            }else{  
19                return currSum;  
20            }  
21        }  
22    }  
23    return ans;  
24 }
```

2. Group Anagrams

The screenshot shows a coding editor interface for a problem titled "Group Anagrams". The solution is marked as "Accepted" and was submitted at Nov 20, 2024 17. The runtime is 7 ms, beating 64.12% of other solutions. The memory usage is 47.58 MB, beating 82.02% of other solutions. The code is written in Java and uses a HashMap to group anagrams by their sorted character keys.

```
Java  
1 class Solution {  
2     public List<List<String>> groupAnagrams(String[] strs) {  
3         Map<String, List<String>> ans = new HashMap<>();  
4         for (String s : strs) {  
5             char[] chars = s.toCharArray();  
6             Arrays.sort(chars);  
7             String key = new String(chars);  
8             if (!ans.containsKey(key)) {  
9                 ans.put(key, new ArrayList<>());  
10            }  
11            ans.get(key).add(s);  
12        }  
13        return new ArrayList<>(ans.values());  
14    }  
15 }
```

3. Interpolation Search

The screenshot shows a LeetCode submission for the 'Interpolation Search' problem. The submission is 'Accepted' and was made by 'Jency R' on Nov 20, 2024, at 22:18. The performance metrics are: Runtime 0 ms (Beats 100.00%), Memory 45.24 MB (Beats 94.57%). The code is written in Java and implements a binary search algorithm. The code is as follows:

```
1 class Solution {
2     public int search(int[] nums, int target) {
3         int n=nums.length;
4         int l=0;
5         int h=n-1;
6         while(l<=h)
7         {
8             int mid=l+(h-1)/2;
9             if(nums[mid]==target)
10            {
11                return mid;
12            }
13            else if(target>nums[mid])
14            {
15                l=mid+1;
16            }
17            else
18            {
19                h=mid-1;
20            }
21        }
22    }
23 }
```

4. Next permutation

The screenshot shows a LeetCode submission for the 'Next Permutation' problem. The submission is 'Accepted' and was made by 'Jency R' on Nov 20, 2024, at 22:40. The performance metrics are: Runtime 0 ms (Beats 100.00%), Memory 42.82 MB (Beats 75.05%). The code is written in Java and implements the next permutation algorithm. The code is as follows:

```
1 class Solution {
2     public void nextPermutation(int[] nums) {
3         int i = nums.length - 2;
4         while (i>=0 && nums[i] >= nums[i + 1]){
5             i--;
6         }
7         if (i != -1) {
8             int j = nums.length-1;
9             while (j>=0 && nums[i] >= nums[j]) {
10                j--;
11            }
12            swap(nums, i, j);
13        }
14        int start = i + 1;
15        int end = nums.length - 1;
16        while (start < end) {
17            swap(nums, start, end);
18            start++;
19            end--;
20        }
21    }
22 }
```