TECHNICAL TRAINING DSA - CODING PRACTICE PROBLEMS

Name: Sachin A
Dept: CSBS
Date: 20-11-2024

3Sum Closest

CODE:

```
class Solution {
public:
    int threeSumClosest(vector<int>& nums, int target) {
        sort(nums.begin(),nums.end());
        int closest = nums[0] + nums[1] + nums[2];
        for (int i = 0; i < nums.size() - 2; i++) {
            int left = i + 1, right = nums.size() - 1;

            while (left < right) {
                int currSum = nums[i] + nums[left] + nums[right];

                if (currSum == target) {
                    return target;
                }

                if (abs(currSum - target) < abs(closest - target)) {
                    closest = currSum;
                }

                if (currSum < target) {
                    left++;
                } else {
                    right--;
                }
            }
        }

        return closest;
    }
};
```

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

Group Anagrams

CODE:

```
class Solution:
    def groupAnagrams(self, strs):
        anagram_map = defaultdict(list)

        for word in strs:
            sorted_word = ''.join(sorted(word))
            anagram_map[sorted_word].append(word)

        return list(anagram_map.values())
```

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

Best time to buy and sell stocks II

CODE

```
int maxProfit(vector& prices) {
int prev=prices[0];
int n=prices.size();
int ans=0;
for(int i=1;i0){
ans+=p;
prev=prices[i];
}
 else
{
prev=min(prices[i],prev);
 }
}
 return ans;
```

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

No Of Islands

CODE:

```
class Solution:
    def numIslands(self, grid: List[List[str]]) -> int:
        if not grid:
            return 0

        def dfs(i, j):
            if i < 0 or i >= len(grid) or j < 0 or j >= len(grid[0]) or grid[i][j] != '1':
                return
            grid[i][j] = '0'  # mark as visited
            dfs(i+1, j)
            dfs(i-1, j)
            dfs(i, j+1)
            dfs(i, j-1)

        num_islands = 0
        for i in range(len(grid)):
            for j in range(len(grid[0])):
                if grid[i][j] == '1':
                    num_islands += 1
                    dfs(i, j)

        return num_islands
```

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

Quick Sort

CODE:

```
int part(vector<int> &arr,int left, int right){
    int pivot=arr[right];
    int i=left-1;
    for(int j=left;j<right;j++){
    if(arr[j]<pivot){
    i++;
    swap(arr[i],arr[j]);
    }
    }
    swap(arr[i+1],arr[right]);
    return i+1;
    }
    void quicksort(vector<int> &arr, int left, int right){
```

```
if(left<right){
int pivot=part(arr,left,right);
quicksort(arr,left,pivot-1);
quicksort(arr,pivot+1,right);
}
}
int main(){
int n;
cout<<"Enter length: ";
cin>>n;
vector<int> arr(n);
for(int i=0;i<n;i++){
cin>>arr[i];
}
quicksort(arr,0,n-1);
for(int x:arr){
cout<<x<<" ";
}
return 0;
}
```

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

Merge Sort

CODE:

```
void mergee(vector<int>& arr,int low,int mid, int high){
vector<int> temp;
int ptr1=low;
int ptr2=mid+1;
while(ptr1<=mid && ptr2<=high){
if(arr[ptr1]<=arr[ptr2]){
temp.push_back(arr[ptr1]);
ptr1++;
}
else{
temp.push_back(arr[ptr2]);
ptr2++;
}
}
while(ptr1<=mid){
temp.push_back(arr[ptr1]);
ptr1++;
```

```
}
while(ptr2<=high){
temp.push_back(arr[ptr2]);
ptr2++;
}
for(int i=low;i<=high;i++){
arr[i]=temp[i-low];
}
}
void mergesort(vector<int>& arr,int low,int high){
if(low>=high) return;
int mid=(low+high)/2;
mergesort(arr,low,mid);
mergesort(arr,mid+1,high);
mergee(arr,low,mid,high);
}
vector<int> sortArray(vector<int>& nums) {
mergesort(nums,0,nums.size()-1);
return nums;
}
```

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

Ternary Search

CODE:

```
int search(vector<int>& nums, int target) {
int left=0;
int right=nums.size()-1;
while(left<=right){
int mid1=left+(right-left)/3;
int mid2=right-(right-left)/3;
if(nums[mid1]==target) return mid1;
else if(nums[mid2]==target) return mid2;
else if(target<nums[mid1]) right=mid1-1;
else if(target>nums[mid2]) left=mid2+1;
else{
```

```
left=mid1+1;
right=mid2-1;
}
}
return -1;
}
```

Time Complexity: $O(\log_3 n)$
Space Complexity: $O(n)$