TECHNICAL TRAINING DSA - CODING PRACTICE PROBLEMS

Name: Rishi Kumar S
Dept: CSBS
Date: 20-11-2024

Question 1: 3Sum Closest

Given an integer array nums of length n and an integer target, find three integers in nums such that the sum is closest to target.
Return the sum of the three integers.
You may assume that each input would have exactly one solution.

CODE:

```
int threeSumClosest(vector<int>& nums, int target) {
sort(nums.begin(), nums.end());
int minn = INT_MAX;
int ans = 0;

for (int i = 0; i < nums.size(); i++) {
if (i > 0 && nums[i] == nums[i - 1]) {
continue;
}
int j = i + 1;
int k = nums.size() - 1;
while (j < k) {
int total = nums[i] + nums[j] + nums[k];
if (abs(total - target) < minn) {
minn = abs(total - target);
ans = total;
}
if (total > target) {
k--;
} else if (total < target) {
j++;
} else {
j++;
while (nums[j] == nums[j - 1] && j < k) {
j++;
}
}
}
}
return ans;
}
```

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

Question 2: Group Anagrams

Given an array of strings strs, group the
anagrams
together. You can return the answer in any order.

CODE:

```
vector<vector<string>> groupAnagrams(vector<string>& strs) {
unordered_map<string,vector<string>> map;
vector<vector<string>> ans;
for(string s: strs){
string rem=s;
sort(s.begin(),s.end());
map[s].push_back(rem);
}

for(auto [a,b]:map){
ans.push_back(b);
}
return ans;

}
```

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

Question 3: Best Time to Buy and Sell Stock II

You are given an integer array prices where prices[i] is the price of a given stock on the ith day.
On each day, you may decide to buy and/or sell the stock. You can only hold at most one share of the stock at any time. However, you can buy it then immediately sell it on the same day.
Find and return the maximum profit you can achieve.

CODE:

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

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

Question 4: Number of Islands

Given an m x n 2D binary grid grid which represents a map of '1's (land) and '0's (water), return the number of islands.
An island is surrounded by water and is formed by connecting adjacent lands horizontally or vertically. You may assume all four edges of the grid are all surrounded by water.

CODE:

```
#include <vector>
#include <queue>
#include <unordered_set>
#include <utility>

using namespace std;

class Solution {
public:
    int numIslands(vector<vector<char>>& grid) {
        if (grid.empty()) return 0;

        int rows = grid.size();
        int cols = grid[0].size();
        vector<vector<bool>> visited(rows, vector<bool>(cols, false));
        int islands = 0;

        auto bfs = [&](int r, int c) {
            queue<pair<int, int>> q;
            visited[r][c] = true;
            q.push({r, c});

            while (!q.empty()) {
                auto [row, col] = q.front();
                q.pop();

                vector<pair<int, int>> directions = {{1, 0}, {-1, 0}, {0, 1}, {0,
-1}};
                for (auto [dr, dc] : directions) {
                    int nr = row + dr, nc = col + dc;
                    if (nr >= 0 && nr < rows && nc >= 0 && nc < cols &&
                        grid[nr][nc] == '1' && !visited[nr][nc]) {
                        q.push({nr, nc});
                        visited[nr][nc] = true;
                    }
                }
            }
        };

        for (int i = 0; i < rows; ++i) {
            for (int j = 0; j < cols; ++j) {
                if (grid[i][j] == '1' && !visited[i][j]) {
                    bfs(i, j);
                    ++islands;
                }
            }
        }

        return islands;
    }
};
```
Time Complexity: O(m*n)
Space Complexity: O(m*n)

QUESTION 5:
Quick Sort

CODE:

```
#include <iostream>
#include <vector>
using namespace std;

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)

Question 6:
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)

Question 7:
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)