Dsa day 7

1. minimum path sum

class Solution {

private:

vector<vector<int>>dp;

int h(vector<vector<int>>&grid, int i, int j){

if(i == grid.size() - 1 && j == grid[0].size() - 1)

return grid[i][j];

if(i >= grid.size() || j >= grid[0].size())

return INT\_MAX;

if(dp[i][j] != -1)

return dp[i][j];

int right = h(grid, i, j + 1);

int down = h(grid, i + 1, j);

return dp[i][j] = grid[i][j] + min(right, down);

}

public:

int minPathSum(vector<vector<int>>& grid) {

dp = vector<vector<int>>(grid.size(), vector<int>(grid[0].size(), -1));

return h(grid, 0, 0);

}

};

2) is subsequence

**#include <string>**

**using namespace std;**

**class Solution {**

**public:**

**bool isSubsequence(string s, string t) {**

**int i = 0, j = 0;**

**while (i < s.size() && j < t.size()) {**

**if (s[i] == t[j]) i++;**

**j++;**

**}**

**return i == s.size();**

**}**

**};**

**3)two sumclass Solution {**

**public:**

**vector<int> twoSum(vector<int>& nums, int target) {**

**int n=nums.size();**

**map<int,int>mpp;**

**for(int i=0;i<n;i++){**

**int num=nums[i];**

**int moreneeded=target-num;**

**if(mpp.find(moreneeded)!=mpp.end()){**

**return{mpp[moreneeded],i};**

**}**

**mpp[num]=i;**

**}**

**return{-1,-1};**

**}**

**};**

4)container with most water

**class Solution {**

**public:**

**int maxArea(vector<int>& height) {**

**int maxArea = 0;**

**int left = 0;**

**int right = height.size() - 1;**

**while (left < right) {**

**maxArea = max(maxArea, (right - left) \* min(height[left], height[right]));**

**if (height[left] < height[right]) {**

**left++;**

**} else {**

**right--;**

**}**

**}**

**return maxArea;**

**}**

**};**

**5)3 sum class Solution {**

**public:**

**vector<vector<int>> threeSum(vector<int>& nums) {**

**vector<vector<int>> res;**

**sort(nums.begin(), nums.end());**

**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 (total > 0) {**

**k--;**

**} else if (total < 0) {**

**j++;**

**} else {**

**res.push\_back({nums[i], nums[j], nums[k]});**

**j++;**

**while (nums[j] == nums[j-1] && j < k) {**

**j++;**

**}**

**}**

**}**

**}**

**return res;**

**}**

**};**