***Week – 9 (******13.06.2021 – 19.06.2021)***

***CODES IN PDF***

1. ***Fibonacci Number:***

class Solution {

public:

int fib(int n) {

int arr[n+2], i;

arr[0] = 0;

arr[1] = 1;

for(i=2; i<=n; i++)

arr[i] = arr[i-1] + arr[i-2];

return arr[n];

}

};

1. ***0-1 Knapsack:***

int Solution::solve(vector<int> &A, vector<int> &B, int C) {

int i, w, n;

n = (int)A.size();

int K[n + 1][C + 1];

for(i=0; i<=n; i++)

{

for(w=0; w<=C; w++)

{

if(i==0 || w==0) K[i][w] = 0;

else if (B[i-1] <= w) K[i][w] = max((A[i-1]+K[i-1][w-B[i-1]]), K[i-1][w]);

else K[i][w] = K[i-1][w];

}

}

return K[n][C];

}

1. ***Minimum Path Sum:***

class Solution {

public:

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

int i, j, m, n;

m = grid.size()-1;

n = grid[0].size()-1;

int t[grid.size()][grid[0].size()];

t[0][0] = grid[0][0];

for(i=1; i<=m; i++)

t[i][0] = t[i-1][0] + grid[i][0];

for(j=1; j<=n; j++)

t[0][j] = t[0][j-1] + grid[0][j];

for (i=1; i<=m; i++)

for (j=1; j<=n; j++)

t[i][j] = min(t[i-1][j],t[i][j-1]) + grid[i][j];

return t[m][n];

}

};

1. ***Jump Game II:***

class Solution {

public:

int jump(vector<int>& nums) {

int i, j, n = nums.size(), min;

int jump[n];

for(i=n-2; i>=0; i--)

{

if(nums[i] == 0) jump[i] = INT\_MAX;

else if(nums[i] >= n-i-1) jump[i] = 1;

else

{

min = INT\_MAX;

for (j=i+1; j<n && j<=nums[i]+i; j++)

if (min > jump[j]) min = jump[j];

if (min != INT\_MAX) jump[i] = min + 1;

else jump[i] = min;

}

}

return jump[0];

}

};

1. ***Maximal Square:***

class Solution {

public:

int maximalSquare(vector<vector<char>>& matrix) {

int i, j, max\_i, max\_j, max\_s, area=0;;

int s[matrix.size()][matrix[0].size()];

for(i=0; i<matrix.size(); i++)

s[i][0] = matrix[i][0]-'0';

for(j=0; j<matrix[0].size(); j++)

s[0][j] = matrix[0][j]-'0';

for(i=1; i<matrix.size(); i++)

{

for(j=1; j<matrix[0].size(); j++)

{

if(matrix[i][j] == '1') s[i][j] = min(s[i][j-1],min(s[i-1][j-1],s[i-1][j]))+1;

else s[i][j] = 0;

}

}

max\_s = s[0][0];

max\_i = 0;

max\_j = 0;

for(i=0; i<matrix.size(); i++)

{

for(j=0; j<matrix[0].size(); j++)

{

if(max\_s < s[i][j])

{

max\_s = s[i][j];

max\_i = i;

max\_j = j;

}

}

}

for(i=max\_i; i>max\_i-max\_s; i--)

for(j=max\_j; j>max\_j-max\_s; j--) area++;

return area;

}

};

1. ***Coin Change 2:***

class Solution {

public:

int change(int amount, vector<int>& coins) {

int i, j, x, y;

int table[amount+1][coins.size()];

for(i=0; i<coins.size(); i++)

table[0][i] = 1;

for(i=1; i<amount+1; i++)

{

for(j=0; j<coins.size(); j++)

{

x = (i-coins[j] >= 0) ? table[i-coins[j]][j] : 0;

y = (j>=1) ? table[i][j-1] : 0;

table[i][j] = x+y;

}

}

return table[amount][coins.size()-1];

}

};

1. ***Longest Increasing Subsequence:***

class Solution {

public:

int lengthOfLIS(vector<int>& nums) {

int i, j, list[nums.size()];

list[0] = 1;

for(i=1; i<nums.size(); i++)

{

list[i] = 1;

for(j=0; j<i; j++)

if(nums[i]>nums[j] && list[i] < list[j]+1) list[i] = list[i]+1;

}

return \*max\_element(list, list+nums.size());

}

};

1. ***Maximum Height by Stacking Cuboids:***

class Solution {

public:

int maxHeight(vector<vector<int>>& cuboids) {

for (auto& a : cuboids) sort(a.begin(), a.end());

cuboids.push\_back({0, 0, 0});

sort(cuboids.begin(), cuboids.end());

int i, j, n = cuboids.size(), res = 0;

vector<int> dp(n);

for (i=1; i<n; i++)

for (j=0; j<i; j++)

if(cuboids[j][0] <= cuboids[i][0] && cuboids[j][1] <= cuboids[i][1] && cuboids[j][2] <= cuboids[i][2])

{

dp[i] = max(dp[i], dp[j] + cuboids[i][2]);

res = max(res, dp[i]);

}

return res;

}

};

1. ***Minimum Insertion Steps to Make a String Palindrome:***

class Solution {

public:

int minInsertions(string s) {

int h, l, gap, table[s.size()][s.size()];

for(l=0; l<s.size(); l++)

for(h=0; h<s.size(); h++)

table[l][h] = 0;

for(gap=1; gap<s.size(); gap++)

for(l=0, h=gap; h<s.size(); l++, h++)

table[l][h] = (s[l] == s[h]) ? table[l+1][h-1] : (min(table[l][h-1],table[l+1][h])+1);

return table[0][s.size()-1];

}

};

1. ***Longest Palindromic Substring:***

class Solution {

public:

string longestPalindrome(string s) {

int n =s.size();

int dp[n][n], gap, i, j;

string str ="";

for(gap=0; gap<n; gap++)

{

for(i=0, j=gap; j<n; i++, j++)

{

if(gap == 0) dp[i][j] = 1;

else if (gap ==1)

{

if(s[i] == s[j]) dp[i][j] =1;

else dp[i][j] =0;

}

else

{

if(s[i] == s[j] && dp[i+1][j-1] ==1) dp[i][j] =1;

else dp[i][j] = 0;

}

if(dp[i][j] && j - i + 1 > str.size())

str = s.substr(i, j - i + 1);

}

}

return str;

}

};

1. ***Longest Palindromic Subsequence:***

class Solution {

public:

int longestPalindromeSubseq(string s) {

int n=s.length(), ans=1, i, j, k, dp[n][n];

for(i=0; i < n; i++)

for(j =0; j < n; j++)

dp[i][j] = 0;

for (i=0; i<n; i++)

{

dp[i][i] = 1;

if(i!=n-1 and s[i]==s[i+1])

{

dp[i][i+1]=2;

ans=2;

}

if(i!=n-1 and s[i]!=s[i+1]) dp[i][i+1]=1;

}

for (k=2; k<n; k++)

for(i=0; i<n-k; i++)

{

j=i+k;

if(s[i] == s[j])

{

dp[i][j] = dp[i+1][j-1]+2;

ans = max(ans,dp[i][j]);

}

else dp[i][j] = max(dp[i][j-1],dp[i+1][j]);

}

return ans;

}

};

1. ***Super Egg Drop:***

class Solution {

public:

int fun(int eggs,int floors, vector<vector<int>> &dp) {

if(floors==1||floors==0) return floors;

if(eggs==1) return floors;

if(dp[eggs][floors]!=-1) return dp[eggs][floors];

int res,ans=INT\_MAX,l=1,h=floors,temp=0;

while(l<=h)

{

int mid=l+(h-l)/2;

int left=fun(eggs-1,mid-1,dp);

int right=fun(eggs,floors-mid,dp);

temp=1+max(left,right);

if(left<right) l=mid+1;

else h=mid-1;

ans=min(ans,temp);

}

return dp[eggs][floors] = ans;

}

int superEggDrop(int eggs, int floors) {

vector<vector<int>> dp(eggs+1,vector<int> (floors+1,-1));

return fun(eggs,floors,dp);

}

};

1. ***Max Sum of Rectangle No Larger Than K:***

class Solution {

public:

int maxSumSubmatrix(vector<vector<int>>& matrix, int k) {

int i, j, l=0, r=0, res=INT\_MIN;

for(l=0; l<matrix[0].size(); l++)

{

vector<int> sum(matrix.size(),0);

for(r=l; r<matrix[0].size(); r++)

{

for(i=0; i<matrix.size(); i++)

sum[i] += matrix[i][r];

for(i=0; i<sum.size(); i++)

{

int s = sum[i];

for(j=i+1; j<sum.size(); j++)

{

if(s <= k) res = max(res, s);

s += sum[j];

}

if(s <= k) res = max(res, s);

}

if(res == k) return k;

}

}

return res;

}

};

1. ***Palindrome Partitioning II:***

class Solution {

public:

int minCut(string s) {

int n = s.size(), i, j;

int cut[n];

bool p[n][n];

memset(p, false, sizeof(p));

for (i=0; i<n; i++)

{

int minCut = i;

for (j=0; j<=i; j++)

{

if (s[i] == s[j] && (i - j < 2 || p[j + 1][i - 1]))

{

p[j][i] = true;

minCut = min(minCut, j == 0 ? 0 : (cut[j - 1] + 1));

}

}

cut[i] = minCut;

}

return cut[n - 1];

}

};

1. ***Number of Dice Rolls With Target Sum:***

class Solution {

public:

int numRollsToTarget(int d, int f, int target) {

if(d>target || d\*f<target) return 0;

vector<vector<int>> dp(d+1, vector<int>(target+1, 0));

int MOD = 1000000007, i, j;

dp[0][0] = 1;

for (i=1; i<=d; i++)

{

for (j=1; j<=target; j++)

{

dp[i][j] = (dp[i][j-1] + dp[i-1][j-1]) % MOD;

if (j > f)

{

dp[i][j] -= dp[i-1][j-f-1];

if (dp[i][j] < 0) dp[i][j] += MOD;

}

}

}

return dp[d][target];

}

};