***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());

}

};