**Algorithmic Problem Solving 2019**

**Q-Box Assignment Set**

Student Name: Vinay Karnam

SRN: 01FE16BCS252

Branch: CSE

**Question 02:**

Title: **AVENGERS**:**ENDGAME**

Level: Medium

Concepts Tested: Dynamic programming , Knapsack problem

**Problem Statement:**

We are in the ENDGAME now!!! , Thanos just wiped out half of the universe with just snap of his fingers. Half of the people in the universe turned in to ash with just the snap, unfortunately spiderman is one among them. Iron Man the lead of avengers wants to bring all the people back on the universe. Thanos is unstoppable in this universe with the infinity gauntlet with him. The iron man should build the strongest avengers team to fight the thanos. Nebula the daughter of Thanos knows where Thanos lives now. But unfortunately there is only one supersonic ship to carry the avengers there. The ship is not capable to carry all the superheroes. Each avenger is marked with strength and weight. Given the capacity of the ship, help the Iron Man to choose the superheroes to build the strongest avengers team and to carry in the ship to fight Thanos.

**Input Format:**

First line of the input contain N , C . N the number of superheroes available and C the capacity of the ship. Next N line contains W and S, W the weight and S the strength of the super hero.

**Constraints:**

0<N<100;

0<=W,S,C<;

**Output Format:**

Print a single value representing the maximum possible total strength of the Avengers team carrying in the supersonic ship.

**Solution:**

**#include<stdio.h>**

**#include<iostream>**

**using namespace std;**

**int max(int a, int b) { return (a > b)? a : b; }**

**int knapSack(int W, int wt[], int val[], int n)**

**{**

**int i, w;**

**int K[n+1][W+1];**

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

**{**

**for (w = 0; w <= W; w++)**

**{**

**if (i==0 || w==0)**

**K[i][w] = 0;**

**else if (wt[i-1] <= w)**

**K[i][w] = max(val[i-1] + K[i-1][w-wt[i-1]], K[i-1][w]);**

**else**

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

**}**

**}**

**return K[n][W];**

**}**

**int main()**

**{**

**int val[100],wt[100],W,n;**

**cin>>n>>W;**

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

**{**

**cin>>wt[i]>>val[i];**

**}**

**printf("%d", knapSack(W, wt, val, n));**

**return 0;**

**}**

**Sample Test Cases:**

Input :

3 50

10 60

20 100

30 120

Output:

220

All possible teams for example are {(10,60),(20,100)},{(10,60),(30,120)},

{(20,100),(30,120)} We can achieve the maximum value from last set.

**Question 01:**

Title: Missing pieces

Level: Easy

Concepts Tested: DFS, BFS

**Problem Statement:**

Kira loves playing puzzles. Unfortunately, she has lost final few pieces needed to complete the puzzle.

Let matrix A denote the puzzle. The values with ones represent the missing piece. Adjacent zeros represent a single missing piece.

Help her find the number of missing pieces.

**Input Format:**

First line contains two integers, M,N.

Following M line contains N spaced integer, the values of matrix A.

**Constraints:**

1 <= M,N <= 100

A[i][j] ∊ [0,1]

**Output Format:**

A single integer value denoting the number of missing pieces.

**Solution:**

#include <stdio.h>

#include <string.h>

#include <stdbool.h>

#define ROW 100

#define COL 100

int isSafe(int M[][COL], int row, int col, bool visited[][COL])

{

return (row >= 0) && (row < ROW) &&

(col >= 0) && (col < COL) &&

(M[row][col] && !visited[row][col]);

}

void DFS(int M[][COL], int row, int col, bool visited[][COL])

{

static int rowNbr[] = {-1, -1, -1, 0, 0, 1, 1, 1};

static int colNbr[] = {-1, 0, 1, -1, 1, -1, 0, 1};

visited[row][col] = true;

for (int k = 0; k < 8; ++k)

if (isSafe(M, row + rowNbr[k], col + colNbr[k], visited) )

DFS(M, row + rowNbr[k], col + colNbr[k], visited);

}

int countIslands(int M[][COL],int row,int col)

{

bool visited[ROW][COL];

memset(visited, 0, sizeof(visited));

int count = 0;

for (int i = 0; i < row; ++i)

for (int j = 0; j < col; ++j)

if (M[i][j] && !visited[i][j])

{

DFS(M, i, j, visited);

++count;

}

return count;

}

int main()

{

int row,col,i,j;

int M[100][100];

scanf("%d %d",&row,&col);

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

{

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

{

scanf("%d",&M[i][j]);

}

}

printf("%d", countIslands(M,row,col));

return 0;

}

**Sample Test Cases:**

**Input:**

6 6

1 1 1 1 1 0

1 0 0 1 1 1

1 1 0 1 1 1

1 1 1 1 1 1

1 1 1 1 0 0

1 1 1 1 0 0

**Output:**

**1**