**LAB 4: DYNAMIC PROGRAMMING**

**Date : 4.4.18**

**Submitted By : Abhineet Singh**

**1. *Write a program that implement a 0/1 knapsack problem using dynamic programming and find an optimal solution to the 0/1 Knapsack problem with instance n=3, m=6 , (p1, p2,P3)=(1,2,5), and (w1,w2,w3)=(2,3,4).***

#include<stdio.h>

int m = 3;

int max(int a, int b) {

return (a > b)? a : b; }

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

{

if (n == 0 || W == 0){

if(n!=0){

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

A[i]=0;

}

}

return 0;

}

if (wt[n-1] > W){

A[n-1]=0;

return knapSack(A,W, wt, val, n-1);

}

else

return max( val[n-1] + knapSack(A,W-wt[n-1], wt, val, n-1),

knapSack(A,W, wt, val, n-1));

}

int main()

{

int val[] = {1, 2, 5};

int wt[] = {2, 3, 4};

int A[] = {1,1,1};

int W = 5;

int n = sizeof(val)/sizeof(val[0]);

m = n;

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

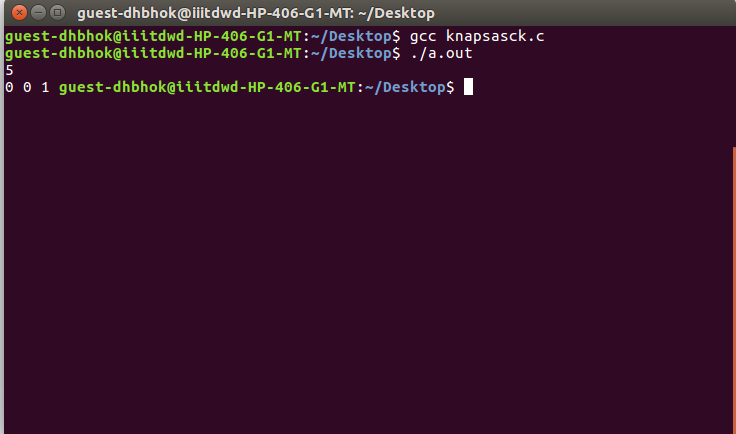
for(int i=0; i<3;i++){

printf("%d ", A[i]);

}

return 0;

}



**2. *Write a program that implement all pair shortest path of a graph i.e., (Floyd-Warshall Algorithm) using Dynamic programming method.***

#include<iostream>

using namespace std;

int main(){

int cost[3][3],a[3][3];

cout<<"For Infinity type 9999."<<endl;

for(int i = 0; i < 3; i++){

for(int j = 0; j < 3; j++){

cin>>a[i][j];

}

}

for(int k=0; k < 3; k++){

for(int j=0;j < 3; j++){

for(int i=0; i < 3; i++){

a[i][j] = min(a[i][j], a[i][k]+a[k][j]);

}

}

}

cout<<"output:"<<endl;

for(int i = 0; i < 3 ; i++){

for(int j = 0; j < 3; j++){

cout<<a[i][j]<<" ";

}

cout<<endl;

}

}

