

**Department of Computer Science and Engineering**

**Mini-Project Report-(02)**

**Course Name:** Algorithms.

**Course Code:** CSE246.

## Section No: 01.

**Date of submission:**  Jan, 2023.

**Problem No:** 02.

**Submitted By:**

**Student’s Name:** Ab. Rahim Ahmed Sowrov.

**Student’s ID:** 2020-3-60-070.

**Student’s Name:** Sumaiya Tabassum Zara.

**Student’s ID:** 2021-1-60-042.

**Student’s Name:** Sumiaya Ahmed.

**Student’s ID:** 2019-3-60-117.

**Student’s Name:** Pulak Islam.

**Student’s ID:** 2020-2-60-193.

## Submitted To

**Dr. Taskeed Jabid**

Faculty of sciences and Engineering

Chairperson, Associate Professor, Department of C SE.

**Problem statement**

You are a participant of an adventure game and there are different rooms where one flag is hidden in each room. You have to collect flags from these rooms, and you may choose or skip any room. You cannot go back once you move forward. Each flag is associated with points. Find the flags in a way so that you can achieve maximum points in the game. Consider that, Color array is c which can be any integer and there are flags of 4 colors and 8 rooms so some flags will have the same-colored flag.

• You can define the color array as per your choice, array size = 8

• Point array for the flags: {2, 8, 3, 5, 11, 9, 1, 4}

• m & n are two integers that can be positive or negative

Where,

a) m will be taken into calculation if you take a same-colored flag that you already have, and it will be multiplied with the corresponding points.

b) n will be taken into consideration if you take a different-colored flag from the ones you already have, and it will be multiplied with the corresponding points.

**Methodology**

This Project has been solved using a greedy approach. The problem is about finding the flags in a way that we can achieve maximum points in the game. So, we are greedy about the flag that gives us the maximum point. We have the option to keep the flag, or we can skip it. So, if there is any flag that will negatively affect our total points, we can simply skip that flag. There are two integers [m],[n]. [n] will be taken into consideration if we take a different-colored flag and [m] will be taken into calculation if you take a same-colored flag. So, we have to keep track of whether we are taking the same-colored flag or not.

Therefore, we will choose the flag in a sequence that chosen flag is giving us the maximum point than the remaining flags. By considering all the flags, we can achieve maximum points from the game.

**Discussion**

This project has been completed in the C program. First, we have defined the color name and index of the color. Then we have taken an array “selectedIndexforColor[4]={-1,-1,-1,-1};” to keep track of whether that color is used or not. Similarly, we have taken an array “ isSelected[8]={0,0,0,0,0,0,0,0};” to keep track of whether we have visited that room or not. Then there are two for loops, inner for loop is to visit all the rooms and find the flag with the maximum point. The outer loop is for overall calculation. The C-code for this code is given below

#include<stdio.h>  
 *int* red=0,blue=1,green=2,yellow=3;  
*int* main(){  
 printf("\n --- --- --- Welcome to this adventure game--- --- ---");  
 printf("\nThere are flags of 4 colors and 8 rooms\n");  
 *int* selectedIndexforColor[4]={-1,-1,-1,-1};  
 *int* c[8]={ red, blue , green, red, yellow,blue, yellow, green};  
 *char* color[8][10]={"red","blue","green","red","yellow","blue","yellow","green"};  
 *int* points[8]={ 2, 8, 3, 5, 11, 9, 1, 4};  
 *int* isSelected[8]={0,0,0,0,0,0,0,0};  
 *int* max\_score=0;  
 printf("\nThe point array for 8 room is: [");  
 *for*(*int* i=0;i<8;i++)  
 printf("%d ",points[i]);  
 printf("]\nFind the flags in a way so that you can achieve maximum points in the game.");  
 printf("\n --- --- --- --- --- --- --- --- --- --- --- ---\n");  
 *int* m,n;  
 printf("\nEnter Value of m: ");  
 scanf("%d",&m);  
 printf("\nEnter Value of n: ");  
 scanf("%d",&n);  
 *int* index,max,tmp\_score,count=0;  
 printf("\nSelected Flags are: ");  
  
 *for*(*int* i=0;i<8;i++){  
 max=0;  
 *for*(*int* j=0;j<8;j++){  
 tmp\_score=n\*points[j];  
 *if*(tmp\_score>=max && isSelected[j]==0){  
 max=tmp\_score;  
 index=j;  
 }  
 }  
 tmp\_score=max;  
 isSelected[index]=1;  
 *if*(selectedIndexforColor[c[index]]==-1){  
 *if*(tmp\_score>=0){  
 max\_score=max\_score+tmp\_score;  
 selectedIndexforColor[c[index]]=index;  
 count++;  
 printf("\n Room no: %d where color of the flag is %s corresponding point is: (%d X %d)=%d",index+1,color[index],points[index],n,tmp\_score);  
 }  
 *else*{  
 printf("\n Room no: %d where color of the flag is %s corresponding point is:%d --->[which is not selected]",index+1,color[index],tmp\_score);  
  
 }  
 }  
 *else*{  
 *int* tmp\_score2=m\*points[index];  
 *if*(tmp\_score2>=0 && tmp\_score2>=tmp\_score){  
 max\_score=max\_score+tmp\_score2;  
 count++;  
 printf("\n Room no: %d where color of the flag is %s corresponding point is: (%d X %d)=%d",index+1,color[index],points[index],m,tmp\_score2);  
 }  
 *else*{  
 printf("\n Room no: %d where color of the flag is %s corresponding point is:(%d) --->[which is not selected]",index+1,color[index],tmp\_score2);  
 }  
  
 }  
  
 }  
 printf("\n\n Maximum Points: %d",max\_score);  
 printf("\n The Number of Selected flag is: %d\n",count);  
 }

**Time Complexity**

As the following code consists of two loops, therefore the time complexity should be the order of O(n^2). In this project, there are 8 rooms so the number of operations will be,

8^2=16

This complexity can be reduced by sorting the point array in descending order and the first index of the point array will be the flag with the maximum point. So, we don’t have to use the nested loop and overall complexity should be in the order of O(n Log n).