

DERSİN ADI: Algoritma Analizi

DERSİN EĞİTMENİ: Dr. Öğr. Üyesi Mehmet Amaç GÜVENSAN

ÖĞRENCİ ADI: Ertuğrul ŞENTÜRK

ÖĞRENCİ NO: 18011028

ÖĞRENCİ MAIL: mdesenturk@gmail.com

DÖNEM: 3

GRUP NO: 2

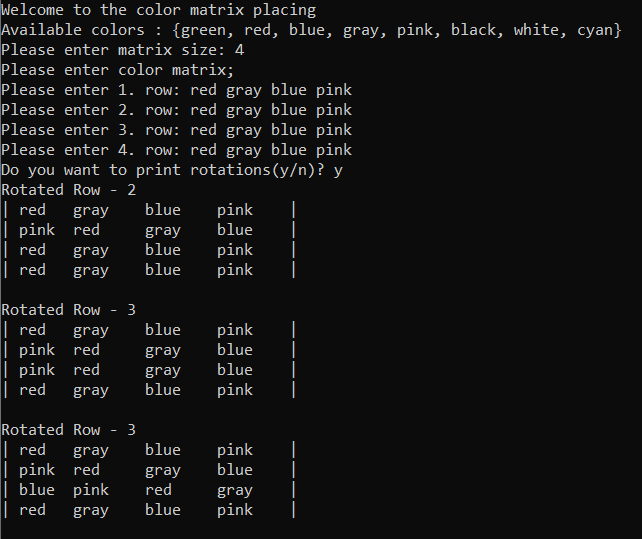
ÖDEV NO: 4

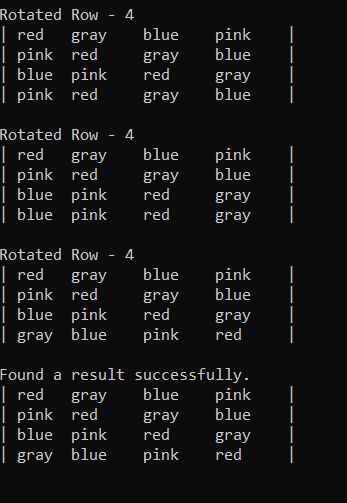
SORU NO: 1

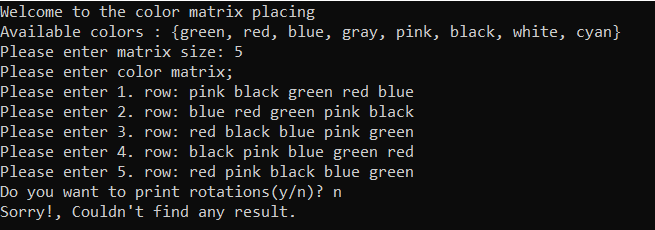
ÖDEV KONUSU: Backtracking

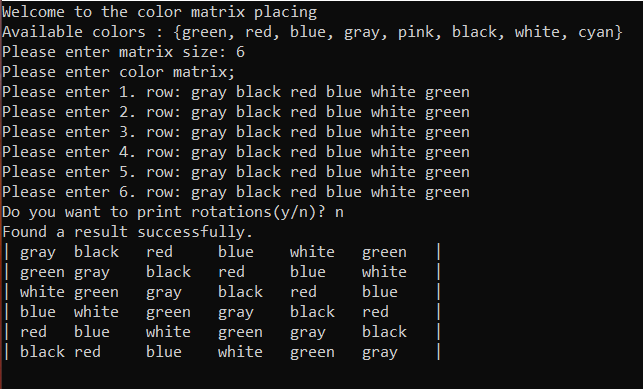
Algoritma:

1. Önceden tanımlı renk dizisi yazdirildi.
2. Kullanıcıdan matris boyutu renk matrisi ve işlem adımlarını isteyip istemediğine dair bilgi alındı.
3. Matris alınırken renk dizisindeki elemanların indisi matrise kaydedildi. Girilen eleman renk dizisinde yoksa program sonlandırıldı.
4. Backtracking fonksiyonu çağırıldı bu fonksiyon en üst satırdan başlanarak hesaplandı.
5. İşlem yapılan her satır kendinden üstteki satırlarla karşılaştırıldı ve aynı sütünda satırdaki elemanlarla aynı olan eleman olup olmadığına bakıldı.
6. Eğer tüm elemanlar farklı ise fonksiyon bir alt satır için çağırıldı.
7. En alt satırı geçtiğimiz durumda fonksiyon true döndürdü.
8. Eğer satırdaki elemanlardan biri aynı ise satırdaki eleman sayısı kadar satır döndürülerek işlem tekrarlandı.
9. Döndürme işlemi her yapıldığında eğer kullanıcı işlem adımlarını istemişse output verildi.
10. Eğer en üst satırda satır elemanı kadar dönme işlemi yapılmışsa sonuç false olarak döndürüldü.
11. Sonucun true veya false olma durumuna göre kullanıcıya output verildi.

Ekran görüntüleri:







#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <string.h>

// maximum renk sayisi

#define MAX\_COLOR\_COUNT 8

// renk kelimesi icin olusturulan buffer boyutu

#define COLOR\_SIZE 50

// renk isimlerini tutan dizi

const static char colors[MAX\_COLOR\_COUNT][COLOR\_SIZE] ={"green","red","blue","gray","pink","black","white","cyan"};

// Input fonksiyonlari

int\*\* get\_color\_matrix(int size);

void clear\_color\_matrix(int\*\* matrix,int size);

void print\_colors();

// Rengin dizideki sirasini bulan fonksiyon

int find\_color\_index(char color[COLOR\_SIZE]);

// Matrisi yazdiran fonksiyon

void print\_matrix(int\*\* matrix,int size);

// Matrisin verilen satirini donduren fonksiyon

void rotate\_matrix(int\*\* matrix,int size,int row,char print);

// Satirdaki elemanlari ust satirlardaki ile karsilastirip satirin gecerli olup olmadigini donduren fonksiyon

bool is\_valid\_row(int\*\* matrix,int size,int row);

// Backtracking ile rekursif olarak matrisin elemanlarinin sirasini duzenleyen fonksiyon

bool back\_tracking(int\*\* matrix,int size,int row,char print);

/\* tanim : main fonksiyon

- Kullanicidan gerekli inputlar get\_color\_list ve get\_color\_matrix ile alinip matris olusturulmustur.

- Daha sonra matris back\_tracking fonskiyonu ile duzenlenmistir.

- Sonucun basarili veya basarisiz olmasi durumlari kullaniciya yazdirilmistir.

i = iterator

size = matris boyutu

print = adimlarin yazdirilip yazdirilmayacagini tutan degisken

colors = renklerin isimlerini tutan dizi

matrix = renklerin integer degerlerinin saklandigi matris

return 0 = program tamamlandi.

\*/

int main(){

int i;

int size;

char print;

int\*\* matrix;

printf("Welcome to the color matrix placing\n");

print\_colors();

printf("Please enter matrix size: ");

scanf("%d",&size);

if(size<3||size>8){

printf("Invalid size\n");

return 0;

}

matrix = get\_color\_matrix(size);

if(matrix == NULL)

return 0;

printf("Do you want to print rotations(y/n)? ");

scanf(" %c",&print);

bool result = back\_tracking(matrix,size,0,print);

if(result){

printf("Found a result successfully.\n");

print\_matrix(matrix,size);

}

else

printf("Sorry!, Couldn't find any result.\n");

clear\_color\_matrix(matrix,size);

return 0;

}

/\*

tanim : verilen sayi matrisinin istenilen satirini saga dogru rotate eden fonksiyon.

islem adimlari:

- verilen matrisin verilen satirindaki son eleman saklanir

- satirdaki elemanlar sondan baslanarak bir sonraki elemana tasinir.

- satirin saklanan sol elemani ilk elemana atanir.

- print degiskeni 'y' veya 'Y' gelmisse yazdirma yapilir.

parametreler:

matrix = input olarak alinan sayilari tutan matris

size = matris boyutu

row = dondurulecek satir

print = dondurme adimlarinin yazdirilip yazdirilmayacagini belirten degisken

degiskenler:

i = iterator

temp = matrisin verilen satirinin son elemanini saklayan degisken

\*/

void rotate\_matrix(int\*\* matrix,int size,int row,char print){

int i;

int temp = matrix[row][size-1];

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

matrix[row][i+1]=matrix[row][i];

matrix[row][0] = temp;

if(print == 'y' || print == 'Y'){

printf("Rotated Row - %d\n",row+1);

print\_matrix(matrix,size);

}

}

/\*

tanim : verilen sayi matrisinin istenilen satirini saga dogru rotate eden fonksiyon.

islem adimlari:

- verilen matrisin verilen satirindaki son eleman saklanir

- satirdaki elemanlar sondan baslanarak bir sonraki elemana tasinir.

- satirin saklanan sol elemani ilk elemana atanir.

- print degiskeni 'y' veya 'Y' gelmisse yazdirma yapilir.

parametreler:

matrix = input olarak alinan sayilari tutan matris

size = matris boyutu

row = dondurulecek satir

print = dondurme adimlarinin yazdirilip yazdirilmayacagini belirten degisken

degiskenler:

i = iterator

temp = matrisin verilen satirinin son elemanini saklayan degisken

\*/

void rotate\_matrix(int\*\* matrix,int size,int row,char print){

int i;

int temp = matrix[row][size-1];

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

matrix[row][i+1]=matrix[row][i];

matrix[row][0] = temp;

if(print == 'y' || print == 'Y'){

printf("Rotated Row - %d\n",row+1);

print\_matrix(matrix,size);

}

}

/\*

tanim : verilen sayi matrisinin istenilen satirinin tum elemanlarini ust satirdakilerle karsilastirip ayni eleman olup olamdigini

kontrol eden fonksiyon.

islem adimlari:

- satir 0 icin islem yapilmaz true dondurulur.

- her eleman kendinden dusuk satir numarasina sahip ve ayni sutunda olan elemanlarla karsilastirilir.

- eger herhangi bir esitlik olursa false dondurulur

- tum satir elemanlari icin esitlik yosa true dondurulur.

parametreler:

matrix = input olarak alinan sayilari tutan matris

size = matris boyutu

row = dondurulecek satir

degiskenler:

i,j = iterator

return:

true, false = satirin gecerli olup olmadiginin bilgisi

\*/

bool is\_valid\_row(int\*\* matrix,int size,int row){

int i,j;

if(row==0)

return true;

for(i=0;i<size;i++){

j=row-1;

while(j>=0 && matrix[row][i]!=matrix[j][i])

j--;

if(j>=0)

return false;

}

return true;

}

/\*

tanim : verilen matrisi backtracking algoritmasi ile ayni sutunda ortak eleman olmayacak sekilde duzenleyen fonksiyon

islem adimlari:

- eger satir sayisi matris boyutuna ulasmissa artik bi karsilastirma yapilamayacagindan sonuca ulasilmis olur o nedenle true donduruldu

- matrisin verilen satiri gecerli bir satirsa fonksiyon rekursif olarak bir alt satir icin de cagirilir.

- eger gecersiz durum varsa o satir renk sayisi kadar rotate edilir ve tum durumlara bakilir.

- tum durumlar gecersizse bir ust satira geri donulup onun da tum durumlarina bakilir.

- eger ilk satir renk sayisi kadar dondurulmus ve hala sonuc bulunamamissa false dondurulur.

- eger en alt satirdan en ust satira kadar rekursif fonksiyon true donmusse fonksiyon true dondurur.

parametreler:

matrix = input olarak alinan sayilari tutan matris

size = matris boyutu

row = dondurulecek satir

print = dondurme adimlarinin yazdirilip yazdirilmayacagini belirten degisken

degiskenler:

i = iterator

return:

true, false = gecerli sonuc olup olmadiginin bilgisi

\*/

bool back\_tracking(int\*\* matrix,int size,int row,char print){

int i = 0;

if(row == size)

return true;

for(i=0;i<size;i++){

if(is\_valid\_row(matrix,size,row) && back\_tracking(matrix,size,row+1,print))

return true;

rotate\_matrix(matrix,size,row,print);

}

return false;

}

/\*

tanim : verilen rengi renkler dizisinde arayip indexini donduren fonksiyon.

parametreler:

color = aranilan renk

size = matris boyutu

degiskenler:

i = rengin dizideki indexi

colors = renklerin isimlerini tutan dizi

return:

colors = renklerin isimlerini tutan dizi

\*/

int find\_color\_index(char color[COLOR\_SIZE]){

int i=0;

while(i<MAX\_COLOR\_COUNT && strcmp(color,colors[i]))

i++;

if(i>=MAX\_COLOR\_COUNT)

return -1;

return i;

}

/\* tanim : istenilen adette renk input olarak alinip dinamik memory allocation ile bir dizi olusturan fonksiyon.

parametreler:

size = matris boyutu

degiskenler:

i,j = iterator

color\_index = input olarak alinan rengin renk dizisindeki indisi

colors = renklerin isimlerini tutan dizi

temp = input olarak alinan rengi saklayan buffer

matrix = input olarak alinan sayilari tutan matris

return

matrix = input olarak alinan sayilari tutan matris

\*/

int\*\* get\_color\_matrix(int size){

int i,j;

int color\_index;

char temp[COLOR\_SIZE];

int\*\* matrix;

printf("Please enter color matrix;\n");

matrix = (int\*\*)malloc(sizeof(int\*)\*size);

for(i=0;i<size;i++){

printf("Please enter %d. row: ",i+1);

matrix[i] = (int\*)malloc(sizeof(int)\*COLOR\_SIZE);

for(j=0;j<size;j++){

scanf("%s",temp);

color\_index = find\_color\_index(temp);

if(color\_index==-1){

printf("Invalid entry\n");

return NULL;

}

matrix[i][j] = color\_index;

}

}

return matrix;

}

/\*

tanim : input olarak alinan sayilari tutan matrisi temizleyen fonksiyon.

parametreler:

matrix = input olarak alinan sayilari tutan matris

size = matris boyutu

\*/

void clear\_color\_matrix(int\*\* matrix,int size){

int i;

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

free(matrix[i]);

free(matrix);

}

/\* tanim : girilebilecek renkleri yazdiran matris.

degiskenler:

i = iterator

colors = renklerin isimlerini tutan dizi

\*/

void print\_colors(){

int i;

printf("Available colors : {");

for(i=0;i<MAX\_COLOR\_COUNT-1;i++){

printf("%s, ",colors[i]);

}

printf("%s}\n",colors[i]);

}

/\*

tanim : verilen sayi matrisini renkler matrisindeki degerlere gore yazdiran fonksiyon.

parametreler:

matrix = input olarak alinan sayilari tutan matris

size = matris boyutu

degiskenler:

i,j = iterator

colors = renklerin isimlerini tutan dizi

\*/

void print\_matrix(int\*\* matrix,int size){

int i,j;

for(i=0;i<size;i++){

printf("| ");

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

printf("%s\t",colors[matrix[i][j]]);

printf("|\n");

}

printf("\n");

}