

TÜRKİYE CUMHURİYETİ
YILDIZ TEKNİK ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ



ALGORİTMA ANALİZİ
DÖNEM PROJESİ

Öğrenci No: 19011010

Öğrenci Adı Soyadı: Beyda Güler

Öğrenci e-posta: beyda.guler@std.yildiz.edu.tr

Ders / Grup: BLM3021-Algoritma Analizi / Grup:1

Ders Yürütücüsü:
Mine Elif KARSLIGİL
Ocak , 2023

İçindekiler

YÖNTEM	3
Problem	3
Çözüm	3
UYGULAMA.....	3
MENU	3
NORMAL MOD.....	4
DETAYLI MOD	5
SONUÇ	7
RotateTheRow() Fonksiyonunun Zaman ve Yer Karmaşıklığı.....	7
IsSafe() Fonksiyonunun Zaman ve Yer Karmaşıklığı	7
Solve() Fonksiyonunun Zaman ve Yer Karmaşıklığı	8
VIDEO LINKİ:	8

YÖNTEM

Problem

Kullanıcıdan alınan N sayısı kadar renkle oluşturulan NxN'lik matrisin her kolonunda renkler bir defa tekrar etmelidir. Bunu sağlamak için backtracking mantığı kullanılması istenmektedir.

Çözüm

```
typedef struct color{
    char colorName[20];
    int colorNo;
}COLOR;
```

- **COLOR** CreateMatrix(COLOR** matrix , int N)**

Kullanıcının girdileriyle NxN'lik bir matris oluşturulur. Matrisin her gözünde bir renk ve o renk için oluşturulan renk kodu bulunmaktadır.

- **int IsSafe(COLOR** matrix, int row, int N)**

Fonksiyona girdi olarak verilen 'row' parametresi o anda bulunduğumuz satırı göstermektedir. IsSafe fonksiyonu, bu satırı üstündeki diğer bütün satırlarla karşılaştırarak çakışma olup olmadığını (safe / not safe) tespit etmektedir. Safe ise 1, not safe ise 0 döndürmektedir.

- **COLOR* RotateTheRow(COLOR* row,int N)**

IsSafe fonksiyonundan not safe olarak dönen satırı safe hale getirmek için rotate yapılmasını sağlayan fonksiyondur. Kaydırma işlemi dairesel bir şekilde gerçekleşmektedir.

- **int Solve(COLOR** matrix, int N, int row,int mode)**

IsSafe ve RotateTheRow fonksiyonlarını kullanarak recursive bir şekilde her satırı safe hale getirmeye çalışmaktadır. Backtracking mantığını kullanır. Yani eğer 1. Ve 2. Satırlar safe durumda fakat 3. Satır not safe durumdaysa bir üstteki satıraçıkılır, yeniden rotate edilir. Rotate'ten sonra 2 hala safe ise 3. Satıra geri dönülür ve bu durum için ihtimaller denenir.

UYGULAMA

MENU

```
19011010
BEYDA GULER

1-NORMAL MODE
2-DETAILED MODE

Select Mode -> 1

Enter the number of colors -> 3
```

- Normal Mod'da sadece girilen matris için eğer çözüm varsa rotate edilmiş matris, eğer çözüm yoksa NO SOLUTION çıktısı verilir.
- Detaylı Mod'da bütün rotation adımları çözüm olsa da olmasa da gösterilir.

NORMAL MOD

*** Her harf bir renk olarak düşünülmüştür.

```
----- NORMAL MODE -----  
  
-----  
INITIAL MATRIX  
-----  
  
a   s   d   f  
s   d   f   a  
s   a   f   d  
f   d   s   a  
  
-----  
ROTATED MATRIX  
-----  
  
a   s   d   f  
d   f   a   s  
s   a   f   d  
f   d   s   a
```

Şekil 1 - Sonuç elde edildiği durum

```
----- NORMAL MODE -----  
  
-----  
INITIAL MATRIX  
-----  
  
RED   BLUE  CYAN  
BLUE  RED   CYAN  
CYAN  RED   BLUE  
  
-----  
ROTATED MATRIX  
-----  
  
NO SOLUTION!!
```

Şekil 2 - Sonuç elde edilemeyen durum

DETAYLI MOD

```
Select Mode -> 2
Enter the number of colors -> 3

***1st row***

1st column ---> BLUE
2st column ---> BLACK
3st column ---> RED

***2st row***

1st column ---> BLUE
2st column ---> BLACK
3st column ---> RED

***3st row***

1st column ---> BLUE
2st column ---> RED
3st column ---> BLACK
```

Şekil 3 - Input

```
----- DETAILED MODE -----

BLUE   BLACK   RED
RED    BLUE   BLACK
BLUE   RED    BLACK

BLUE   BLACK   RED
RED    BLUE   BLACK
BLACK  BLUE   RED

BLUE   BLACK   RED
RED    BLUE   BLACK
RED    BLACK  BLUE

BLUE   BLACK   RED
RED    BLUE   BLACK
BLUE   RED    BLACK

BLUE   BLACK   RED
BLACK  RED    BLUE
BLUE   RED    BLACK

BLUE   BLACK   RED
BLACK  RED    BLUE
BLACK  BLUE   RED

BLUE   BLACK   RED
BLACK  RED    BLUE
RED    BLACK  BLUE

BLUE   BLACK   RED
BLACK  RED    BLUE
BLUE   RED    BLACK

BLUE   BLACK   RED
BLUE   BLACK   RED
BLUE   RED    BLACK

NO SOLUTION!!
```

Şekil 4 - Rotation adımları ve sonuç

```

Select Mode -> 2
Enter the number of colors -> 4

***1st row***
1st column ---> BLUE
2st column ---> RED
3st column ---> WHITE
4st column ---> CYAN

***2st row***
1st column ---> BLUE
2st column ---> RED
3st column ---> WHITE
4st column ---> CYAN

***3st row***
1st column ---> BLUE
2st column ---> RED
3st column ---> WHITE
4st column ---> CYAN

***4st row***
1st column ---> BLUE
2st column ---> RED
3st column ---> WHITE
4st column ---> CYAN

```

Şekil 5 - Input

```

----- DETAILED MODE -----

BLUE  RED  WHITE  CYAN
CYAN  BLUE  RED   WHITE
BLUE  RED  WHITE  CYAN
BLUE  RED  WHITE  CYAN

BLUE  RED  WHITE  CYAN
CYAN  BLUE  RED   WHITE
CYAN  BLUE  RED   WHITE
BLUE  RED  WHITE  CYAN

BLUE  RED  WHITE  CYAN
CYAN  BLUE  RED   WHITE
WHITE  CYAN  BLUE  RED
BLUE  RED  WHITE  CYAN

BLUE  RED  WHITE  CYAN
CYAN  BLUE  RED   WHITE
WHITE  CYAN  BLUE  RED
CYAN  BLUE  RED   WHITE

BLUE  RED  WHITE  CYAN
CYAN  BLUE  RED   WHITE
WHITE  CYAN  BLUE  RED
WHITE  CYAN  BLUE  RED

BLUE  RED  WHITE  CYAN
CYAN  BLUE  RED   WHITE
WHITE  CYAN  BLUE  RED
RED   WHITE  CYAN  BLUE

```

Şekil 6 - Rotation adımları ve sonuç

SONUÇ

CreateMatrix() Fonksiyonunun Zaman ve Yer Karmaşıklığı

```
COLOR** CreateMatrix(COLOR** matrix , int N){  
  
    int i,j,k;  
    for(i = 0 ; i < N ; i++){  
        printf("\n\t\t\t\t ***dst row***\n",i+1);  
        for(j = 0 ; j < N ; j++){  
            printf("\n\t\t\t\t\t dst column ---> ",j+1);  
            scanf("%s",matrix[i][j].colorName);  
            if(i == 0)  
                matrix[i][j].colorNo = j+1;  
            else{  
                k=0;  
                while(strcmp(matrix[0][k].colorName,matrix[i][j].colorName) != 0)  
                    k++;  
                matrix[i][j].colorNo = matrix[0][k].colorNo;  
            }  
        }  
    }  
    return matrix;  
}
```

Zaman : $O(N^2)$

Yer : $O(N^2)$

RotateTheRow() Fonksiyonunun Zaman ve Yer Karmaşıklığı

```
COLOR* RotateTheRow(COLOR* row,int N){  
    int i;  
    COLOR tmp = row[N-1];  
    for(i = N-1 ; i > 0 ; i--)  
        row[i] = row[i-1];  
  
    row[0] = tmp;  
    return row;  
}
```

Zaman : $O(N)$

Yer : $O(N)$

IsSafe() Fonksiyonunun Zaman ve Yer Karmaşıklığı

```
int IsSafe(COLOR** matrix, int row, int N){  
    int i,j;  
    for(i = 0; i<row ; i++){  
        for(j = 0 ; j<N ; j++){  
            if(matrix[row][j].colorNo == matrix[i][j].colorNo)  
                return 0;  
        }  
    }  
    return 1;  
}
```

Zaman : $O(N^2)$

Yer : $O(N^2)$

Solve() Fonksiyonunun Zaman ve Yer Karmaşıklığı

```
int Solve(COLOR** matrix, int N, int row, int mode){
    int i;
    for(i=0; i<N; i++){
        if(IsSafe(matrix, row, N)){ // IF IT IS SAFE
            if(row!=N-1){
                if(Solve(matrix, N, row+1, mode))
                    return 1;
            }
            else{
                matrix[row]=RotateTheRow(matrix[row], N);
                if(mode==2)
                    PrintMatrix(matrix, N);
            }
        }
        else{
            return 1; // IF IT IS NOT SAFE
        }
        matrix[row]=RotateTheRow(matrix[row], N);
        if(mode==2)
            PrintMatrix(matrix, N);
    }
}
return 0;
}
```

Zaman : $O(N!)$

Yer : $O(N^2)$

VIDEO LINKİ:

<https://youtu.be/LyNcWriZu9Y>