**TÜRKİYE CUMHURİYETİ**
**YILDIZ TEKNİK ÜNİVERSİTESİ**
**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**


# ALGORİTMA ANALİZİ
# PROJE

**Öğrenci No:** 20011055
**Öğrenci Adı Soyadı:** Şeymanur KORKMAZ
**Öğrenci e-posta:** seymanur.korkmaz@std.yildiz.edu.tr

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

Ders Yürütücüsü:
Mine Elif KARSLIGİL
**Ocak, 2023**

# Table of Contents

# 1. YÖNTEM

## PROBLEM

Verilen N*N'lik oyun tahtasında N adet renk her satırda farklı sıra ile yer almaktadır. Oyun tahtasını backtracking yöntemi kullanarak rekürsif bir şekilde, her sütunda her renkten sadece 1 tane olacak şekilde düzenlememiz istenmektedir.

## ÇÖZÜM

Backtracking yöntemi kullanılarak her bir satır için şart sağlanana kadar rotate işlemi yürütülür. Satırların her biri için tüm ihtimaller denendikten sonra oyun tahtasının hala düzenlenemiş olma ihtimali vardır. Bu durumda kullanıcıya tahtanın düzenlenemediğine dair bilgi verilir.

- **void copyMatrix(int **matrix1, int **matrix2, int N):**
  Matrisin başlangıçtaki halini kaybetmemek adına, *matrix1* isimli dizi *matrix2* isimli yeni bir matrise kopyalanıp yapılacak olan işlemler bu matrisin üzerinde yapılır.

- **int colorize(char **colors, char *color, int *N):**
  Kullanıcının matrisin her bir indisi için girdiği renk öncelikle *'colors'* dizisinde mevcut mu diye bakılır. Eğer renk mevcut ise matrisin o andaki indisine, rengin *'colors'* dizisindeki indisi yerleştirilir. Eğer renk mevcut değilse '*colors'* dizisine yerleştirilir ve dizinin son indisi matrisin ilgili gözüne yazılır.

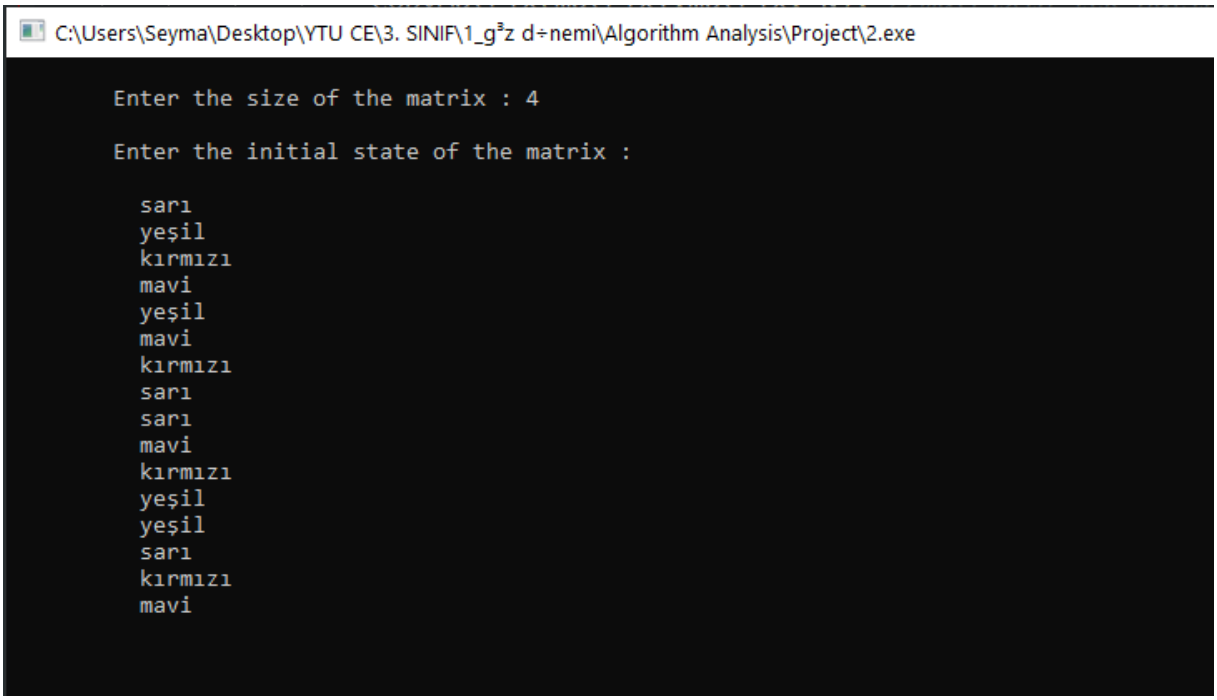
- **void rotate(int **matrix, int rotate_row, int N):**
  Matrisin 'rotate_*row'* indisindeki satırı 1 defa sağa kaydırılır. Örneğin sıralama mavi, sarı, yeşil şeklindeyse *'rotate'* fonksiyonu ardından yeşil, mavi, sarı haline gelir.

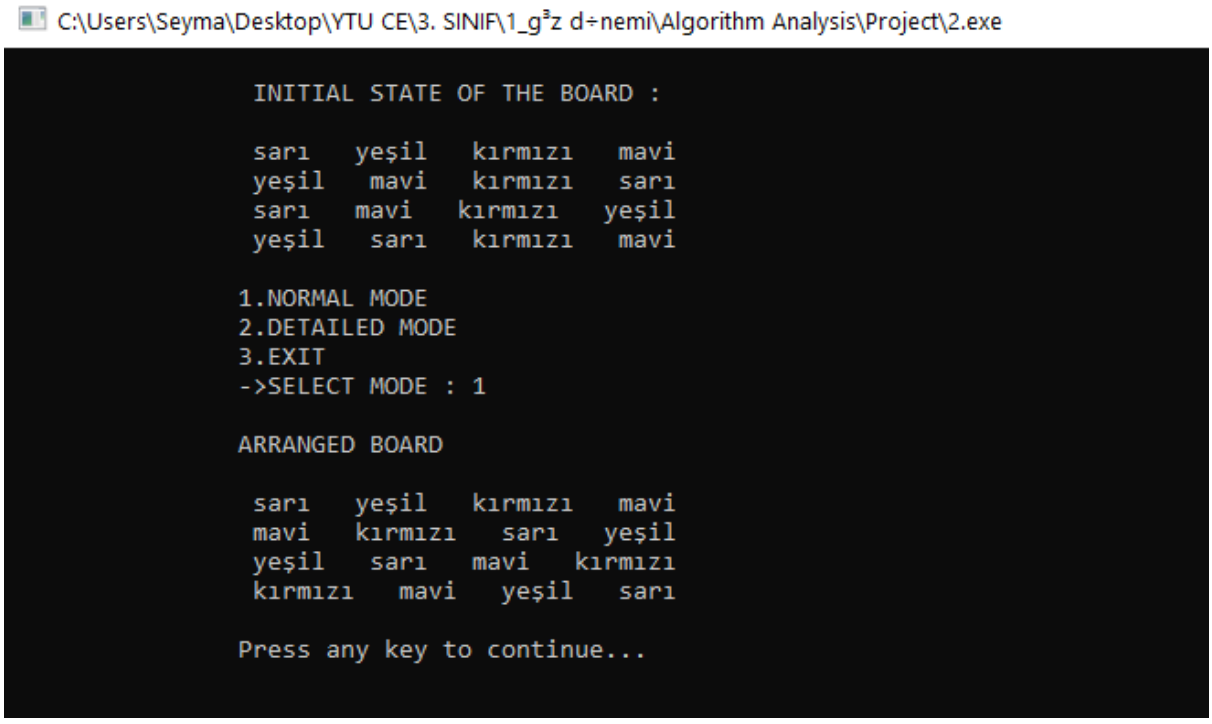
- **int backTracking(int **matrix, int i, int N, char **colors, int mode):**
  Mevcut satır için (i. Satır) 'isValid' fonksiyonu çağrılır, eğer sonuç 1 ise yani bulunduğumuz satıra kadar olan satırlar için şart sağlanıyorsa bir sonraki satır için '*backTracking'* fonksiyonu çağrılır. Bu şekilde son satıra kadar gidilirse tahta düzenlenmiş demektir. Eğer '*isValid'* fonksiyonundan 0 değeri dönerse mevcut satır rotate edilerek yeniden '*backTracking'* fonksiyonu çağrılır. Rotate işlemi N defa tekrarlandıktan sonra işlem başarısız demektir.

- **int isValid(int **matrix, int N, int row):**
  Mevcut satıra kadar (row. satır) olan satırlar için her sütunda bir renkten yalnızca 1 adet olması şartının sağlanma durumu kontrol edilir.

# 2. UYGULAMA

## NORMAL MOD



C:\Users\Seyma\Desktop\YTU CE\3. SINIF\1_g³z d÷nemi\Algorithm Analysis\Project\2.exe

```
Enter the size of the matrix : 3

Enter the initial state of the matrix :

  sarı
  siyah
  kırmızı
  sarı
  kırmızı
  siyah
  siyah
  sarı
  kırmızı
```

C:\Users\Seyma\Desktop\YTU CE\3. SINIF\1_g³z d÷nemi\Algorithm Analysis\Project\2.exe

```
          INITIAL STATE OF THE BOARD :

        sarı    siyah   kırmızı
        sarı    kırmızı siyah
        siyah   sarı    kırmızı

        1.NORMAL MODE
        2.DETAILED MODE
        3.EXIT
        ->SELECT MODE : 1

        Arrange failed.

        Press any key to continue...
```

## DETAYLI MOD

C:\Users\Seyma\Desktop\YTU CE\3. SINIF\1_g³z d÷nemi\Algorithm Analysis\Project\2.exe

```
 INITIAL STATE OF THE BOARD :

kırmızı mavi    yeşil
yeşil   mavi    kırmızı
kırmızı yeşil   mavi

1.NORMAL MODE
2.DETAILED MODE
3.EXIT
->SELECT MODE : 2


kırmızı mavi    yeşil
yeşil   mavi    kırmızı
kırmızı yeşil   mavi


kırmızı mavi    yeşil
kırmızı yeşil   mavi
kırmızı yeşil   mavi


kırmızı mavi    yeşil
mavi    kırmızı yeşil
kırmızı yeşil   mavi


kırmızı mavi    yeşil
yeşil   mavi    kırmızı
kırmızı yeşil   mavi

Arrange failed.

Press any key to continue...
```

C:\Users\Seyma\Desktop\YTU CE\3. SINIF\1_g³z d÷nemi\Algorithm Analysis\Project\20011055_2.exe

```
Enter the size of the matrix : 3

Enter the initial state of the matrix :

  mor
  turuncu
  pembe
  mor
  turuncu
  pembe
  mor
  turuncu
  pembe
```

C:\Users\Seyma\Desktop\YTU CE\3. SINIF\1_g³z d÷nemi\Algorithm Analysis\Project\2.exe

```
          INITIAL STATE OF THE BOARD :

        mor     turuncu pembe
        mor     turuncu pembe
        mor     turuncu pembe

        1.NORMAL MODE
        2.DETAILED MODE
        3.EXIT
        ->SELECT MODE : 2


        mor     turuncu pembe
        mor     turuncu pembe
        mor     turuncu pembe


        mor     turuncu pembe
        pembe   mor     turuncu
        mor     turuncu pembe


        mor     turuncu pembe
        pembe   mor     turuncu
        turuncu pembe   mor

        ARRANGED BOARD

        mor     turuncu pembe
        pembe   mor     turuncu
        turuncu pembe   mor

        Press any key to continue...
```

## 3. SONUÇ

### isValid Fonksiyonu Karmaşıklığı

```
int isValid(int **matrix, int N, int row) {

    int i,j;

    for (i=0; i < row; i++) {
        for(j=0 ; j < N; j++){
            if(matrix[row][j] == matrix[i][j]){
                return 0;
            }
        }
    }

    return 1;
}
```

İç içe 2 adet for döngüsü bulunmaktadır. Bundan dolayı fonksiyonun zaman karmaşıklığı $O(N^2)$'dir.

Bu fonksiyonun yer karmaşıklığı sadece parametre olarak verilen "*matrix*" matrisinin büyüklüğüne bağlıdır ve bu büyüklük $O(N^2)$'dir.

## rotate Fonksiyonu Karmaşıklığı

```
void rotate(int **matrix, int rotate_row, int N){ //saga kaydirma

    int temp = matrix[rotate_row][N-1];
    int i;

    for (i = N-1; i >0 ; i--) {
        matrix[rotate_row][i] = matrix[rotate_row][i-1];
    }

    matrix[rotate_row][0] = temp;

}
```

0'dan N'e kadar arası for döngüsünden dolayı bu fonksiyonun zaman karmaşıklığı O(N)'dir.

Bu fonksiyonun yer karmaşıklığı parametre olarak verilen "*matrix*" matrisinin büyüklüğüne bağlıdır. Fonksiyon içerisinde temp ve i integer'ları harici bellek kullanılmamıştır. Bu nedenle yer karmaşıklığı $O(N^2)$'dir.

## colorize Fonksiyonu Karmaşıklığı

```
int colorize(char **colors, char *color, int *N){

    int i;

    for(i=0;i<*N;i++){
        printf("%s",color);
        if(strcmp(colors[i],color) == 0){
            return i;
        }
    }
    (*N)++;
    strcpy(colors[(*N)-1], color);
    printf("%s",colors[(*N)-1]);
    return (*N)-1;

}
```

0'dan N'e kadar arası for döngüsünden dolayı bu fonksiyonun zaman karmaşıklığı O(N)'dir.

Bu fonksiyonun karmaşıklığı parametre olarak verilen '*colors*' isimli char matrisine bağlıdır. Bunun haricinde fonksiyon içerisinde integer i harici bellek kullanılmamıştır. Bu nedenle yer karmaşıklığı $O(N^2)$'dir.

## backTracking Fonksiyonu Karmaşıklığı

```
int backTracking(int **matrix, int i, int N, char **colors, int mode){
    int k;

    for(k = 0; k<N; k++){
        if(isValid(matrix, N, i)){  // mevcut satıra kadar olan satirlar sarti sagl
            if (i == N-1) {  //backTracking ile son satira kadar gidildiyse tahta c
                return 1;
            }else {
                if(backTracking(matrix,(i + 1),N,colors,mode)){
                    return 1; //backTracking ile son satira kadar gidildiyse tahta
                }else{
                    rotate(matrix, i, N); //mevcut satira kaar olan satirlar sarti
                }
            }
        }else{
            rotate(matrix, i, N); //mevcut satira kaar olan satirlar sarti saglamı
            if(mode == 2){      // detay modda tüm adımların tek tek yazdırılması m
                printMatrix(matrix,colors,N);
            }
        }
    }
    return 0;

}
```

Fonksiyon içinde yer alan '*backTracking*' fonksiyonu , recursive olarak çağrılmaktadır. Bu nedenle, bu fonksiyonun zaman karmaşıklığı, recursive olarak çağrılma sayısına da bağlıdır. Bu fonksiyon recursive olarak en az 1 kez ve en fazla N kez çağrılabilir. Bu nedenle, zaman karmaşıklığı O(N!) olacaktır.

Bu fonksiyonun zaman karmaşıklığı, '*isValid*' fonksiyonunun ve '*rotate*' fonksiyonunun zaman karmaşıklığını da içerir. isValid fonksiyonunun karmaşıklığını $O(N^2)$, 'rotate' fonksiyonunun zaman karmaşıklığını O(N) bulmuştuk. Bu nedenle bu fonksiyonun karmaşıklığı $O(N^3*N!)$ olacaktır.

Bu fonksiyonun karmaşıklığı parametre olarak verilen '*matrix*' ve '*colors*' isimli matrisleri bağlıdır. Bu matrisler haricinde fonksiyon içerisinde integer k harici bellek kullanılmamıştır. Bunlar da çok küçük belleklerdir. Bu nedenle yer karmaşıklığı $O(N^2)$'dir.

## 4. VİDEO LİNKİ

https://youtu.be/wgHCEcIeBzY