TÜRKİYE CUMHURİYETİ

YILDIZ TEKNİK ÜNİVERSİTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

Öğrenci No: 20011620

Öğrenci Adı Soyadı: Esma Nur Ekmekci

Öğrenci e-posta: nur.ekmekci@std.yildiz.edu.tr

DERS: Algoritma Analizi

PROJE

YÖNTEM

Temelde 3 fonksiyonum var. O satırı döndürüyorum ve kontrol ediyorum şartımı (aynı sütunda aynı değer denk gelmeyecek). Eğer şartım sağlanırsa sonraki satırdan kontrollerime devam ediyorum.

- ✓ ***rotateRight*** fonksiyonum verilen satırdaki değerleri bir tur sağa kaydıracak (sondaki de başa geçecek).
- ✓ ***check*** fonksiyonum verilen satırın o anki konumunun uygun olup olmadığını kontrol ediyor. Verilen satır, bir üstündeki satırdan başlanarak sıfırıncı satıra kadar değerleri karşılaştırılır. Aynı değer gördüğü an sıfır döndürür ve çağrıldığı yerde bu değer if içinde kullanılır.
- ✓ ***backtrack*** fonksiyonum *recursive* bir fonksiyondur ve durma koşulum o anki satır değerimin N değerine ulaşması. Eğer kontrol ede ede gittiğimiz o satır N'e ulaşmışsa gerekli koşulları sağlamış demektir. Bir satır en fazla N defa döner bundan bir fazlası aynı şeylerin tekrarıdır artık. For döngüm içinde önce o anki bir sağa döndürüyorum ve sonucunda *check* fonksiyonuma gönderiyorum eğer şartları sağlamışsa *backtrack* fonksiyonumu bu sefer bir sonraki satır değeri için çağırıyorum.

VİDEO LİNKİ: https://youtu.be/_5b1xNdh20Y

UYGULAMA

```
        BACKTRACK PROBLEM
How many colour do you have?(2<n<9) 4
Enter your colours number side by side with spaces.
Your colours are starting 0 and goes to 3
Row 1: 0 1 2 3
Row 2: 1 3 2 0
Row 3: 0 3 2 1
Row 4: 1 0 2 3

Your matrix
-------------
0 1 2 3
1 3 2 0
0 3 2 1
1 0 2 3
1->Normal Mode
2->Detailed mode
Your choice: 2
Your matrix
-------------
0 1 2 3
0 1 3 2
0 3 2 1
1 0 2 3

Your matrix
-------------
0 1 2 3
2 0 1 3
0 3 2 1
1 0 2 3

Your matrix
-------------
0 1 2 3
3 2 0 1
0 3 2 1
1 0 2 3

Solved!

Your matrix
-------------
0 1 2 3
3 2 0 1
1 0 3 2
2 3 1 0
```

```
        BACKTRACK PROBLEM
How many colour do you have?(2<n<9) 3
Enter your colours number side by side with spaces.
Your colours are starting 0 and goes to 2
Row 1: 0 2 1
Row 2: 2 1 0
Row 3: 0 2 1

Your matrix
-------------
0 2 1
2 1 0
0 2 1
1->Normal Mode
2->Detailed mode
Your choice: 1

Solved!

Your matrix
-------------
0 2 1
1 0 2
2 1 0
```

```
How many colour do you have?(2<n<9) 4
Enter your colours number side by side with spaces.
Your colours are starting 0 and goes to 3
Row 1: 1 2 3 4
Row 2: 2 3 4 1
Row 3: 4 3 2 1
Row 4: 3 2 1 4

Your matrix
-------------
1 2 3 4
2 3 4 1
4 3 2 1
3 2 1 4
1->Normal Mode
2->Detailed mode
Your choice: 1

Solved!

Your matrix
-------------
1 2 3 4
3 4 1 2
2 1 4 3
4 3 2 1
```

Kullandığım Matris:

1 2 3 4

2 3 4 1

4 3 2 1

3 2 1 4

SONUÇ

Yer Karmaşıklığı:

```
printf("How many colour do you have?(2<n<9) ");
scanf("%d", &N);
A = (int**) malloc(N * sizeof(int*));
```

Dinamik bir bellek yapısında tuttum matris değerlerimi.

NxN'lik bir matris üzerinde çalışma yapıyoruz. Yer karmaşıklığımız $O(N^2)$

Zaman Karmaşıklığı:

```
void printMatrix(int** A,int N){
    int i,j;
    printf("\nYour matrix\n------------\n");
    for (i=0; i < N; i++){
        int *R = A[i];
        for(j=0; j<N; j++){
            printf("%d ",R[j]);
        }
        printf("\n");
    }
}
```

printMatrix: $O(N^2)$

```
void rotateRight(int** A, int N,int row){
    int i, tmp;
    int* R = A[row];
    tmp = R[N - 1]; //last element goes to head

    for(i=N-1; i>0; i--){
        R[i] = R[i-1];
    }
    R[0] = tmp;
}
```

rotateRight: $O(N)$

```
int check(int** A, int N,int row){
    int i,j;
    int* R = A[row];
    int* R2;
    j = 0;
    for(i = row-1; i>=0; i--){
        R2 = A[i]; //the row which
        //we check every number at
        for(j=0 ; j < N; j++) {
            if (R[j] == R2[j])
                return 0;
        }
    }
    return 1; // 1 means check
}
```

check: $O(N^2)$

```
int backtrack(int** A, int N, int row, int mode){
    int result;
    int* R = A[row];
    int i = 0;

    if (N == row) //it means we alredy checked all
        return 1;
    //we can rotate at most N times and if there i
    for(i; i<N; i++){
        rotateRight(A,N,row);
        if(mode == 2){
            printMatrix(A,N);
        }
        if (check(A, N, row)) {
            result = backtrack(A,N,row+1,mode); //

            if (result == 1)
                return 1;
        }
    }
    return 0;
}
```

backtrack: $O(N^N)$

Her satırımızın N farklı kombinasyonu olabilir ve bir sonraki satır mesela duruma dahil edildiğinde bu ikisi birlikte N*N farklı kombinasyon oluşturur. N satırımız var yani zaman karmaşıklığımız $O(N^N)$ olarak değerlendirilir.