Ders Adı: Algoritma Analizi

Öğrenci Bilgileri: Ömer Buğrahan Çalışkan – 17011076

Ödev İçeriği: NxN'lik bir matris görünümündeki oyun tahtasında her satırda aynı N renk farklı sıra ile yer almaktadır. Bir satırdaki renklerin sıralanışı, renkler sağa doğru kaydırılarak değiştirilebilmektedir. Örneğin satırdaki renkler sırası ile kırmızı, mavi, yeşil, mor ise satır 1 defa sağa kaydırıldığında yeni sıralama mor, kırmızı, mavi, yeşil olur. Bir defa daha sağa kaydırılırsa yeşil, mor, kırmızı, mavi elde edilir. Sonuç matrisinde her sütunda her renkten sadece 1 tane olacak şekilde satırları geri-izleme(backtracking) yöntemi ile rekürsif olarak düzenleyen algoritmayı tasarlayınız.

Kod Açıklaması:

shift_right Fonksiyonu: Matrix, n değeri ve kaydırılacak satır parametrelerini alır. Temp değeri oluşturularak matrisin belirtilen satırındaki son eleman atanır. Ardından kalan elemanlar döngü içerisinde sırayla bir sonraki elemanın olduğu yere yazılır. En son ilk başta tempe atadığımız değer sağa kaydırma işlemi yaptığımız için belirtilen satırın 0. indexindeki elemana atanır.

Col_control Fonksiyonu: Matrix, Kontrol edilecek satır ve sütun parametreleri alınır. Matrix belirtilen satırın üstündeki bütün satırlardaki aynı sütunlarda çakışma var mı diye kontrol eder. Çakışma var ise true yok ise false döndürür.

Total_shift_calculator Fonksiyonu: Programda oluşabilecek maksimum shift sayısı hesaplanır ve döndürülür.

solveMatrix Fonksiyonu: Matix, colors(renklerin olduğu dizi),n sayısı, belirtilen satır, toplam shift sayısı ve secim parametrelerini alır. Program çalıştırıldığında eğer yapılabilecek maksimum shift sayısına ulaşıldıysa Sonuc Yok gösterilir ve program kapatılır. While döngüsünde col değişkeni satırdaki sütun indexini temsil eder, flag ise yapılan shift sayısının n değerinden büyük olup olmadığını kontrol eder. Eğer aynı sütunda çakışan değerler var ise shift_count değişkeni arttırılır, sütun değişkeni sıfırlanır çünkü kaydırılan satır için tekrar sıfırdan kontrol edilmesi gerekecek ve satır bir sağa kaydırılır, son olarak total_shift değişkenimiz de bir arttırılır. Eğer sütunda herhangi bir çakışma yok ise col++ yazılarak aynı satırdaki diğer elemana geçilir.

Döngüden çıkıldığında yapılan shift sayısı total_shift sayısına eklenir.

Döngüden nasıl çıkıldığını kontrol etmek için flag değişkenine bakıyoruz eğer shift sayısı n değerinden fazla olduğu için çıkıldıysa ve ilk satırda değil ise üst satır bir sağa kaydırılır ve bu satırın tekrar kontrol edilmesi için rekürsif olarak çağrılır. Eğer en üst satırdaysak (zaten 0. Satır kontrolüne gerek yok çünkü altındakiler onu referans alarak sıralanabilir.) sonuç olmadığı mesajı gösterilir.

Eğer döngüden belirtilen satırdaki bütün sütunlar bittiği için çıkıldıysa ve en aşağıdaki satırda değil ise bir alttaki satıra geçilir eğer en alt satırdaysak return verilerek döngü sonlandırılır.

PrintMatrix Fonksiyonu: Matrix, colors, n değeri, i değeri(yazdırılacak satır) ve seçim parametrelerini alır. Matrix yazdırılır ve secime bağlı olarak satir tamamlandiginda tamalanan satırın goruntusunu gösterir.

Get_Matrix Fonksiyonu: Matrix, colors ve n değerini alır. Kullanıcıdan renk dizisi içerisinden girdiği n sayısı kadar renk girmesi istenir ve girdiği renklerin dizide olmasının kontrolü yapılır. Ardından renk matrisi girişi istenir ve girilen her rengin seçilen renklerde olup olmadığının kontrolü yapılır. Eğer eşleşir ise renk kolaylık açısından sayı matrisine çevrilir.

Main Fonksiyonu: Renk matrisi açılır ve kullanıcıya yazdırılır. Kullanıcıdan N sayısı, ardından çıktı tipi istenir. Olusan matris ekrana yazdırılır. solveMatrix fonksiyonuyla çözülür ve sonuç matrisi ekranda gözükür.

Not:

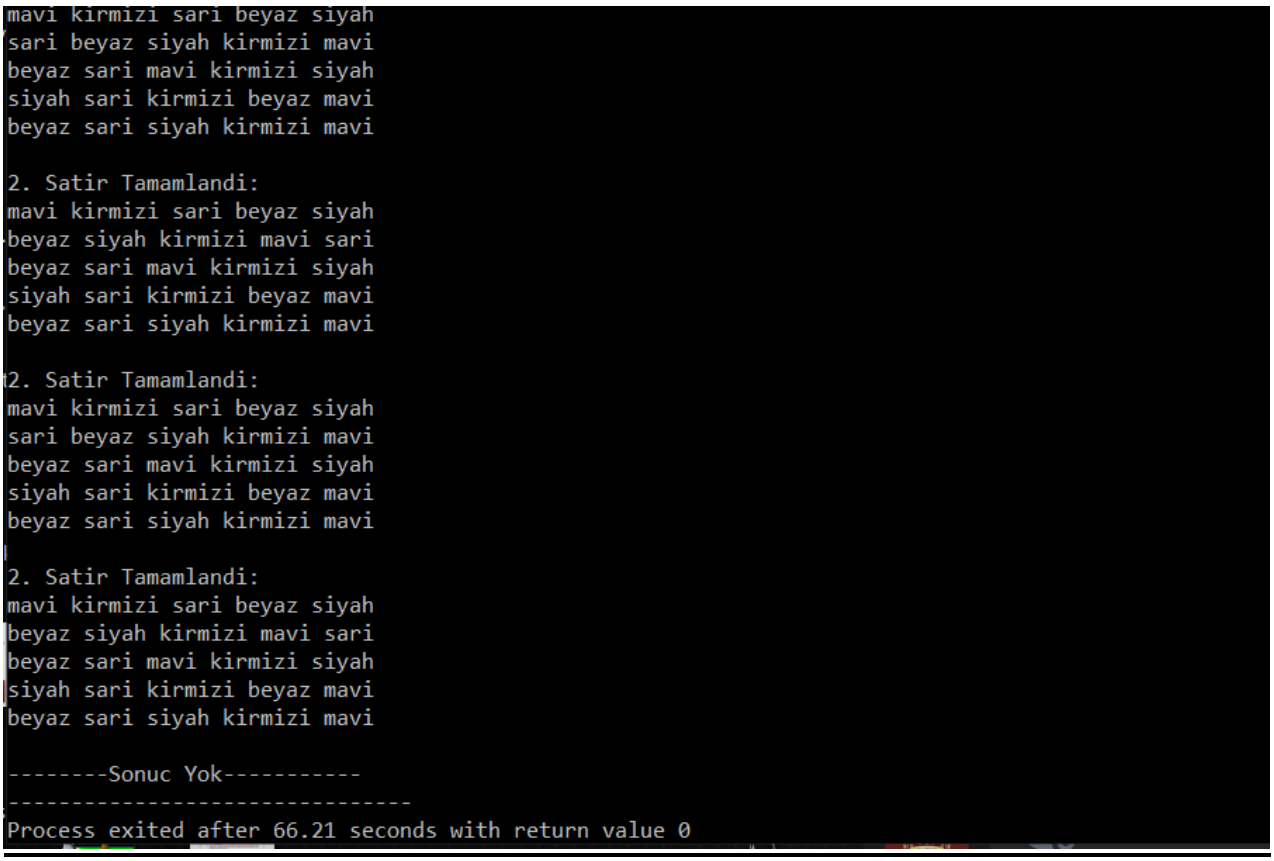
0. Satırın kontrolü gerekmediği için fonksiyona 1. Satir parametresiyle gönderildi.

Sonuç Çıktısı aynı zamanda aşamalı adımın son adımını temsil etmektedir.

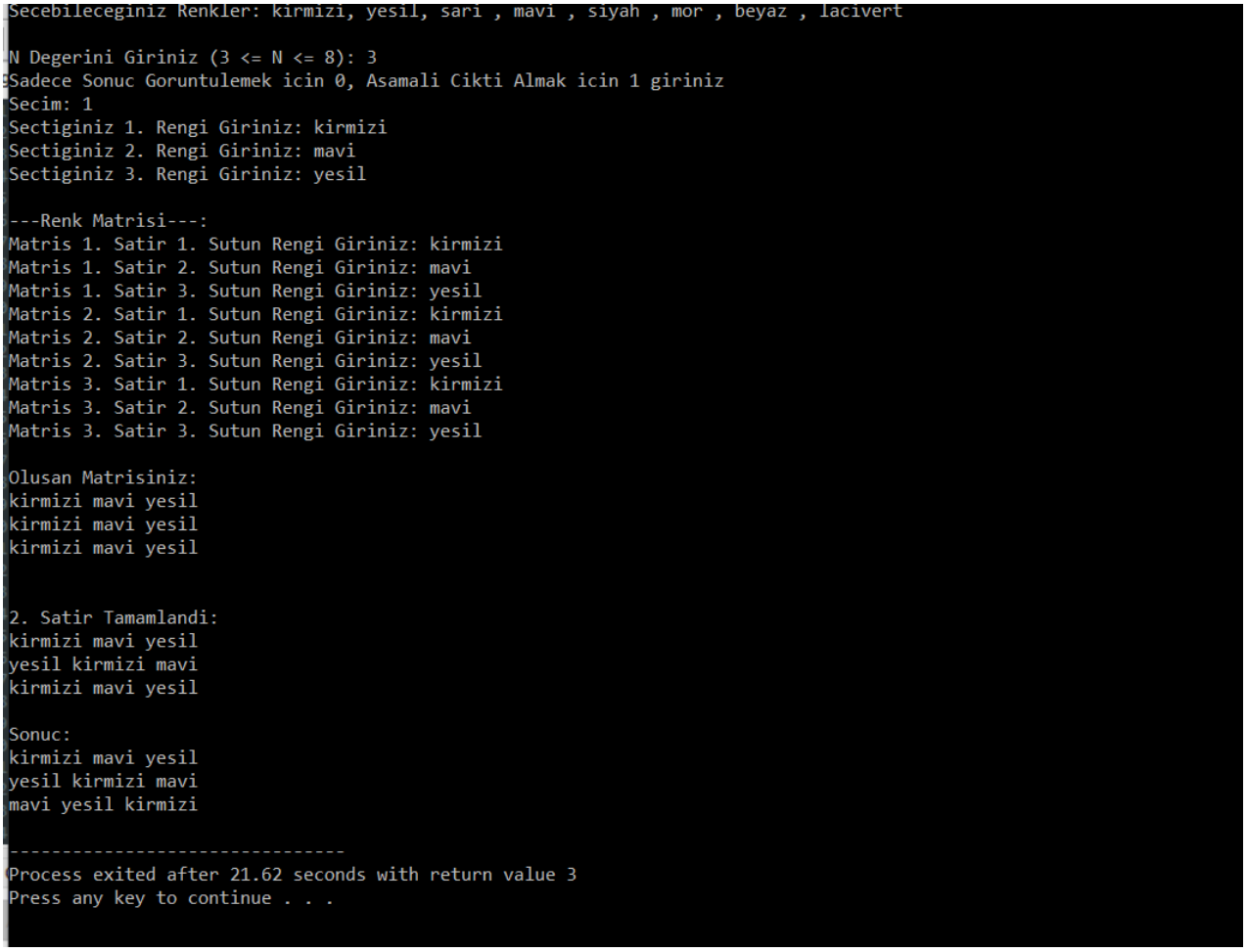
Ekran Görüntüleri:

4x4 Matrisin Aşama Aşama Sonuç Bulunduğu Görüntü. (Sonuç matrisi aynı zamanda son satırın da tamamlandığı adımdır.)

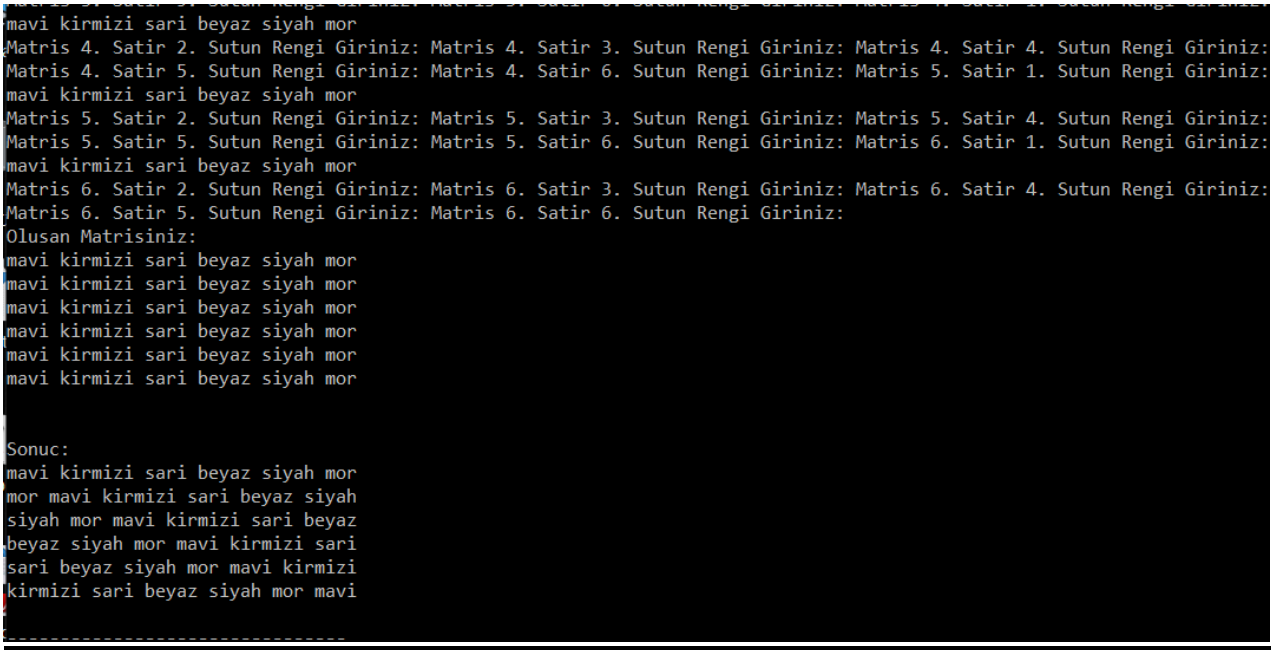
5x5 Matrisin Aşama aşama denenmesi fakat sadece 2. Satirin tamamlanabilmesi kalan satırların tamamlanaması durumunda sonuç yok mesajının görünmesi.



3x3 Matrisin Aşama Aşama Tamamlanması ve Girdi Biçiminin Gösterimi



6x6 Matrisin Sadece Çıkan Doğru Sonucu Göstermesi



C Kodu:

```
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>

void printMatrix(int **matrix, char *colors[8], int n, int i, int secim);

void shift_right(int **matrix, int n, int row){
        int i;
        int temp = matrix[row][n-1];
   for(i=n-1;i>0;i--){                  // Satir sağa kaydırılır
      matrix[row][i]=matrix[row][i-1];
   }
   matrix[row][0]=temp;
}

bool col_control(int **matrix, int row, int col){
        int i,j;

        for(i=0;i<row;i++){ // Matrisin belirtilen satırının üstündeki bütün sütunların aynı
olup olmadığının kontrolü
                        if(matrix[row][col]==matrix[i][col])
                                return true;
        }
        return false;
}

int total_shift_calculator(int n){
        int i, sum=0;           // Toplam olabilecek maksimum shift sayısı hesaplanır
        for(i=1;i<n;i++)
                sum += pow(n,i);
        return sum;
}

void solveMatrix(int **matrix, char *colors[8], int n, int row, int total_shift, int secim){
        int col = 0;
        int shift_count = 0;
        bool flag = true;
```

```
	if(total_shift >= total_shift_calculator(n)){		// Maks shift sayısı geçilirse program sonlanır
		printf("\n--------Sonuc Yok-----------");
		exit(0);
	}

	while( col<n && flag){	// sütun ve satir icin olabilecek maksimum shift sayısı kontrolü
		if( col_control(matrix,row,col)){
			if(shift_count < n){ // Eğer shift sayısı n değerini geçerse farklı satır denenecek
				shift_count++;
				col = 0;
				shift_right(matrix,n,row);
			}
			else flag = false;
		}
		else col++;	// Eşleşme olmadıysa sütunda ilerlenecek.
	}
	total_shift += shift_count;  // Tüm satırlar için toplam shift sayısı tutuldu
	if(!flag){
		if(row != 1){ // İlk satırda değil ise ve shift sayısı asıldıysa bir üstteki satır shift edilerek tekrar kontrol edilir.
			shift_right(matrix,n,row-1);
			return solveMatrix(matrix,colors,n,row-1,total_shift,secim);
		}
		else{	// İlk satıra kadar gelindiyse sonuc yoktur
			printf("\n--------Sonuc Yok-----------");
			exit(0);
		}
	}
	else{
		if(row != n-1){		// Son satırda değil isek tamamlanan satırın altına geçilir ve secim 1 ise tamamlanan satır ekrana bastırılır.
			if(secim==1) printMatrix(matrix,colors,n,row+1,secim);
			return solveMatrix(matrix,colors,n,row+1,total_shift,secim);
		}
		else return;
	}
}

void printMatrix(int **matrix, char *colors[8], int n, int i, int secim){
```

```
	int row,col;		// Matris ekrana yazılır, eger secim 1 ise tamamlanan satırın bilgisi de gösterilir.
	if(secim==1 && i != 0){
		printf("\n%d. Satir Tamamlandi:\n",i);
	}
	for (row = 0; row < n; row++) {
		for (col = 0; col < n; col++)
			printf("%s ", colors[matrix[row][col]]);
		printf("\n");
	}
}

void get_Matrix(int **matrix, char *colors[8], int n){

	int i,j,k,z;
	char get_color[10];
	char selected_colors[n][10];
	int flag = 0;
	char temp[10];
	int flag2 = 0;

	for(i=0;i<n;i++){	// Kullanıcıdan girilen rengin renk dizisinde olup olmamasına göre alımı
		printf("Sectiginiz %d. Rengi Giriniz: ",i+1);
		scanf("%s",temp);
		flag2 = 0;
		for(j=0;j<8;j++){
			if(strcmp(temp,colors[j])==0){
				flag2 = 1;
			}
		}

		if(flag2==1){
			strcpy(selected_colors[i],temp);
			}
		else{
			printf("Lutfen dogru renk giriniz\n");
			i--;
		}
	}

	printf("\n---Renk Matrisi---:\n");
```

```
	for(i=0;i<n;i++){	// // Kullanıcıdan girilen rengin seçilen renk dizisinde olup olmamasına göre alımı
		for(j=0;j<n;j++){
			printf("Matris %d. Satir %d. Sutun Rengi Giriniz: ",i+1,j+1);
			scanf("%s",get_color);
			flag = 0;
			for(z=0;z<n;z++){
				if(strcmp(get_color,selected_colors[z])==0){
					flag = 1;
				}
			}

			if(flag == 1){
				for(k=0;k<8;k++){
					if(strcmp(get_color,colors[k])==0){
						matrix[i][j] = k;
						break;
					}
				}
			}
			else{
				printf("Lutfen Sectiginiz Renklerden Birini Giriniz...\n");
				j--;
			}
		}
	}
}

int main(){

	int n,i,secim;

	char *colors[8] = {"kirmizi","yesil","sari","mavi","siyah","mor","beyaz","lacivert"};
	printf("Secebileceginiz Renkler: kirmizi, yesil, sari , mavi , siyah , mor , beyaz , lacivert\n\n");
	printf("N Degerini Giriniz (3 <= N <= 8): ");
	scanf("%d",&n);

	if(n<3 || n>8){
		printf("Hatali deger girdiniz, program sonlanacak...");
		return 0;
	}
```

```
	printf("Sadece Sonuc Goruntulemek icin 0, Asamali Cikti Almak icin 1 giriniz\nSecim: ");
	scanf("%d",&secim);

	if(secim!=0 && secim!=1){
		printf("Hatali deger girdiniz, program sonlanacak...");
		return 0;
	}

	int **matrix;
	matrix = (int**)malloc(sizeof(int*)*n);
	for(i=0;i<n;i++)
		matrix[i] = (int*)malloc(sizeof(int)*n);

	int total_shift = 0;
	get_Matrix(matrix,colors,n);
	printf("\nOlusan Matrisiniz:\n");
	printMatrix(matrix,colors,n,0,secim);
	printf("\n");
	solveMatrix(matrix,colors,n,1,total_shift,secim);
		printf("\nSonuc: \n");
	printMatrix(matrix,colors,n,0,secim);

}
```