Algoritma Analizi Dersi 3.Ödev Raporu

Backtracking(Geri izleme) Algoritması Uygulaması

Bu uygulamada verilen renk değerleri için bir matris oluşturulmuş ve her bir sütunda renk değerlerinin bir kez tekrar etmesi sağlanacak şekilde düzenleme yapan rekürsif bir backtracking algoritması tasarlanmıştır.

M.Yasin SAĞLAM

15011804

ALGORİTMA ANALİZİ

GRUP 1

Algoritma Analizi Dersi 3.Ödev Raporu

Backtracking(Geri izleme) Algoritması Uygulaması

Yöntem

Verilen problemde kullanıcıdan NxN lik bir renk matrisi kullanıcıdan sayısal değerler olarak Sarı=1, mavi=2, yeşil=3, mor=4, beyaz=5, siyah=6, kırmızı=7, gri=8 olacak şekilde okunarak, her bir rengin bulunduğu sütun bazında bir kez tekrar edilmesi sağlanacak şekilde düzenlenmesinin rekürsif bir backtracking algoritmasıyla gerçekleştirilmesi beklenmektedir. Çıktı olarak sonuç ve işlem adımları olarak iki farklı şekilde gözlem yapılmasına imkan verecek bir algoritma tasarlanmıştır. Problem aşağıda belirtilen ve açıklanan fonksiyonlar ile modüler olarak çözülmüştür. Program menüsü 3 adet opsiyon içermektedir ve kullanıcı çıkış yapana kadar çalışmaktadır.

Programın Menüsü İçin Input Açıklamaları:

1-Kullanıcıdan matris boyutu alınır, matris oluşturulur, doldurtulur ve çözülür.

2-Boyut sabit kalır, matris yeniden doldurulur ve çözülür.

3-Yeni bir Matris boyutu alınır, matris oluşturulur, doldurtulur ve çözülür.

0- Programdan çıkmayı sağlar.

1-) void printMatrix(int **arr,int row,int n) → Verilen integer değerlerden oluşan matrisin belirtilen satırına kadar olan içeriğini ekrana yazdıran fonksiyondur.

2-) void printColor(int **arr,int row,int n)→ Verilen integer değerlerden oluşan matrisin belirtilen satırına kadar renk karşılıklarını ekrana yazdıran fonksiyondur.

3-) void shifteR(int *arr,int n) → Verilen n elemanlı diziyi 1 kez sağa shift eden fonksiyondur.

4-) int control(int **arr,int row,int n) → Renklerin integer karşılıklarını içeren matrisin verilen satırındaki değerlerin istenilen duruma(**her sütunda her renkten sadece 1 tane olacak şekilde**) uygunluğunu kontrol eden uygunsa 1 değilse 0 döndüren fonksiyondur.

5-) int solve(int **arr,int row,int n) → 3 ve 4 numaralı fonksiyonları kullanarak istenilen duruma rekürsif olarak backtracking yaparak ulaşmaya çalışan, ulaşıldıysa durumu ekrana yazdıran ve shift edilen satır ve kaçıncı kez shift işlemi yapıldığını ekrana yazdıran fonksiyondur. Fonksiyon her adımda sadece bulunduğu satırın üstündeki satırları kontrol eder ve bir satırda (n-1) defa shift yapıldı ve doğru sonuca ulaşılamadıysa backtrace ile bir üst satıra geçer.

Fonksiyonun C kodu:

```
int solve(int **arr,int row,int n){
    int i;
    if(row<n){ //eger bakilan satir eleman sayisindan kucukse
        for(i=0;i<n;i++){ //eleman sayisi kadar
            if(control(arr,row,n)){ //varilen satiri kontrol et
                solve(arr,row+1,n); //fisible ise bir sonraki satira gec
            }
            if(i<n-1){ //fisible degil ise ve n-1 defa ile sinirli olarak
                //printf("\n\nRow : %d Shift number : %d \n",row+1,i+1);  //kontrol amacli yaz
dirma
                shifteR(arr[row],n); //ilgili satiri 1 saga kaydir.
                //printMatrix(arr,row,n);  //kontrol amacli yazdirma
            }
        }
    }
    else{ //degilse
        row--; //yazdirmada bir sorun cikmamasi icin row u 1 azalt
        printf("\n\nSOLVED !!!\nSolution is : ");
        //printMatrix(arr,row,n);
        printColor(arr,row,n); //bulunan cozumu ekrana renk olarak yazdir
        return 1; // 1 dondur
    }
return 0; //cozum bulunamazsa 0 dondur
}
```

6-) int **Allocator(int size) → Verilen boyut kadar bir integer matris allocate eden ve matrisi dışarı döndüren fonksiyondur.

7-) void ReAllocator(int ***matrix,int size) → Verilen integer matrisi verilen yeni bir boyut reallocate ederek yeniden oluşturan dizidir.

1, 2, 6 ve 7 numaralı fonksiyonlar yazdırma ve memory allocation işlemlerinden oluşabilecek kod karmaşasını azaltmak için tasarlanmışlardır.

Uygulama

Örnek Sonuçlar

1-) Program Menüsü

```
1.Create matrix and fill(REQUIRED)
2.Change content of matrix that created in option 1 (SIZE is SAME)
3.Change size and fill new matrix

Please enter the choice (0 for exit): 1
```

2-) Çözümü olmayan bir örnek(duyuruda yayınlanan) için bir input girişi ve sonuç

```
Please enter matrix size : 4

 Please fill the matrix that has size 4X4 :
 2 4 3 8
 3 4 2 8
 4 8 2 3
 2 8 3 4

1.Create matrix and fill(REQUIRED)
2.Change content of matrix that created in option 1 (SIZE is SAME)
3.Change size and fill new matrix

Please enter the choice (0 for exit):
```

3-)Çözümü olmayan bir örnek için işlem adımları

```
Row : 2 Shift number : 1

MATRIX :
 1  2  3
 2  1  3


Row : 2 Shift number : 2

MATRIX :
 1  2  3
 3  2  1


Row : 1 Shift number : 1

MATRIX :
 3  1  2
```

Resim.1

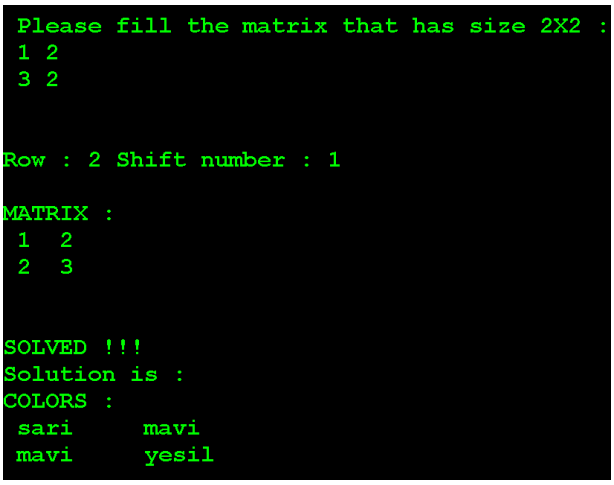
```
Row : 2 Shift number : 1

MATRIX :
 3  1  2
 1  3  2


Row : 2 Shift number : 2

MATRIX :
 3  1  2
 2  1  3


Row : 1 Shift number : 2

MATRIX :
 2  3  1
```

Resim.2

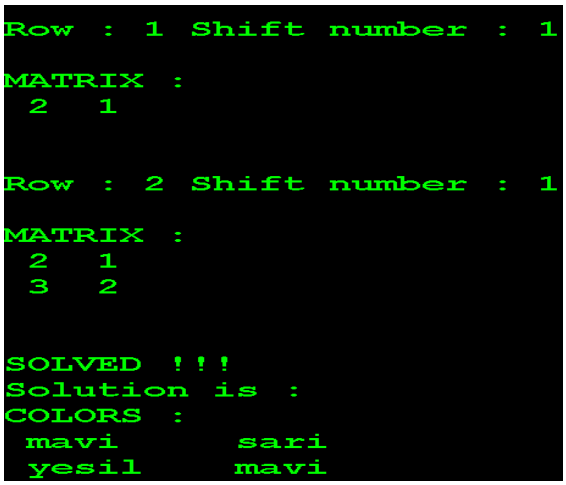
```
Row : 2 Shift number : 1

MATRIX :
 2  3  1
 3  2  1


Row : 2 Shift number : 2

MATRIX :
 2  3  1
 1  3  2
```

Resim.3

4-) Çözümü olan bir örnek için bir input girişi ve sonuçların renk olarak ekrana yazdırılması

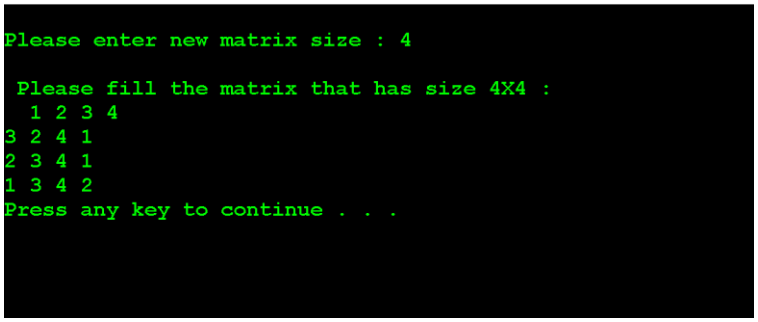


Resim.1

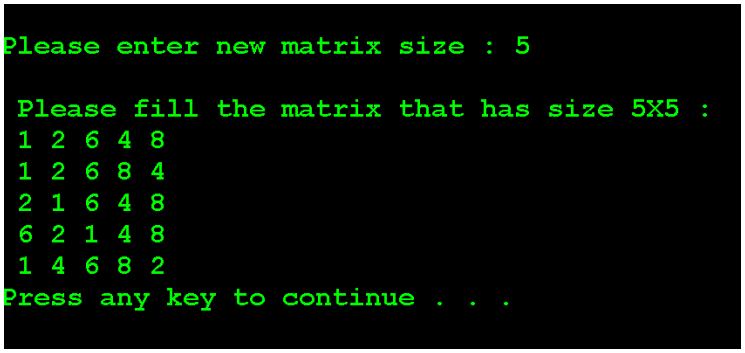


Resim.2

N=4 için örnek görsel



N=5 için örnek görsel



N=6 için örnek görsel

```
Please enter new matrix size : 6

 Please fill the matrix that has size 6X6 :
 1  2 3 4 5 6
 1 2 3 4 5 6
 1 2 3 4 5 6
 1 2 3 4 5 6
 1 2 3 4 5 6
 1 2 3 4 5 6


SOLVED !!!
Solution is :
COLORS :
 sari     mavi     yesil    mor      beyaz    siyah
 siyah    sari     mavi     yesil    mor      beyaz
 beyaz    siyah    sari     mavi     yesil    mor
 mor      beyaz    siyah    sari     mavi     yesil
 yesil    mor      beyaz    siyah    sari     mavi
 mavi     yesil    mor      beyaz    siyah    sari


SOLVED !!!
Solution is :
COLORS :
 sari     mavi     yesil    mor      beyaz    siyah
 siyah    sari     mavi     yesil    mor      beyaz
 beyaz    siyah    sari     mavi     yesil    mor
 mor      beyaz    siyah    sari     mavi     yesil
 mavi     yesil    mor      beyaz    siyah    sari
 yesil    mor      beyaz    siyah    sari     mavi


SOLVED !!!
Solution is :
COLORS :
 sari     mavi     yesil    mor      beyaz    siyah
 siyah    sari     mavi     yesil    mor      beyaz
 beyaz    siyah    sari     mavi     yesil    mor
 yesil    mor      beyaz    siyah    sari     mavi
 mavi     yesil    mor      beyaz    siyah    sari
 mor      beyaz    siyah    sari     mavi     yesil
```

Sonuç

Kullanıcıdan input okuma vs gibi işlemleri en büyük değeri etkilemeyeceği için karmaşıklık hesabına katmadan asıl işlemlerin yapıldığı fonksiyona bakacak olursak bu fonksiyon bir for döngüsünde n defa dönmektedir. Her bir iterasyon shift edilmesi gerekli mi değil mi ? kontrolüne göre bir sonraki satıra inme durumunu belirlemektedir. Böyle bir durumda kontrol işlemleri için her bir dallanma işlemi T(N) = N*(N-1)*(N-2).. = O(N!) olarak ifade edildiği gibi N! karmaşıklığa sahiptir. Bu işlemin döngüde N defa yapıldığı düşünülürse Worst Case için karmaşıklık

O(N)=N*N! olacaktır.

Kod

```
1.  /**
2.  @file
3.
4.  Bu uygulamada verilen renk degerleri icin bir matris olusturulmus ve her bir sutunda verilen s
    ayi(yada renk) degerinin 1 kez
5.  tekrar etmesi saglanacak sekilde rekursif bir backtracking algoritmasi tasarlanmistir.
6.  Program menusu 3 kisimdir.
7.  1-Kullanicidan matris boyutu alinir, matris olusturur, doldurtulur ve cozulur.
8.  2-Boyut sabit kalir matris yeniden doldurulur ve cozulur.
9.  3-Yeni bir Matris boyutu alinir, matris olusturur, doldurtulur ve cozulur.
10. 0- Programdan cikmayi saglar.
11.
12. @author
13.
14. Name              :       Muhammed Yasin SAGLAM
15. Student No        :       15011804
16. Date              :       18/11/2017
17. E-Mail            :       myasinsaglam1907@gmail.com
18. Compiler Used     :       GCC
19. IDE               :       DEV-C++(Version 5.11)
20. Operating System  :       Windows 10 Educational Edition
21. */
22. #include <stdio.h>
23. #include <stdlib.h>
24.
25. /*
26. verilen satiri 1 kez saga shift eden fonksiyon
27. */
28. void shifteR(int *arr,int n){
29.     int temp,i;
30.     temp=arr[n-1]; //son elemani tempe atariz.
31.     for (i=1; i<n; i++) {
32.         arr[n-i]=arr[n-i-1]; //diger elemanlari 1 er saga kaydir...
33.     }
34.     arr[0]=temp; //son elemani en basa yaz
35. }
36.
37. /*
38. verilen matrisi ekrana sayi degerleri olarak verilen satirina kadar yazdiran fonksiyon
39. */
40. void printMatrix(int **arr,int row,int n){
41.     int i,j;
42.     printf("\nMATRIX :\n");
43.     for (i=0; i<row+1; i++) {  //verilen satira kadar
44.         for(j=0;j<n;j++){ //matrisin tum sutunlarini
45.             printf(" %d ",arr[i][j]); //yazdir
46.         }
47.         printf("\n");
48.     }
49. }
50.
51. /*
52. verilen matrisi ekrana karsiligi olan renk degerlerine gore verilen satirina kadar yazdiran fo
    nksiyon
53. */
54. void printColor(int **arr,int row,int n){
55.     int i,j;
```

```
56.     char colors[8][10]={"sari","mavi","yesil","mor","beyaz","siyah","kirmizi","gri"}; //soruda
     verilen renkler indislere uygun tanimlaniyor
57.     printf("\nCOLORS :\n");
58.     for (i=0; i<row+1; i++) { //verilen satira kadar
59.         for(j=0;j<n;j++){ //tum sutunlardaki renkleri
60.             printf(" %-7s ",colors[arr[i][j]-1]); //yazdir
61.         }
62.         printf("\n");
63.     }
64. }
65.
66. /*
67. verilen satirin her bir elamaninin bulundugu sutundaki ust satirlara bakarak aynisinin olup ol
    madigini kontrol eden,
68. varsa 0 yoksa 1 donduren fonksiyon
69. */
70. int control(int **arr,int row,int n){
71.     int i=0, j=row;
72.     int temp;
73.
74.     if(row>0){ //satir 0 degilse
75.         for (i=0; i<n; i++) { //satirin elemanlarini sirayla al
76.             temp=arr[row][i];
77.             for (j=row-1; j>=0; j--) { //ayni sutununun ust degerleri icin
78.                 if(temp==arr[j][i]){ //ayni sayidan bulunuyorsa
79.                     //printf("\nnot fisible...\n");
80.                     return 0; //0 dondur
81.                 }
82.             }
83.         }
84.         return 1;   //ilk satirsa 1 dondur
85.     }
86.     return 1; //0 ile cikilmamissa hersey fisible demektir 1 dondur
87. }
88.
89. /*
90. verilen matrisin rekursif cozumu yapan fonksiyon
91. eger fisible olmussa 1 satir ilerler degilse eleman sayisi-1 kez shift islemi yapar
92. */
93. int solve(int **arr,int row,int n){
94.     int i;
95.     if(row<n){ //eger bakilan satir eleman sayisindan kucukse
96.         for(i=0;i<n;i++){ //eleman sayisi kadar
97.             if(control(arr,row,n)){ //varilen satiri kontrol et
98.                 solve(arr,row+1,n); //fisible ise bir sonraki satira gec
99.             }
100.                 if(i<n-1){ //fisible degil ise ve n-1 defa ile sinirli olarak
101.                     //printf("\n\nRow : %d Shift number : %d \n",row+1,i+1);  //kontrol ama
    cli yazdirma
102.                     shifteR(arr[row],n); //ilgili satiri 1 saga kaydir.
103.                     //printMatrix(arr,row,n);  //kontrol amacli yazdirma
104.                 }
105.             }
106.         }
107.         else{ //degilse
108.             row--; //yazdirmada bir sorun cikmamasi icin row u 1 azalt
109.             printf("\n\nSOLVED !!!\nSolution is : ");
110.             //printMatrix(arr,row,n);
111.             printColor(arr,row,n); //bulunan cozumu ekrana renk olarak yazdir
112.             return 1; // 1 dondur
113.         }
```

```
114.        return 0; //cozum bulunamazsa 0 dondur
115.        }
116.
117.        /*
118.        Dinamik memory allocate eden ve diziyi disari donduren fonksiyon
119.        */
120.        int **Allocator(int size){
121.            int **array;
122.            int i;
123.            array=(int**)malloc(sizeof(int*)*size);
124.            for(i=0;i<size;i++){
125.                array[i]=(int*)malloc(sizeof(int)*size);
126.            }
127.            if(!array){
128.                system("COLOR c");
129.                printf("Array Not Allocated !!! Quitting...");
130.                exit(0);
131.            }
132.            return array;
133.        }
134.
135.        /*
136.        Verilen integer dizinin boyutunu realloc islemiyle degistiren fonksiyon
137.        */
138.        void ReAllocator(int ***matrix,int size){
139.            int i;
140.            (*matrix)=realloc((*matrix),sizeof(int*)*size);
141.            for(i=0;i<size;i++){
142.                (*matrix)[i]=malloc(sizeof(int)*size);
143.            }
144.
145.        }
146.
147.        int main(int argc, const char * argv[]) {
148.            int **matrix; //matris tanimlaniyor
149.            int i,j; //cevrim degiskenleri
150.            int n; //matris boyutu degiskeni
151.            int choice; //menu icin secenek degiskeni
152.
153.            //menu bilgi mesaji
154.            printf("\n1.Create matrix and fill(REQUIRED) \n2.Change content of matrix that crea
    ted in option 1 (SIZE is SAME)\n3.Change size and fill new matrix\n\nPlease enter the choice (
    0 for exit): ");
155.            scanf("%d",&choice);            //Kullanicidan tercih okunuyor
156.            system("CLS");
157.            while(choice!=0){ //0-Cikis secilmedigi muddetce
158.                if(choice==1){ //matris olustur ve doldur
159.                    printf("\nPlease enter matrix size : ");
160.                    scanf("%d",&n); //eleman sayisini oku
161.                    matrix=Allocator(n); //matrisi allocate et
162.                    printf("\n Please fill the matrix that has size %dX%d :\n ",n,n);
163.                    for(i=0;i<n;i++){
164.                        for(j=0;j<n;j++){
165.                            scanf("%d",&matrix[i][j]); //matrisi doldur
166.                        }
167.                    }
168.                    solve(matrix,0,n); //matrisi coz
169.                    system("PAUSE");
170.                    system("CLS");
171.                }
```

```
172.                if(choice==2){ //1. tercihte olusturulan matrisin icerigini degistir.Boyut ayni
    ...
173.                    printf("\n Please fill the matrix that has size %dX%d :\n ",n,n);
174.                    for(i=0;i<n;i++){
175.                        for(j=0;j<n;j++){
176.                            scanf("%d",&matrix[i][j]); //yeni icerik okunuyor
177.                        }
178.                    }
179.                    solve(matrix,0,n); //matris cozunuyor
180.                    system("PAUSE");
181.                    system("CLS");
182.                }
183.                if(choice==3){ // yeni boyut al ve yeni bir matris olustur.
184.                    printf("\nPlease enter new matrix size : ");
185.                    scanf("%d",&n); //yeni matris boyutu kullanicidan okunuyor
186.                    ReAllocator(&matrix,n); //verilen boyut icin yer reallocate ediliyor
187.                    printf("\n Please fill the matrix that has size %dX%d :\n ",n,n);
188.                    for(i=0;i<n;i++){
189.                        for(j=0;j<n;j++){
190.                            scanf("%d",&matrix[i][j]); //yeni boyutlu matrisin icerigi doldurul
    uyor
191.                        }
192.                    }
193.                    solve(matrix,0,n); //matris cozuluyor
194.                    system("PAUSE");
195.                    system("CLS");
196.                }
197.            printf("\n1.Create matrix and fill(REQUIRED) \n2.Change content of matrix that crea
    ted in option 1 (SIZE is SAME)\n3.Change size and fill new matrix\n\nPlease enter the choice (
    0 for exit): ");
198.            scanf("%d",&choice);            //Kullanicidan tercih okunuyor
199.            system("CLS");
200.            }
201.
202.            //matris free ediliyor...
203.            for(i=0;i<n;i++){
204.                free(matrix[i]);
205.            }
206.            free(matrix);
207.            return 0;
208.        }
```