TÜRKİYE CUMHURİYETİ

YILDIZ TEKNİK ÜNİVERSİTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

ALGORİTMA ANALİZİ DÖNEM PROJESİ RAPORU

Öğrenci No: 20011023

Öğrenci Adı Soyadı: Mehmet Alperen Ölçer

Öğrenci E-Posta: alperen.olcer@std.yildiz.edu.tr

VİDEO LİNK : https://youtu.be/ieeZNa-Y8Tc

Ders/Grup: BLM3021 Algoritma Analizi/ 1.Grup

Ders Yürütücüsü

PROF DR MİNE ELİF KARSLIGİL

19 Kasım 2022

YÖNTEM:

Problem : Matriste satırı sağa bir kaydırma

Çözüm : En son sutundaki değeri temp'e alıp sondan başa gelerek soldakini bulunduğum indexe kopyalayarak gidiyorum. En son birinci elemanı da temp yapıyorum.

Problem : Matrix okuma ve sayı matrisi oluşturma

Çözüm : NxN lik dinamik sayı matrisi oluşturuluyor. Sonrasında N lik dinamik bir different_strings dizisi de oluşturuluyor. İlk satir different_strings içine okunuyor ve ilk int matrix satırı 0'dan N-1'e değerler oluyor. Sonra her yeni kelimenin different_strings dizideki indexi sayı matrisine yerleştiriliyor. Ekstradan bir string matrisi tutulmuyor.

Problem : Üst satırların değerlerinde tekrar eden değer var mı kontrolü

Çözüm : Bir satırdaki elemanlar aynı index kullanılarak üst satırlarda aranıyor. Tekrar etmiş olup olmamasına göre cevap döndürülüyor.

Problem : Backtracking ile çözmek

Çözüm : ALGORITHM

First generate an array called shift_counts initial values are 0s

Initialize row_index to 1

call Funtion (exit conditions are ; row_index is equal to N or -1 (if recursion breaks because of row_index=-1, then no solution. if row_index=N, then matrix is reordered))

In function

check if N and above satisfies the rule

satisfies:

increment row_count

not satisfies:

check if the row is shiftable (shift_counts[row_index] < N-1)

shiftable:

shift and increase shift_counts[row_counts]

not shiftable:

move to upper row

shift_counts[row_index] = 0

row_index --;

search for a shiftable row upwards, everytime going up reset shift count

if not found

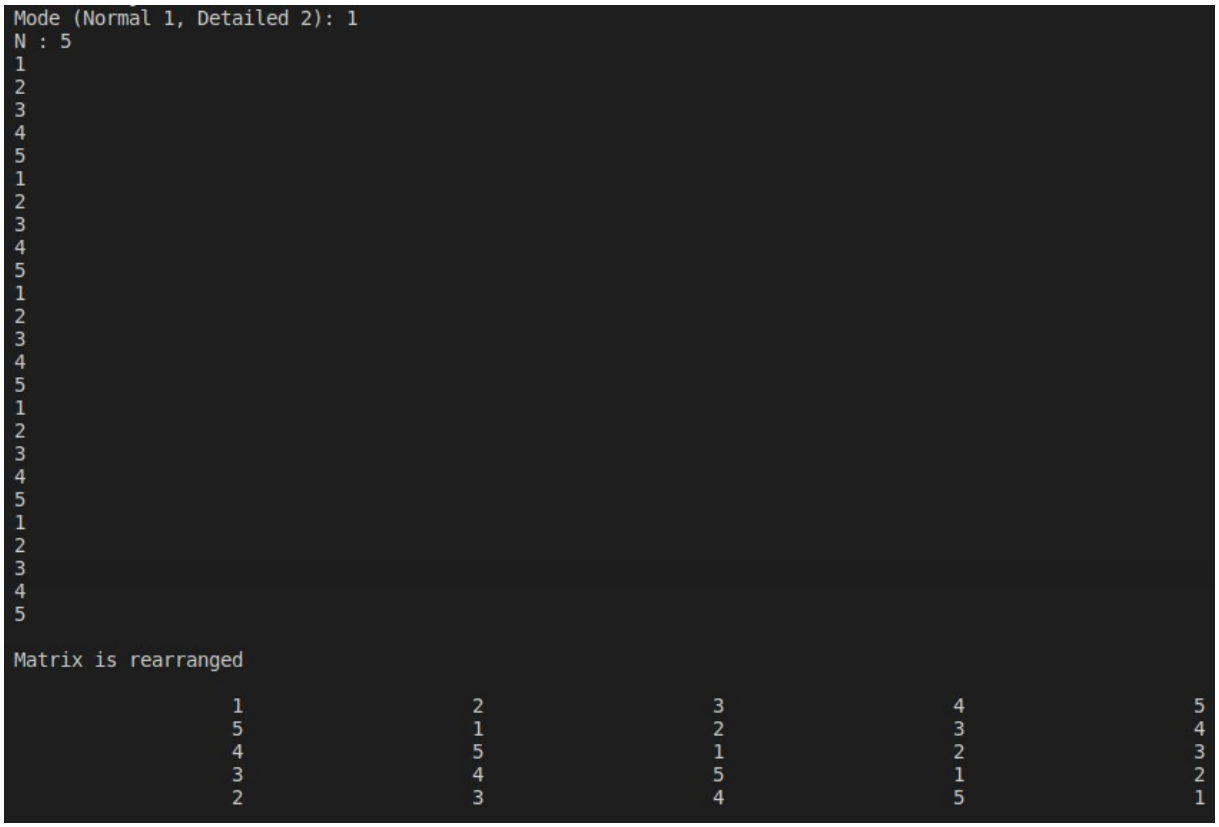
exit with no solution

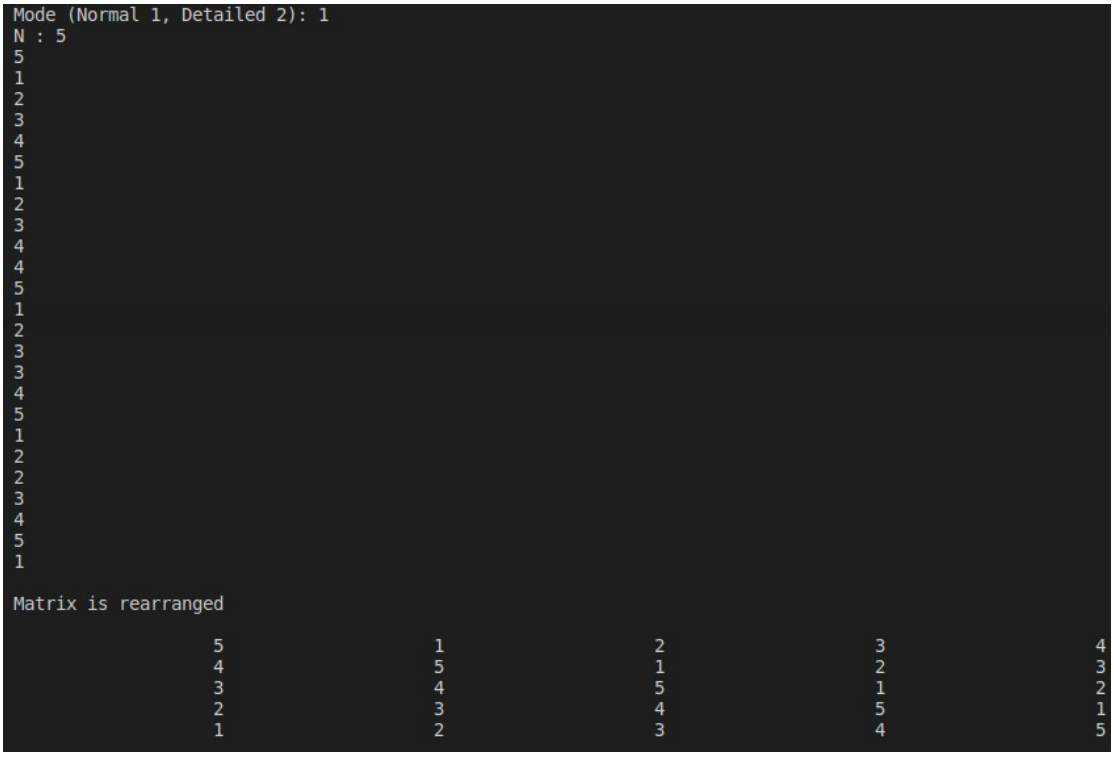
if found

shift that row

Call Funtion

UYGULAMA:





```
Mode (Normal 1, Detailed 2): 2
N : 4
s
y
k
m
y
m
k
s
s
m
k
y
y
s
k
m

 0  1  2  3
 1  3  2  0
 0  3  2  1
 1  0  2  3


              s              y              k              m
              y              m              k              s
              s              m              k              y
              y              s              k              m


 0  1  2  3
 0  1  3  2
 0  3  2  1
 1  0  2  3


 0  1  2  3
 2  0  1  3
 0  3  2  1
 1  0  2  3


 0  1  2  3
 3  2  0  1
 0  3  2  1
 1  0  2  3


 0  1  2  3
 3  2  0  1
 1  0  3  2
 1  0  2  3


 0  1  2  3
 3  2  0  1
 1  0  3  2
 3  1  0  2


 0  1  2  3
 3  2  0  1
 1  0  3  2
 2  3  1  0


Matrix is rearranged

 0  1  2  3
 3  2  0  1
 1  0  3  2
 2  3  1  0


              s              y              k              m
              m              k              s              y
              y              s              m              k
              k              m              y              s
```

```
Mode (Normal 1, Detailed 2): 2
N : 3
Kirmizi
Mavi
Yesil
Yesil
Mavi
Kirmizi
Kirmizi
Yesil
Mavi

  1   2   3
  3   2   1
  1   3   2


             Kirmizi                Mavi                Yesil
               Yesil                Mavi             Kirmizi
             Kirmizi              Yesil                  Mavi


  1   2   3
  1   3   2
  1   3   2


  1   2   3
  2   1   3
  1   3   2


  3   1   2
  2   1   3
  1   3   2


  3   1   2
  3   2   1
  1   3   2


  3   1   2
  1   3   2
  1   3   2


  2   3   1
  1   3   2
  1   3   2


  2   3   1
  2   1   3
  1   3   2


  2   3   1
  3   2   1
  1   3   2


Matrix can not be rearranged
```

```
Mode (Normal 1, Detailed 2): 2
N : 3
Mor
Yesil
Siyah
Siyah
Mor
Yesil
Yesil
Siyah
Mor

  1   2   3
  3   1   2
  2   3   1


                Mor               Yesil               Siyah
              Siyah                 Mor               Yesil
              Yesil               Siyah                 Mor


Matrix is rearranged

  1   2   3
  3   1   2
  2   3   1


                Mor               Yesil               Siyah
              Siyah                 Mor               Yesil
              Yesil               Siyah                 Mor
```

```
Mode (Normal 1, Detailed 2): 2
N : 3
a
b
c
b
a
c
c
a
b

  1   2   3
  2   1   3
  3   1   2


                a                b                c
                b                a                c
                c                a                b


  1   2   3
  3   2   1
  3   1   2


  1   2   3
  1   3   2
  3   1   2


  3   1   2
  1   3   2
  3   1   2


  3   1   2
  2   1   3
  3   1   2


  3   1   2
  3   2   1
  3   1   2


  2   3   1
  3   2   1
  3   1   2


  2   3   1
  1   3   2
  3   1   2


  2   3   1
  2   1   3
  3   1   2


Matrix can not be rearranged
```

SONUÇ:

--» Complexity is O(N^2)

```
// checks if the row values repeat themself at upper rows
int check_columns_upward(int **m, int row_index, int N)
{
   if (row_index == 0)
      return 1;

   int i, j=0, flag=1;
   while ( j < N && flag )
   {
      i = row_index;
      while ( i-- > 0 && flag )
         if (m[row_index][j] == m[i][j])
            flag = 0;
      j ++;
   }
   return flag;
}
```

--» Complexity is O(N)

```
// shifts a row
void shift(int **m, int row_index, int N, int mode)
{
   int j, temp;
   temp = m[row_index][N-1];
   for ( j = N-1; j >= 0; j--)
```

```
        m[row_index][j+1] = m[row_index][j];
    m[row_index][0] = temp;
    char **tempp;
    if (mode != 1)
        print_matrix(m, tempp, N, 1);
}
```

--» Complexity is O(N^N)

PSEUDO CODE

```
    shift_counts[N] = {0}
    row_index = 1
    solution(matrix, shift_counts, row_index, N, mode)
    solution:
        if (row_index != N && row_index != -1)
            if check_columns_upward(matrix, row_index, N):
                row_index += 1
            else:
                if shift_counts[row_index] < N-1:
                    shift(matrix, row_index, N)
                    shift_counts[row_index] += 1
                else:
                    shift_counts[row_index] = 0
                    row_index -= 1
                    flag = 0;
                    while row_index >= 0 and (not flag):
                        if shift_counts[row_index] < N-1:
                            shift(matrix, row_index, N)
                            shift_counts[row_index] += 1
                            flag = 1
                        else:
                            shift_counts[row_index] = 0
```

```
                    row_index -= 1
        solution(matrix, shift_counts, row_index, N, mode)
    else if row_index == N:
        solution found
    else if row_index == N:
        solution not found
```

VİDEO LİNK : https://youtu.be/ieeZNa-Y8Tc