YILDIZ TEKNİK ÜNİVERSİTESİ BİLGİSAYAR MÜHENDİSLİĞİ

BLM 1012- YAPISAL PROGRAMLAMAYA GİRİŞ DÖNEM PROJESİ

FLOOD FILL ALGORİTMASI

Öğrenci NO:21011003

İsim: Doğukan Baş

E-mail: dogukan.bas@std.yildiz.edu.tr

Ders Yürütücüsü: Öğr.Gör.Dr. Ahmet Elbir

YOUTUBE VIDEO LINKI: https://www.youtube.com/watch?v=LZ2gj86_x6w

İçindekiler

Algoritmanın Açıklaması 3
Kullanım Alanları 3
Örnek 1 3
Örnek 2 4
Örnek 3 5
Avantaj ve Dezavantajları 5
Avantajları 5
Dezavantajları 5
Zaman Karmaşıklığı 6
Best Case 6
Worst Case 7
Ortalama bir örnek 9
Boundary Fill algoritmasıyla karşılaştırılması 9
C Dilinde kodu 10
Kaynakça 12

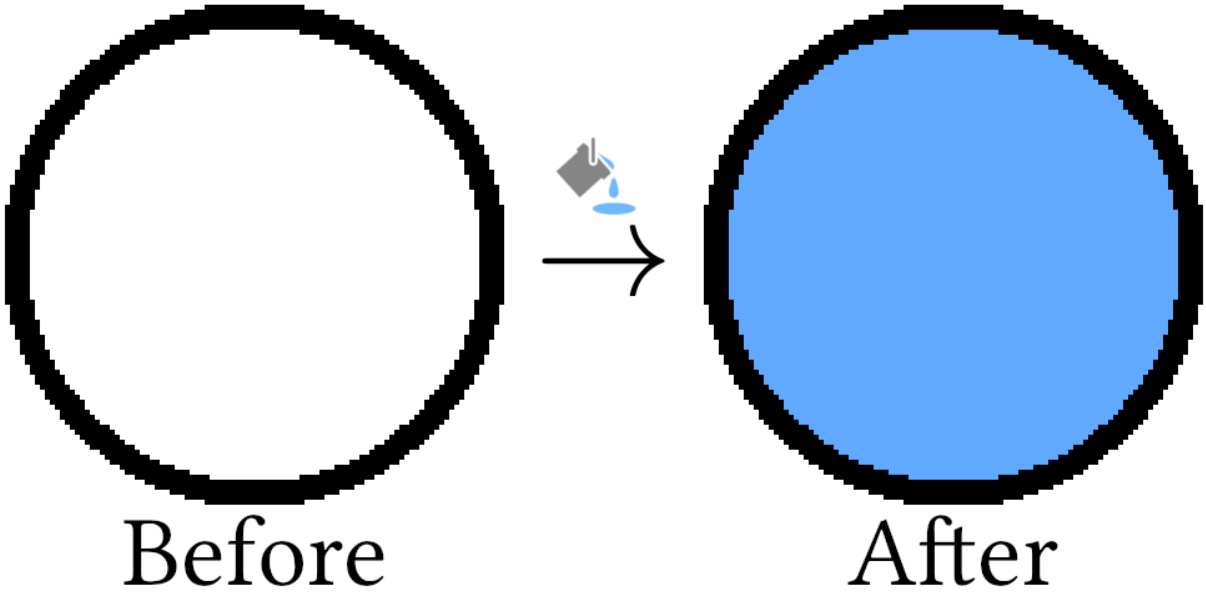
Algoritmanın Açıklaması

Başlangıç elemanı ve o elemanın çevresindeki aynı özelliğe sahip elemanlara recursive bir şekilde (komşuların komşuları için de fonksiyon çağrılır) yeni bir özellik atamak için kullanılır. Bu algoritmada durma koşulu yeni elemanın özelliğinin ilk seçilen elemanın özelliğiyle aynı olmaması veya matrisin sınırlarının dışına çıkılması durumudur.

Kullanım Alanları

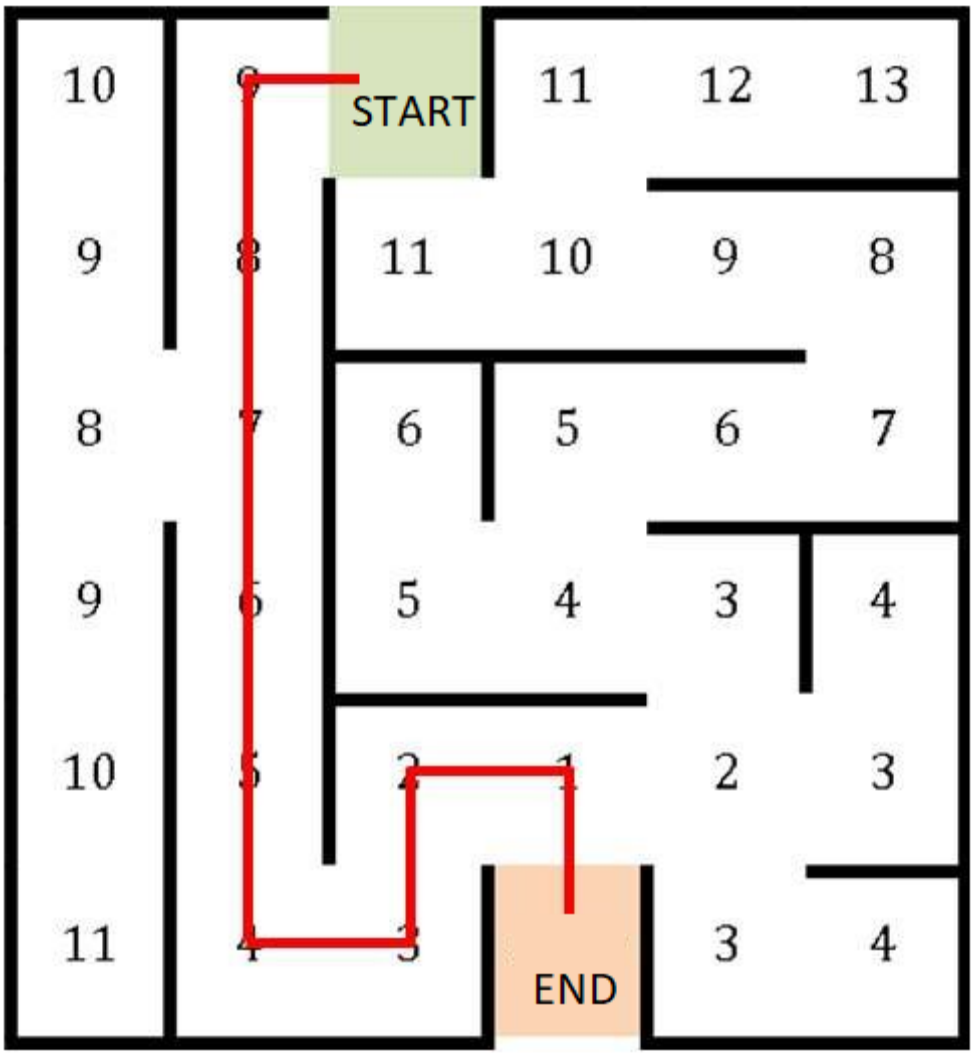
Örnek 1

En bilinen uygulması paint uygulamalarındaki “bucket fill” seçeneğidir. Örneğin paintte beyaz üzerine siyah bir çember çizip mavi rengi seçip bucket fill ile bu çemberin ortasındaki boşluğa dokunursanız çemberin içi bastığınız noktadan itibaren tek tek komşulara yayılarak maviye boyanacaktır. Bu bir flood fill algoritması örneğidir.



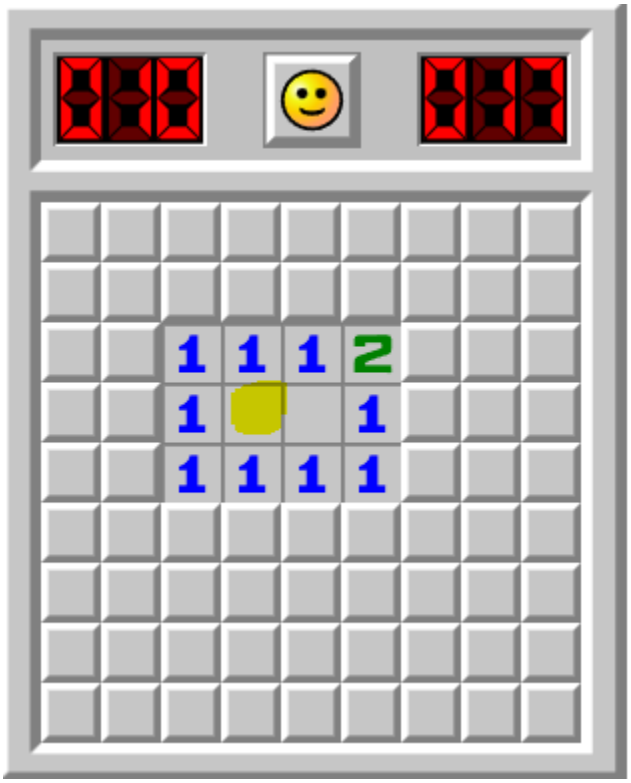
Örnek 2

Flood Fill algoritması aynı zamanda labirent problemlerinin çözümünde de kullanılabilir. Çıkış noktasından başlanılır ve çıkış noktasına 1 sayısı atanır. Sonra 1 sayısı verilen elemanın komşularına 2 sayısı atanır. Daha sonra 2 sayısı atanan elemanın komşularına 3 sayısı atanır ve bu işlem labirentin başlangıcına gelene kadar yapılır. Labirentin başlangıcına gelindiğinde program başlangıçtan itibaren komşu elemanlardan en küçük değere sahip komşuya atlayarak yolu çıkışa kadar tamamlar. Örnek bir çözüm aşağıda verilmiştir.



Örnek 3

Flood fill algoritmasının kullanıldığı bir diğer uygulama ise mayın tarlası oyunudur. Mayın tarlası oyununda eğer seçilen elemanın komşularında bomba yoksa o bölge boş olarak açılır. Eğer seçilen elemanın komşusunun komşuları da boş ise o eleman de boş olarak açılabilir. Ta ki komşularında bomba olan bir bölgeye gelene kadar komşuları açar. Yani oyun bizim için gereksiz yerleri temizler. Burada da flood fill algoritması kullanılmıştır. Aşağıda bir örnek verilmiştir:



Sarıyla işaretli kare açıldığında. Program komşularında bomba olan bir yer bulana kadar flood fill algoritmasıyla açar. Eğer o anki elemanın komşularında bomba varsa orada durur ve o eleman için daha ileri gitmez.

Avantaj ve Dezavantajları

Avantajları

1. Verimlilik ve doğrululuğu yüksektir.
2. İmplementasyonu kolaydır.
3. Kullanım alanı çoktur.

Dezavantajları

1. Kaç tane alt elemanın doldurulacağı önceden bilinmediği için stackte çok birikmeye sebep olabilir.

Zaman Karmaşıklığı

Best Case

N x N lik bir matris için four- way flood fill algoritmasında en iyi durum ilk çağrılan elemanın komşularının sınır olmasıdır(yani ilk çağrılan elemanın komşularının başka renk olması). İlk eleman için fonksiyon çağrılır daha sonra elemanın komşuları için de fonksiyon çağrılır fakat bu sefer o elemanlar base case durumuna girer ve komşuların komşuları için fonksiyon çağrılmaz. Dolayısıyla toplam 1+4 = 5 kez fonksiyon çağrılır. Karmaşıklık O(5)'dir bu da O(1) demektir.

$\begin{bmatrix} 2 & 4 & 4 \\ 3 & \mathbf{1} & 3 \\ 4 & 4 & 2 \end{bmatrix}$ Şeklindeki bir matriste en ortadaki kalın yazılı elemandan başlayarak komşu elemanlara yayılan ve komşu elemanları base case değilse 0'a boyayan flood fill algoritması best case durumuna örnektir. Toplam işlem sayısı ve her işlemdeki matris durumu aşağıda verildiği gibidir.

```
----------------------
1th operation
2 4 4
3 1 3
4 4 2

----------------------
2th operation
2 4 4
3 4 3
4 4 2

----------------------
3th operation
2 4 4
3 4 3
4 4 2

----------------------
4th operation
2 4 4
3 4 3
4 4 2

----------------------
5th operation
2 4 4
3 4 3
4 4 2

FINAL MATRIX
2 4 4
3 4 3
4 4 2

COMPLEXITY:
*****
```

Base case durumunda bile matris ekrana bastırılmıştır. Bu yüzden işlem yapılmasa dahi matris ekrana bastırılır. Görüldüğü üzere 5 çağrılmanın sadece 1 tanesinde işlem yapılmıştır. Diğer 4 çağırmada fonksiyon base case içerisine girmiştir. Toplam çağrılma sayısı 5'tir

Aynı zamanda burda fark edilecek bir diğer şey base case kontrolünün yeni pikselin sınır olup olmaması değil, yeni pikselin ilk seçilen elemanın rengiyle aynı olup olmadığı olduğundan sınırımızı birden fazla renkle oluşturabiliriz.

Worst Case

N x N lik bir matris için four- way flood fill algoritmasında en kötü durum başlangıçta matrisin ortasındaki elemanın seçilerek tüm matrisin boyanması durumudur .Her bir eleman 4 kez tekrar çağrılır. Dolayısıyla en kötü durumda ilk seçim yapıldığında çağrılan fonksiyonu da hesaba katarsak $4\ x\ N^2 + 1$ kere işlem yapılır. Bu da O(N^2) demektir. $\begin{bmatrix} 1 & 1 & 1 \\ 1 & \mathbf{1} & 1 \\ 1 & 1 & 1 \end{bmatrix}$ Şeklindeki bir matriste en ortadaki kalın yazılı elemandan başlayarak komşu elemanlara yayılan flood fill algoritması worst case durumuna örnektir. Toplam işlem sayısı ve her işlemdeki matris durumu bir sonraki sayfada verildiği gibidir.

```
1th operation
111
111
111

2th operation
111
101
111

3th operation
101
101
111

4th operation
101
101
111

5th operation
101
101
111

6th operation
100
101
111

7th operation
100
101
111

8th operation
100
100
111

9th operation
100
100
111
```

```
10th operation
100
100
110

11th operation
100
100
110

12th operation
100
100
110

13th operation
100
100
110

14th operation
100
100
100

15th operation
100
100
100

16th operation
100
100
100

17th operation
100
100
100

18th operation
100
100
000
```

```
19th operation
100
000
000

20th operation
000
000
000

21th operation
000
000
000

22th operation
000
000
000

23th operation
000
000
000

24th operation
000
000
000

25th operation
000
000
000

26th operation
000
000
000

27th operation
000
000
000
```

```
27th operation
000
000
000

28th operation
000
000
000

29th operation
000
000
000

30th operation
000
000
000

31th operation
000
000
000

32th operation
000
000
000

33th operation
000
000
000

34th operation
000
000
000

35th operation
000
000
000
```

```
36th operation
0 0 0
0 0 0
0 0 0

37th operation
0 0 0
0 0 0
0 0 0

FINAL MATRIX
0 0 0
0 0 0
0 0 0

COMPLEXITY:
**********
```

Base case durumunda bile matris ekrana bastırılmıştır. Bu yüzden hiçbir değişiklik olmadığında dahi kod ekrana bastırılmıştır. Görüldüğü üzere toplam fonksiyon çağrılma sayısı 37 Yani $4\ x\ N^2 + 1$' dir.

Best case durumunda da worst case durumunda da matriste hiçbir değişiklik olmamasına rağmen fonksiyona girme sebebi bu algoritmada daha önce ziyaret edilen elemanın bir daha çağrılmaması yönünde bir mekanizma olmamasıdır. Aynı eleman için fonksiyon 4 defaya kadar çağrılabilir.

Ortalama bir örnek

$\begin{bmatrix} 1 & 1 & 4 \\ 3 & 2 & \mathbf{2} \\ 2 & 2 & 1 \end{bmatrix}$ durumunda kalın olan değerden başlayarak 9 a boyayan flood fill algoritması.

```
1th operation
1 1 4
3 2 2
2 2 1

2th operation
1 1 4
3 2 9
2 2 1

3th operation
1 1 4
3 2 9
2 2 1

4th operation
1 1 4
3 2 9
2 2 1

5th operation
1 1 4
3 2 9
2 2 1

6th operation
1 1 4
3 9 9
2 2 1

7th operation
1 1 4
3 9 9
2 2 1

8th operation
1 1 4
3 9 9
2 9 1

9th operation
1 1 4
3 9 9
2 9 1

10th operation
1 1 4
3 9 9
2 9 1

11th operation
1 1 4
3 9 9
2 9 1

12th operation
1 1 4
3 9 9
9 9 1

13th operation
1 1 4
3 9 9
9 9 1

14th operation
1 1 4
3 9 9
9 9 1

15th operation
1 1 4
3 9 9
9 9 1

16th operation
1 1 4
3 9 9
9 9 1

17th operation
1 1 4
3 9 9
9 9 1

FINAL MATRIX
1 1 4
3 9 9
9 9 1

COMPLEXITY:
****************
```

Toplam 17 kere fonksiyon çağrılmıştır.Bu durum best case'den kötü , worst caseden ise çok iyidir.

Boundary Fill algoritmasıyla karşılaştırılması

Boundary fill algoritmasında içeride birden fazla renk olabilir. Sınırlara kadar tüm bölgeyi seçilen yeni renkle boyar. Flood fill algoritmasında ise sınır belirtilmez , başlangıç ile aynı renge sahip komşular boyanır.

1. Boundary fill algoritmasında sınırın rengi programa verilir.
2. Flood fill algoritmasında ise sınırın rengi verilmez, başlangıçta boyanacak eleman verilir.
3. Boundary fill algoritmasında kontrol yeni pikselin sınır olup olmadığıdır.
4. Flood fill algoritmasında kontrol yeni pikselin ilk elemanın rengiyle aynı olup olmadığıdır.
5. İkisinin de best case durumları ve worst case durumlarının karmaşıklığı aynıdır.

C Dilinde kodu

```c
#include <stdio.h>
#define MAX 50
int floodFill(int matrix[MAX][MAX],int fillRow,int fillColumn, int row,int
column,int color,int fillColor);
int main(){
    int matrix[MAX][MAX],i,j,row,column,fillColumn,fillRow,fillColor;
    printf("amount of rows:");
    scanf("%d",&row);
    printf("amount of columns:");
    scanf("%d",&column);

    for(i=0;i<row;i++){
        for(j=0;j<column;j++){
            printf("insert %dth row %dth column:",i,j);
            scanf("%d",&matrix[i][j]);
        }
    }

    printf("THE MATRIX\n");
    for(i=0;i<row;i++){
        for(j=0;j<column;j++){
            printf("%d",matrix[i][j]);
        }
        printf("\n");
    }

    printf("enter the row of the element you want to start filling from:(first
row is 0th row):");
    scanf("%d",&fillRow);
    printf("enter the column of the element you want to start filling from (first
column is 0th column):");
    scanf("%d",&fillColumn);
    printf("Which color would you like to fill");
    scanf("%d",&fillColor);
    floodFill(matrix,fillRow,fillColumn,row,column,matrix[fillRow][fillColumn],fi
llColor);
    printf("\n");

    printf("FINAL MATRIX\n");
    for(i=0;i<row;i++){
        for(j=0;j<column;j++){
            printf("%d",matrix[i][j]);
```

```
        }
        printf("\n");
    }

    return 0;
}

int floodFill(int matrix[MAX][MAX],int fillRow,int fillColumn, int row,int
column,int color,int fillColor){
    //eger matrisin disina cikiliyorsa, veya yeni elemanin rengi ilk secile
elemanin rengiyle ayni degilse bu base casedir.
    if(fillRow<0 || fillRow>=row ||fillColumn<0 || fillColumn>=column||
matrix[fillRow][fillColumn]!=color){

        return 0;
    }
    else{
        matrix[fillRow][fillColumn]=fillColor;

        //eleman boyandıktan sonra o elemanın 4 komsusu da boyanir.
        floodFill(matrix,fillRow-1,fillColumn,row,column,color,fillColor);
        floodFill(matrix,fillRow+1,fillColumn,row,column,color,fillColor);
        floodFill(matrix,fillRow,fillColumn+1,row,column,color,fillColor);
        floodFill(matrix,fillRow,fillColumn-1,row,column,color,fillColor);
    }
}
```

Kaynakça

https://en.wikipedia.org/wiki/Flood_fill

https://techdifferences.com/difference-between-flood-fill-and-boundary-fill-algorithm.html

https://www.geeksforgeeks.org/difference-between-flood-fill-and-boundary-fill-algorithm/

https://www.hackerearth.com/practice/algorithms/graphs/flood-fill-algorithm/tutorial/

https://www.researchgate.net/publication/315969093_Maze_Exploration_Algorithm_for_Small_Mobile_Platforms