Basel Kelziye
20011906

Veri Yapıları Ve Algoritmalar

30/05/2022

PROJE ÖDEVİ

Grup 3: DR. Göksel Biricik.

Problem Tanımı: Labirentten Çıkma Problemi

Algoritmik Yaklaşım: DFS

Gidilecek bütün Olasılıkları Denersek Labirentten çıkarız(Eğer çıkış var ise) mantığı ile çözüm Ürettim. Bunu Uygulamanın En güzel yolu rekürsiv fonksiyonlardır.

1-)Labirenti Dosyadan Okuma:

```c
void determine_dimension_of_maze(int *row_length, int *column_length)
{ // row satir ,column sutun
    const char EOL = '\n';
    char c;
    char buff[255];
    FILE *fp = fopen("maze.txt", "r");
    if (fp == NULL)
    {
        printf("\nHata:Dosya Acilamadi!");
        exit(101);
    }

    *column_length = 0;
    while ((c = fgetc(fp)) != EOL)
    {
        *column_length = *column_length + 1;
    }
    printf("\nColumn length -> %d", *column_length);
    //     *row_length = *row_length + 1;
    //  }
    *row_length = 0;
    do
    {
        *row_length = *row_length + 1;
    } while (fgets(buff, 255, fp) != NULL);

    fclose(fp);
}
```

Ilk Once Dosyadan Okunan labirentin Boyutlarını Tespit edip,Ona göre dinamik Matrix Oluşturdum. Ve Labirenti Matrise Kopyaladım.

```c
void read_maze_from_txt_to_matrix(char **matris, int row_length, int column_length)
{
    FILE *fp = fopen("maze.txt", "r");
    int i, j;
    char c;
    for (i = 0; i < row_length; i++)
    {
        for (j = 0; j < column_length + 1; j++)
        {
            c = fgetc(fp);
            matris[i][j] = c;
        }
    }

    fclose(fp);
}
```

Matrisi Okuduktan Sonra Ekrana Yazdıralım.

```
**************** NEW RUN TIME***********


+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       |       |                                                       |
+ +-+ + +-+-+ +-+-+-+-+-+-+-+-+-+-+ +-+-+-+ +
|   |   |     |   |   |           |   |     |
+-+ + +-+ +-+-+ + + + + +-+-+-+ +-+ + +-+-+
|   | | |   |     |   | | |     | | |   |   | |
+ +-+ + +-+ +-+-+-+-+-+-+ +-+-+-+ +-+-+-+ +-+
| | |   | | | |           | | |   |     | |   |
+ +-+-+ +-+ + +-+-+-+ + + + + +-+ + +-+-+ +
|     | | |   |   |   | | | |   |   | |       |
+ +-+ + + +-+-+ +-+ + +-+-+-+-+ + +-+-+-+-+-+
| |   | | |     |   | |         | |       | |
+ + +-+ +-+-+-+-+-+ + +-+-+-+-+ + +-+ +-+ +-+
| | |               |         | |   | | |   |
+ + + +-+-+-+-+-+-+-+ +-+-+ + +-+ + + +-+ +
| |     |   |               | |   | |   |   |
+ +-+-+ + + + +-+ +-+-+ +-+-+ + + +-+-+ + +-+
| |   |   | | | |     | | |   |   | |   |
+ +-+ +-+ + + + +-+-+-+-+-+ + +-+ + + +-+-+
|       | | | |           | | |   | |     |
+-+-+-+ +-+ + +-+-+-+-+-+ + + +-+-+-+-+-+ +
|b          |           |   |         |   c|
+---+-+-+-+-+---+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

2-) Elmaları Yerleştirme:

```c
void elmalari_yerlestir(char **matrix, int row, int column)
{
    int target_elma_sayisi = 20;
    int tmp_row, tmp_col;
    // if (row > column)
    // {
    //     target_elma_sayisi = row / 2;
    //     target_elma_sayisi = target_elma_sayisi / 2;
    // }
    // else
    // {
    //     target_elma_sayisi = row / 2;
    //     target_elma_sayisi = target_elma_sayisi / 2;
    // }

    printf("\n Target Elma -> %d", target_elma_sayisi);
    int current_elma = 0;

    while (current_elma != target_elma_sayisi)
    {
        tmp_row = rand() % row;
        tmp_col = rand() % column;
        if (matrix[tmp_row][tmp_col] == ' ')
        {
            matrix[tmp_row][tmp_col] = 'O';
            current_elma++;
        }
    }
}
```

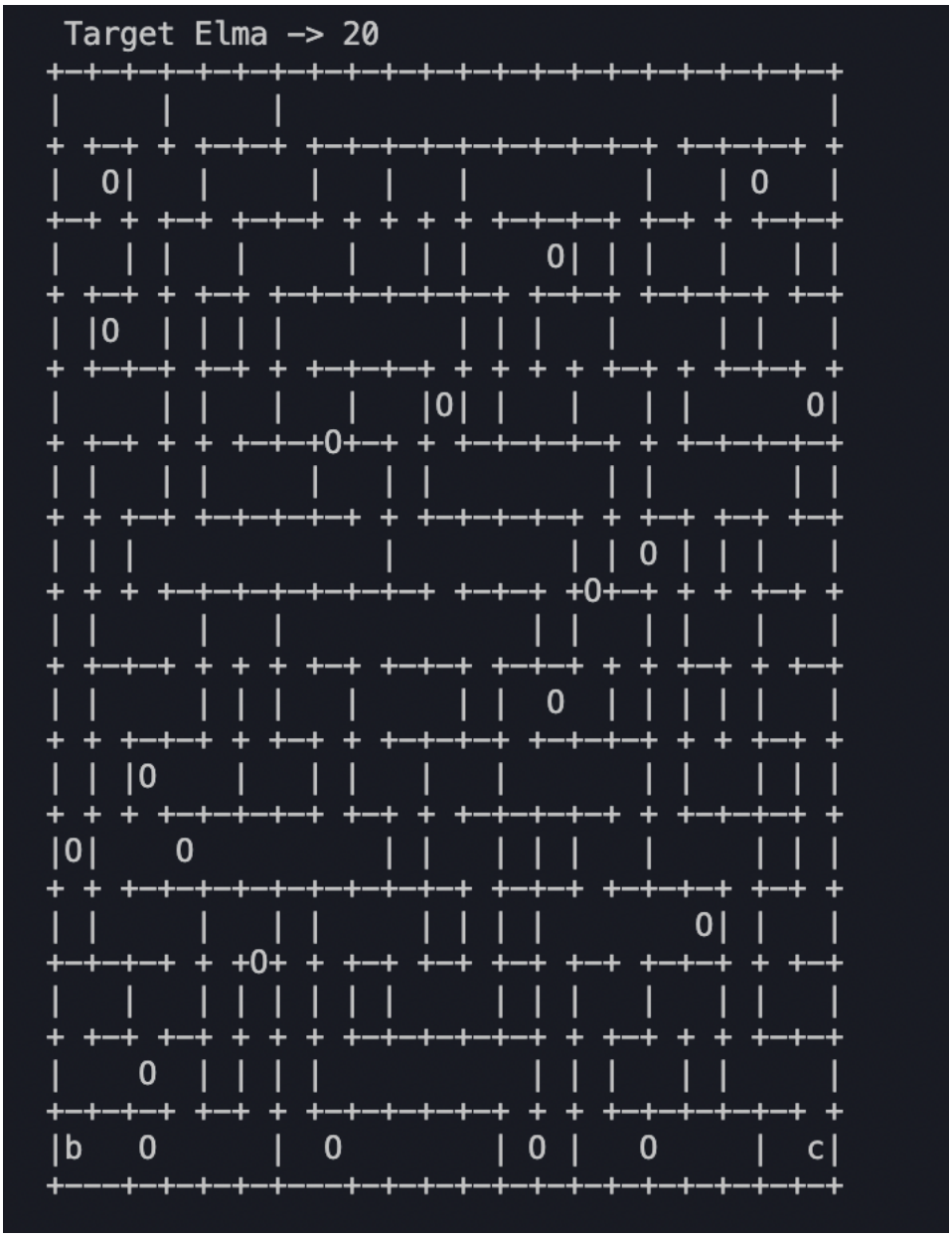
Burda videoda Kullandığım elma sayısını hardcode ladım.

Ama kullanıcı dilerse yorum satırları kaldırıp 100% Dinamik olur.

Yerleştirme Mantığı: matrisin sınırları içerisinde bir random göz seç ve seçilen göz boş ise bir elma yerleştir.

Yerleşen elma Sayısını 1 arttır. Aynı işlemi Yerleştirmek istediğimiz Elma Sayısı kadar devam ettir.

Elmaları Yerleştirdikten sonra Matris:



3-) DFS ALGORİTMASI:

ilk önce AŞAĞI sonra SAĞ sonra YUKARI sonra SOL u kontrol ediyor Boş yol buldukça O tarafa ilerler.

(burda yukarı için belirtilen kontrol her yön için ayrı ayrı yapılmıştır olup birtanesi gösterilmiştir.)

```
if (start_col + 1 != mtr_ln[1] && bulundu != 1)
{
    if (matrix[start_row][start_col + 1] == ' ' || matrix[start_row][start_col + 1] == 'c' || matrix[start_row][start_col + 1] == '0')
    {                                                                                    <- yukarı Boş veya elma var ise
        cikmazSayisi = 1;

        if (matrix[start_row][start_col + 1] == '0')    <- Elma var ise Puan Arttır
            *toplam = *toplam + 10;

        if (matrix[start_row][start_col] != 'b')
            matrix[start_row][start_col] = '*';         <- Geçtiğimiz yol u işaretle

        bulundu = dfs(matrix, mtr_ln, start_row, start_col + 1, toplam);   <- bir Daha dfs Çağır
        if (bulundu == -1)
        {
            matrix[start_row][start_col] = ' ';         <-Eğer Çıkış Bulunmadıysa Dönüş yapıyoruz demektir
            print_matrix(matrix, mtr_ln[0], mtr_ln[1], *toplam);          ve geçtiğimiz yolu temizle
        }
    }
}
```

*Eğer Elma Üzerine geçersek puan Arttırılır.

*Eğer çıkış bulmayıp ve dfs çağırışı biterse bu demektir ki deadend e geldik ve yolu temizlememiz gerekir.

(Algoritmanin Detayları videoda Açıklanmıştır.)

Önemli Not: Başlangıç Değeri Hardcodelanmıştır!

https://www.youtube.com/watch?v=8KV7Dc0z11M