Labirent Yol Bulma

Serhat KAÇMAZ
Bilişim Sistemleri Mühendisliği
Kocaeli Üniversitesi
İstanbul, Türkiye

***Özet*— Bu çalışmada belirli bir text dosyası içerisinde bazı özel karakterler ile doldurulmuş labirent, programa girdi olarak girer ve program labirentteki başlangıç noktasından bitiş noktasına kadar olan en iyi yolu bulup çözülmüş formatta bir text dosyasına yazar.(özet)**

Kelimeler—labirent, klasik, güçlendirilmiş, başlangıç, bitiş (*kelimeler****)***

I. GİRİŞ

Labirent çözmek için birden fazla algoritma geliştirilmiştir. Bu algoritmalardan birkaçı olan Deaf-end filling, Trémaux's, recursive gibi algoritmalar vardır. Bu projede labirenti çözmek için özyinelemeli algoritma kullanarak sonuca ulaşmak hedeflenmektedir. Özyinelemeli algoritma kendi kendini çağırarak kullanılan fonksiyondur. Özyinelemeli algoritmanın kullanılma mantığı ise başlangıç koordinatlarından başlayarak labirent üstünde sol, sağ, yukarı ve aşağı olmak üzere dört yöne gitmeye çalışır. Gidilen yolları hafızasında tutar ve sonunda varış noktasına gelince sonlanır ve doğru yolu bulur.

II. YÖNTEM

A. Klasik Labirent

Klasik bir labirenti çözmek için kullanılan yöntemde labirentin başlangıç ve bitiş koordinatları elde edilir, daha sonrasında programda kullanılan numaralandırma işlemine yarayan özyinelemeli fonksiyon ile başlangıç noktasından varış noktasına kadar gidilebilen yollarda numaralandırma işlemi yapılır. Bu numaralandırma işlemi ile tüm yollar numaralandırılmış olur. Bu numaralandırma işlemi belirli bir yön sırasına göre olur ve başlangıç koordinatındayken başlangıç noktasına bir rakamı verilerek başlar. Bu numaralandırmada kullanılan yön sırası ise sol, yukarı, sağ ve aşağı yönlerine sırasıyla bakma ile olur. Bulunduğun hücrede iken yön sıralamasına göre hücrelere gitmeye başla ve her seferinde gezilmemiş bir hücreye gelince numara değeri bir değer arttırılmalıdır. Bir hücreden diğer hücreye geçme işleminde çıkmaz bir hücreye yani sola, sağa, yukarı ve aşağı hareket edememe durumunda kalınırsa ise bu son hücreye gelmeden önceki hücreye döner ve tekrar yön kontrolüne bakılır eğer yine yeni bir hücreye gidilemese yine bir önceki hücreye dön şeklinde devam eder. Ve bu işlemler sonucunda labirentte bulunan yollar ve başlangıç ve bitiş noktası numaralandırılır. Daha sonrasında bitiş ile varış arasındaki yolu bulmak için yolu bulan özyinelemeli fonksiyona bitiş noktasının koordinatları yollanarak ve bitiş noktasının numaralandırma işleminden sonra oluşan numarası parametre olarak yollandığında, varış hücresinin sol, sağ, aşağı ve yukarı hücrelerine bakılır ve bitiş hücresinin numarasından bir eksik ise yol olarak işlenir. Daha sonrasında işlenen hücreye geçilir. Ve bu işlem numara değerinin yani başlangıca kadar gelinmesinde geçilen hücreler yol olarak bir rakamı ile işlenir. Ve labirent yolu başlangıcından bitişe kadar bir rakamı ile işlenmiş hale gelir. Ve gidilmeyen yollar ve duvarların yerine sıfır rakamı işlenir. Ve daha sonrasında oluşan sıfır ve bir rakamlarından oluşan labirent çıktı dosyasına kayıt edilmesinden sonra program sonlanır.

B. Güçlendirilmiş Labirent

Güçlendirme noktası ile gelen labirent çözümünde ise işlenen yöntem klasik labirenti çözme yöntemiyle aynı mantıktadır. Ama bu bölümde işlenen yöntemde öncellikle başlangıç noktasından güçlendirme noktasına kadar olan en iyi yol bulunur ve bulunan kısımdaki koordinatlar bir rakamı ile işlenir. Daha sonrasında güçlendirme noktasını sanki yeni bir başlangıç noktası gibi alıp tekrardan labirent üstündeki yollarda numaralandırma işlemi yapılır. Ve bu seferde güçlendirme noktasından bitişe olan yol bulunur. Ve bir rakamı ile işlenir. Özetle güçlendirme noktalı olarak gelen bir labirentin çözülme mantığı başlangıç noktasından güçlendirme noktasına, güçlendirme noktasından varış noktasına şeklinde bir yol izlenmektedir. Daha sonra sıfır ve bir rakamlarından oluşan labirent çıktı dosyasına kayıt edilmesinden sonra program sonlanır.

III. YALANCI KOD

Yalancı kod, herhangi bir programlama diline bağlı kalmadan sözler ile bir programın tasarlanması diyebiliriz. Yalancı kod yazımında programlama dilinde geçen ifadeler kullanılmadan yazılır. Yalancı kod yazımında mantıksal sırasıyla yapılacak işlemler yazılır. Bu projenin yalancı kodu;

- Argümanları al
- Argüman eksik veya fazla ise bitir
- Girilen text Dosyasını oku
- Girilen text dosyasını labirent listesine at
- Başlangıç ve bitiş noktalarını bul
- Eğer girdi guclu_girdi.txt ise H koordinatını al
- Değilse alma
- Eğer girdi.txt dosyası geldiyse

- Yolları numaralandıran metoda başlangıç koordinatlarını yolla
- Yolu bulan bir fonksiyon ile bitiş noktasının koordinatlarını yolla
- Yol bulunduktan sonra yolu '1' diğerlerini '0' yap
- Labirentin çözümlü halini cikti.txt dosyasına yaz ve bitir
- Eğer guclu_girdi.txt dosyası geldiyse
- Yolları numaralandıran metoda başlangıç koordinatlarını yolla
- Yolu bulan metoda H' ın koordinatını yolla
- Yolu bulunduktan sonra yolu '1' yap
- Sonra Labirentin içinde numaralanan hücreleri eski haline getir
- Sonra Yolları numaralandıran metoda H' ın koordinatlarını yolla
- Yolu bulan metoda bitiş noktasının koordinatlarını yolla;
- Yolu bulduktan sonra yolu '1' yap
- Yol haricindeki koordinatlara '0' ata
- Labirentin çözümlü halini cikti.txt dosyasına yaz ve bitir

IV. ÖRNEK

Aşağıda bulunan resimlerde klasik veya güçlendirilmiş labirentin programa gönderdiği girdisi ve programın sonunda almış olduğu çıktıları gösterilmektedir.

```
WPPWWW          0, 0, 0, 0, 0, 0
WWPWPS          0, 0, 0, 0, 1, S
WWPWPW          0, 0, 0, 0, 1, 0
PPPPPW          1, 1, 1, 1, 1, 0
FWwWWW          F, 0, 0, 0, 0, 0
WPPPPW          0, 0, 0, 0, 0, 0
```

Resim 1

```
WPPWWWWW        0, 0, 0, 0, 0, 0, 0, 0
WWPWPSPW        0, 0, 0, 0, 1, S, 0, 0
WWPPPWPW        0, 0, 1, 1, 1, 0, 0, 0
PPHWWWPW        1, 1, 1, 0, 0, 0, 0, 0
PWWWWPPW        1, 0, 0, 0, 0, 0, 0, 0
FPPPPPWW        F, 0, 0, 0, 0, 0, 0, 0
```

Resim 2

Resim 1’ de klasik bir labirentin özel karakterli bulunan girdi text dosyası ve programa gönderildikten sonraki çıktı dosyasına yazılacak olan çözümlü halinin çıktısını gösterir.

Resim 2’ de Güçlendirilmiş bir labirentin özel karakterler ile oluşturulmuş girdi dosyası ve programdan sonra çıktı dosyasına yazılacak olan çözümlü halini gösterir.

REFERANSLAR

[1] https://www.codesenior.com/tutorial/Labirent-Cozme-Algoritmalari

[2] https://selcukaltintav.wordpress.com/2014/07/02/stack-veri-yapisiyla-problem-cozmeklabirent-algoritmasi/

[3] http://bryukh.com/labyrinth-algorithms/#collapse-dfs-code

[4] https://www.laurentluce.com/posts/solving-mazes-using-python-simple-recursivity-and-a-search/

[5] https://levelup.gitconnected.com/solve-a-maze-with-python-e9f0580979a1

[6] https://www.makeschool.com/academy/track/standalone/trees-and-mazes/solving-the-maze

[7] https://www.geeksforgeeks.org/python-program-for-rat-in-a-maze-backtracking-2/

[8] https://cstart.mines.edu//python/pages/projects/maze/

[9] https://www.youtube.com/watch?v=ZuHW4fS60pc

[10] https://www.techwithtim.net/tutorials/breadth-first-search/

[11] https://www.youtube.com/watch?v=ZuHW4fS60pc&t=35s