2023-2024 Veri Yapıları Ve

Algoritmaları Dersi Ödev Raporu

**Labirent (Maze) Oyunu: Backtraking Uygulaması**

**Ad-Soyad:Umut Kuzey Görgeç**

**Numara:22010903075**

|  |
| --- |
| **Algoritmanın Adımlarının Detaylı Açıklaması**  **class node{ Bu şekilde bir nodesınıfı olacak ve**  **public: içinde x, koordinatını ve dirdeğişkeninde de yönü tutacağız. Sonra yukarıdan başlayarak**  **intx.y,dir; saatin tersi yönünde tüm yönleri deneyeceğiz.dir=0 YUKARI, dir=1 SOL, dir=2 AŞAĞI, dir=3 SAĞ Başta**  **node(){ kontrol için i=0 j=0 ve dir=0 olan bir düğümü yığıta ekleyeceğiz. En üstteki düğümün tüm yönlerine saat x=0; yönünün tersine birer birer ilerleyeceğiz ve her seferinde yeni bir yol denedikçe o düğümü yığıtın içine y=0; iteceğiz.Sonrasında dir++ yaparak tüm yönleri deneyeceğiz ve dir= 4 ise o düğümü yığıtdan**  **dir=0; çıkaracağız.Buda geldiğimiz yola bir adım geri çekilmek gibi oluyor.**  **}**  **node(int i,int j){ Hangi yolların denendiğini tutmak için ise char visited[Uzunluk][Genişlik] dizisini**  **x=i; kullanacağız.Herhangibir yönü denerken çıkmaz sokak olup olmadığını ve labirentin dışında olup**  **y=j; olmadığını kontrol edeceğiz.Bunu ya en üstteki düğümün koordinatları bitiş koordinatlarına eşit olana dir=0;} kadar ya da yığıt boşalana kadar yani bitişe ulaşmamızın artık hiçbir yolu kalmayana kadar devam**  **}; edeceğiz.**  **Son olarak Harita.txt isimli dosyadan okuma yapılarak charmaze[Uzunluk][Genişlik] dizisine aktarılıp dir değişkeninde göre semboller eklenecek.Bu arada Harita.txt dosyası okunurken ‘S’ start yani başlangıç olacak ‘F’ finish yani bitiş olacak ve bunlar node S ve node F olarak tanımlanıp konumları belirlendikten sonra boş ‘ ’ karakterle değiştirilecek.Labirentimizin duvarları ‘#’ karakteriyle boşluklar ‘ ’ boşluk karakteriyle temsil edilmektedir.** |

C++ Kodları ve Dosyaların Açıklanması

**Dosyaların açıklanması**

**Header.hpp**

**#ifndef HEADER\_HPP**

**#define HEADER\_HPP**

**#include <iostream> Burada kütüphaneler ekleniyor ve bu headerı kullanan cpp dosyaları bu kütüphaneleri kullanabilecek.**

**#include <stack>**

**#include<unistd.h>**

**#include <conio.h>**

**#include <cstring>**

**#define R 100 Satırı(Row) Sütünu (Column) R ve C ile temsil ettim.**

**#define C 100**

**class node { node classını ve constructorlarını tanımladım.**

**public:**

**int x,y;**

**int dir;**

**node(int i, int j);**

**node();**

**};**

**void SetStartingPosition(int &i,int &j); Kullanacağım fonksiyonları tanımladım.**

**void SetEndingPosition(int &fx,int &fy);**

**void readMaze(FILE\* fp);**

**void writeMaze(FILE\* fp);**

**void printMaze();**

**bool isReachable(char maze[R][C], std::stack<node>& path);**

**void printPath(std::stack<node> path, FILE\* fp);**

**bool FindStartAndFinishPositions(char maze[R][C]);**

**extern int n, m, row; Diğer dosyalarımda tanımlayacağım değişkenlerimi belirttim.**

**extern char maze[R][C];**

**extern int i,j;**

**extern int fx, fy;**

**extern bool visited[R][C];**

**extern node S;**

**extern node F;**

**#endif**

C++ Kodları ve Dosyaların Açıklanması

**Dosyaların açıklanması**

**Konum.cpp**

**#include "header.hpp" Header eklendi.**

**using namespace std;**

**node S; Start ve Finish nodeları tanımlandı.**

**node F;**

**int i\_start = S.x; S ve F nin x ve y koordinatları değişkenlere tanımlandı.**

**int j\_start = S.y;**

**int fx\_end = F.x;**

**int fy\_end = F.y;**

**node::node(){ headerda tanımladğım constructoru yazdım.**

**x = 0;**

**y = 0;**

**dir = 0;**

**}**

**void SetStartingPosition(int& i, int& j) { Başlangıç ve bitiş değerlerinin belirlenmesi için fonksiyonlar headerda tanımladığım I ve j ye S ve F nin koordinatlarını aktarıyorlar.**

**i = S.x;**

**j = S.y;**

**}**

**void SetEndingPosition(int& fx, int& fy) {**

**fx = F.x;**

**fy = F.y;**

**}**

**bool FindStartAndFinishPositions(char maze[R][C]) { Bu fonksiyon tüm diziyi alıp tarayıp S ve F yi bulup koordinatlarını eşitliyor.**

**bool foundStart = false;**

**bool foundFinish = false;**

**for (int t = 0; t < n; t++) {**

**for (int y = 0; y < m; y++) {**

**if (maze[t][y] == 'S') {**

**S.x = t; Burada koordinatlar eşitlendi.**

**S.y = y;**

**maze[t][y]=' ';**

**foundStart = true;**

**}**

**if (maze[t][y] == 'F') {**

**F.x = t; Burada koordinatlar eşitlendi.**

**F.y = y;**

**maze[t][y]=' ';**

**foundFinish = true;**

**}**

**if (foundStart == true && foundFinish == true) { İki bool da true olursa yani S ve F bulunursa true döndürecek.**

**return true;**

**}**

**}**

**}**

**cout << "Hata: 'S' veya 'F' karakterleri bulunamadi." << endl; S ve F’nin bulunamaması durumunda hata yazısı.**

**exit(1);**

**}**

C++ Kodları ve Dosyaların Açıklanması

**Dosyaların açıklanması**

**labirent.cpp**

**#include "header.hpp" header eklendi.**

**#include <conio.h>**

**using namespace std;**

**int n = R;**

**int m = C; n:satır m:sütun i:başlangıç x’i j:başlangıç y’si fx:bitiş x’i fy:bitiş y’si**

**int fx,fy;**

**int i,j;**

**bool visited[R][C]; visited dizisi tanımlandı.**

**node::node(int i, int j) {**

**x = i; headerda tanımladğım constructoru yazdım.**

**y = j;**

**dir = 0;**

**}**

**bool isReachable(char maze[R][C], stack<node>& path) { maze dizisini ve path isimli stack’I tanımladım.**

**memset(visited, true, sizeof(visited)); visited dizisini true yapıyorum.**

**FindStartAndFinishPositions(maze);**

**SetStartingPosition(i,j); S ve F karakterlerini arayıp sonra o değerleri i,j ve fx,fy değişkenlerine veriyorum**

**SetEndingPosition(fx,fy);**

**stack<node> s; s stackini tanımlayıp geçici temp node’unda saklıyorum.**

**node temp(i, j);**

**s.push(temp);**

**while (!s.empty())**

**{**

**temp = s.top(); En üstteki düğümü tempe atıyorum.Düğümün yönlerini i ve j‘ye veriyorum.**

**int d = temp.dir;**

**i = temp.x, j = temp.y;**

**temp.dir++; Yönü bir arttırıp en üstteki değeri pop ile çıkarıp tempi ekliyorum.**

**s.pop();**

**s.push(temp);**

**if (i == fx and j == fy) Son düğümün koordinatları fx ve fy ye eşit ise true döndürüyor.**

**{**

**while (!s.empty()) Eğer s stacki boş değilse path stackine en üstteki top değer eklenir ve tek değer o olduğundan sonrasında pop ile silinir.**

**{**

**path.push(s.top());**

**s.pop();**

**}**

**return true;**

**} Burada yönleri control edicez ve gidilmemişse visited dizisinin o andaki konumundaki elemanlarını false yapıcaz.**

**if (d == 0) Yukarı’yı kontrol ediyoruz.**

**{**

**if (i - 1 >= 0 and maze[i - 1][j] == ' ' and visited[i - 1][j])**

**{**

**node temp1(i - 1, j);**

**visited[i - 1][j] = false;**

**s.push(temp1);**

**}**

**}**

**else if (d == 1) Sol’u kontrol ediyoruz.**

**{**

**if (j - 1 >= 0 and maze[i][j - 1] == ' ' and visited[i][j - 1])**

**{**

**node temp1(i, j - 1);**

**visited[i][j - 1] = false;**

**s.push(temp1);**

**}**

**}**

**else if (d == 2) Aşağı’yı kontrol ediyoruz.**

**{**

**if (i + 1 < n and maze[i + 1][j] == ' ' and visited[i + 1][j])**

**{**

**node temp1(i + 1, j);**

**visited[i + 1][j] = false;**

**s.push(temp1);**

**}**

**}**

**else if (d == 3) Sağ’I control ediyoruz.**

**{**

**if (j + 1 < m and maze[i][j + 1] == ' ' and visited[i][j + 1])**

**{**

**node temp1(i, j + 1);**

**visited[i][j + 1] = false;**

**s.push(temp1);**

**}**

**}**

**else**

**{**

**visited[temp.x][temp.y] = true; Burada eğer hiçbir yön çıkışa ulaştıramıyorsa visited dizisinin o andaki konumundaki elemanlarını true yap.**

**s.pop(); s stackini temizle.**

**}**

**}**

**return false; eğer stack boşsa ve hiçbir yol bulunamadıysa false döndürür.**

**}**

C++ Kodları ve Dosyaların Açıklanması

**Dosyaların açıklanması**

**main.cpp**

**#include "header.hpp" header eklendi.**

**using namespace std;**

**int row = 0; headerda tanımladığım row ve char maze’I yazdım.**

**char maze[R][C];**

**void readMaze(FILE\* fp) { labirenti okumak için FILE tipinde bir pointer kullanıyorum.**

**if (fp == NULL) fp ==NULL ise dosya açma hatası veriyor ve ekranda bir tuşa basıldığında programdan çıkıyor.**

**{**

**cout << "Dosya açma hatasi" << endl;**

**getch();**

**exit(1);**

**}**

**while (fgets(maze[row], C, fp) != NULL) dosyada gezerek labirentin kaç satır olması gerektiğini buluyoruz ve row++ ile row değişkenine kaydediyoruz.**

**{**

**row++;**

**}**

**}**

**void writeMaze(FILE\* fp) { burada dosyada değişklik yapmak için**

**if (fp == NULL) burada da aynı şekilde fp ==NULL ise dosya açma hatası veriyor ve ekranda bir tuşa basıldığında programdan çıkıyor.**

**{**

**cout << "Dosya açma hatası" << endl;**

**getch();**

**exit(1);**

**}**

**fseek(fp, 0, SEEK\_SET); dosyanın başlangıcını belirttim ki dosya okumaya başlandığında hep baştan başlasın.**

**for (int a = 0; a < row; a++) dosyayı bir döngüyle tarıyoruz ve readMaze fonksiyonuyla bulduğumuz row yani satır sayısı kadar işlemden sonra dosyamızdaki labirent maze dizimize**

**{ aktarılmış oluyor.**

**fputs(maze[a], fp);**

**}**

**}**

**void printMaze() {**

**system("cls"); bu kodu her yaptığım değişiklikte ekranı temizlemek için kullanıyorum.**

**for (int b = 0; b < row; b++)**

**{**

**cout << maze[b]; bu diziyi ekrana yazdırmak için.**

**}**

**cout << endl**

**<< endl;**

**}**

**void printPath(stack<node> path, FILE\* fp) { labirent.cpp’deki isReachable fonksiyonunda oluşturup içini doldurduğum path stackini yazdırmak için kullanılıyor.**

**while (!path.empty()) {**

**node current = path.top(); eğer stack boş değilse en üstteki değeri current yani şu anki konuma atacak.**

**char pathSymbol = ' '; pathSymbol ile gittiğim yönlere yönümü ekranda belirten işaretler koyacağım ama default yani varsayılan olarak ‘ ’ boş karakter koyuyorum.**

**if (current.dir == 4) { böylece semboller önce ekranda yazılacak sonra boş yapılacak.**

**pathSymbol = '>';**

**} else if (current.dir == 1) { normalde 0 yukarı 1 sol 2 aşağı 3 sağ idi ama biz isReachable fonksiyonunda en başta daha kontrolleri yapmadan önce dir++ yaptığımızdan**

**pathSymbol = '^'; dolayı bende yönleri bir fazlasına göre düzenledim.**

**} else if (current.dir == 2) {**

**pathSymbol = '<';**

**} else if (current.dir == 3) {**

**pathSymbol = 'v';**

**}**

**maze[current.x][current.y] = pathSymbol; Burada sembolü ilgili koordinata veriyorum ve yazdırıyorum.**

**writeMaze(fp);**

**printMaze();**

**sleep(1); burada sleep bekleme süresi koydum.**

**maze[current.x][current.y] = ' '; ve boşaltma işlemini gerçekleştiriyorum sonrasında da koyduğum sembolün olduğu düğümü boşaltıyorum ki bellekte boş yer kaplamasın.**

**path.pop();**

**}**

**}**

**int main()**

**{**

**stack<node> path;**

**FILE\* fp = fopen("Harita.txt", "r+"); path stackini tanımladım.fp pointerimi tanımladım ve r+ yani hem yazma hem okuma modunda açtım.**

**readMaze(fp); dosyayı okudum.**

**if (isReachable(maze, path)) isReachable ile çıkış olup olmadığını control ediyorum.**

**{**

**printPath(path,fp); Çıkış varsa doğru yolu yazdırıyor.**

**}**

**else**

**{ Çıkış Yoksa çıkış bulunamad**

**cout << "Cikis bulunamadi. <<endl;**

**}**

**fclose(fp); dosyayı kapatıp herhangi bir tuşa basılınca çıkış bekiyoruz.**

**getch();**

**return 0;**

**}**

Örnek Labirent

**Örnek Bir Harita.txt Ve Çözümü**

###########################################

S ### #

############## ### ##### ############### ##

# # ## ##

######################## ##################

# #

#### ####################################

# #

########## ################################

########## ############################## F

# #

##########################################



Labirent çözümün içeren dosya açılmazsa bu dosya Video klasörünün içinde de var.