



DATA

STRUCTURES

PROJECT REPORT

Creation and Solving of Virtual Maze
with Computer AI

By

Harsh Tekriwal 18103219

Tarush Goyal 18103238

Rupam Rajan 18103339

Batch-B7

CONTENTS

1. Problem Statement

2. Introduction

**3. List of Data Structures used in the
project**

4. Detailed Design of the project

5. Implementation details and results

6. Conclusion

PROBLEM STATEMENT

In today's world, there is a growing need of finding the **shortest path** between a source and destination. This helps in saving time, money and utilization of manpower. Many modern softwares and technologies use this implementation to solve their problems such as Facebook, Google maps, Uber etc. We solve this problem in our project by using **Graphs and Breadth First Search** to find the shortest path. We also print this shortest path and save it in an individual file for further use.

INTRODUCTION

The use of Graphs in the modern era is very diverse and versatile. It is used by social networking sites such as Facebook, Instagram etc. to pair people with their mutual friends, known associates or distant relatives. Navigation apps such as Google Maps and Uber use this to find the shortest path between source and destination. This motivated us in making a project based on this concept and making something that is widely used in today's industry and will help us in learning key concepts which may help us in our future jobs.

DATA STRUCTURES

Following Data Structures have been used in our project:

- **Linked List**
- **Graphs**
- **Standard Template Library (STL)**
- **File Handling**

DETAILED DESIGN

We are creating a Maze Solving Application, where User will first create his / her account and then provide their maze (or mazes) created in a notepad format which will be solved by the computer using AI and solution will be saved also as a notepad format. Furthermore, each maze will be given rating by the computer based on the time taken by the computer to solve that given maze. Better the rating, the harder the maze will be to solve. Also there will be a ranking system, which will inform which user has given the hardest maze and there will also be a system to count number of mazes given by each user. We are going to use mainly Shortest Path Algorithms among other things for our computer's AI in our project.

CODE IMPLEMENTATION

```
#include<fstream>
#include<iostream>
#include <cstring>
#include<algorithm>
#include<cstdlib>
#include<conio.h>
#include<string.h>
#include <chrono>
#include<queue>
#include<string>
#include<stdio.h>
#include<errno.h>
#include<dirent.h>
#include<stdlib.h>
#include<windows.h>
#include<vector>

using namespace std;
using namespace std::chrono;

int rownumber;
int columnnumber;

class vertex{
public:
    int row;
    int column;
};

struct Ranksystem{

    char name[20];
    char mazenname[20];
    int time;
};

bool compareTime(Ranksystem r1, Ranksystem r2)
{
    return (r1.time > r2.time);
```

```

}
struct player{

    char name[20];
    char pass[20];
}currentData;


void loader()
{
    int i;
    system("cls");
    cout<<"\n \t \t \t \t PLEASE WAIT LOADING.....";
    system("COLOR 0");
    cout<<"\n \n \n \t \t \t ";
    for(i=0; i<60; i++)
    {
        Sleep(50);
        printf("%c",219);
    }

}

void reg()
{

    system("CLS");
    FILE *fp;
    struct player pin;
    fp=fopen("DATA.TXT","ab+");
    if(fp==NULL)
    {
        cout<<"\n UNABLE TO REGISTER YOUR ACCOUNT PLEASE TRY LATER";
        getch();
        return;
    }

    cout<<"\n Enter Username:";
    cin>>pin.name;
    cout<<"\nEnter Password less than 8 characters:";
    char ch;
    int i=0;
    do
    {
        ch=getch();
        if(ch==8)
        {

```



```

        if(i==0)
        {
            continue;
        }
        cout<<"\b";
        i=i-1;
        continue;
    }
    if(i==0&&ch==13)
    {
        cout<<"\n Enter atleast 1 character for password ";
        getch();
        fclose(fp);
        return;
    }
    if(ch==13)
    {
        break;
    }
    cout<<"*";
    pin.pass[i]=ch;
    i++;
}
while(ch!=13);
if(i>7)
{
    cout<<"\n Password is too big...";
    getch();
    fclose(fp);
    return;
}
pin.pass[i]='\0';
struct player pp;
fseek(fp, 0, SEEK_SET);
while(!feof(fp))
{
    fread(&pp,sizeof(struct player),1,fp);
    if(strcmp(pin.name,pp.name)==0)
    {
        cout<<"\n An account with that Username already Exists.";
        getch();
        fclose(fp);
        return;
    }
    else
    {

```

```

        continue;
    }
}
fwrite(&pin,sizeof(struct player),1,fp);
cout<<"\n Account Successfully Registered";
cout<<"\n Press any key to continue";
if (mkdir(pin.name)!=0)
    cerr << "Error : " << strerror(errno) << endl;

else
{

}
getch();
fclose(fp);
return;
}

int login()
{
    int s=0;
    system("CLS");
    FILE *fp;
    fp=fopen("DATA.TXT","rb");
    if(fp==NULL)
    {
        cout<<"\n First Register your account";
        getch();
        return s;
    }
    char username[20];
    char password[10];
    cout<<"\n Enter your username:";
    cin>>username;
    cout<<"\n Enter your Password:";
    int i=0;
    char c;
    for(i=0; i<8; i++)
    {
        c=getch();
        if(c==13)
        {
            break;
        }
        else if(c==8)
        {

```

```

        if(i==0)
        {
            i=i-1;
            continue;
        }
        cout<<"\b";
        i=i-2;
        continue;
    }
    else
    {
        password[i]=c;
        cout<<"*";
    }
}
password[i]='\0';
while(!feof(fp))
{
    fread(&currentData,sizeof(struct player),1,fp);
    if(strcmp(currentData.name,username)==0)
    {
        if(strcmp(currentData.pass,password)==0)
        {
            getch();
            s=1;
            fclose(fp);
            cout<<"\n Login Successful";
            getch();
            system("CLS");
            return s;
        }
        else
        {
            cout<<"\n Incorrect password";
            getch();
            fclose(fp);
            return s;
        }
    }
    else
    {
        continue;
    }
}
cout<<"\n No account with that username exists..";
getch();

```

```

fclose(fp);
return s;
}

int parse(char* line,int i,char **ar,int *sr,int *sc,int *dr,int *dc)
{

char *data;
int destinationcount=0;
int sourcecount=0;
for(int index=0;index<columnnumber;index++)
{
    if(index==0)
    {
        data = strtok(line, " ");
        if(*data!='S'&&*data!='D'&&*data!='.'&&*data!='#'){
            return -1;
        }
        ar[i][index]=*data;
        if(ar[i][index]=='S'){
            *sr=i;
            *sc=index;
            sourcecount++;
        }
        else if(ar[i][index]=='D'){
            *dr=i;
            *dr=index;
            destinationcount++;
        }
    }
    else
        data = strtok (NULL, " ");
    if(*data!='S'&&*data!='D'&&*data!='.'&&*data!='#'){
        return -1;
    }
    ar[i][index]=*data;
    if(ar[i][index]=='S'){
        *sr=i;
        *sc=index;
        sourcecount++;
    }
    else if(ar[i][index]=='D'){
        *dr=i;
        *dc=index;
        destinationcount++;
    }
}

```

```

    }

    return 0;
}

bool isValidvertex(int row,int column){
    if((row>=0&&row<=rownumber-1)&&(column>=0&&column<=columnnumber-1)){
        return true;
    }
    else {
        return false;
    }
}

int shortestpath2(char **ar,int sourcerow,int sourcecolumn,int destinationrow,int destinationcolumn ,char **result){
    bool visited[rownumber][columnnumber]={false};
    ar[sourcerow][sourcecolumn]='.';
    ar[destinationrow][destinationcolumn]='.';
    vertex start;
    start.row=sourcerow;
    start.column=sourcecolumn;
    visited[sourcerow][sourcecolumn]=true;
    queue<vector<vertex>>path_collection;
    vector<vertex> path;
    path.push_back(start);
    path_collection.push(path);
    vertex vert;
    vertex temp;
    while(!path_collection.empty()){
        path=path_collection.front();
        path_collection.pop();
        vert=path[path.size()-1];
        if(vert.row==destinationrow&&vert.column==destinationcolumn){
            for(int i=0;i<path.size();i++){
                result[path[i].row][path[i].column]='.';
            }
            return 1;
        }
        if(isValidvertex(vert.row,vert.column+1)&&ar[vert.row][vert.column+1]!='.'&&!visited[vert.row][vert.column+1]){
            temp.row=vert.row;
            temp.column=vert.column+1;
            vector<vertex> newpath(path);
            newpath.push_back(temp);
            path_collection.push(newpath);
            visited[temp.row][temp.column]=true;
        }
        if(isValidvertex(vert.row+1,vert.column)&&ar[vert.row+1][vert.column]!='.'&&!visited[vert.row+1][vert.column]){

```

```

        temp.row=vert.row+1;
        temp.column=vert.column;
        vector<vertex> newpath(path);
        newpath.push_back(temp);
        path_collection.push(newpath);
        visited[temp.row][temp.column]=true;
    }
    if(isValidvertex(vert.row,vert.column-1)&&ar[vert.row][vert.column-1]!='.'&&!visited[vert.row][vert.column-1]){
        temp.row=vert.row;
        temp.column=vert.column-1;
        vector<vertex> newpath(path);
        newpath.push_back(temp);
        path_collection.push(newpath);
        visited[temp.row][temp.column]=true;
    }
    if(isValidvertex(vert.row-1,vert.column)&&ar[vert.row-1][vert.column]!='.'&&!visited[vert.row-1][vert.column]){
        temp.row=vert.row-1;
        temp.column=vert.column;
        vector<vertex> newpath(path);
        newpath.push_back(temp);
        path_collection.push(newpath);
        visited[temp.row][temp.column]=true;
    }
}

return 0;

}

```

```

void startprogram(){

    system("cls");
    char location[100];
    strcpy(location,"");
    strcat(location,currentData.name);
    cout<<"ENTER NAME OF THE MAZE YOU WANT TO SOLVE :";
    char mazename[20];
    cin>>mazename;
    strcat(location,"/");
    strcat(location,mazename);
    strcat(location,".txt");

    char location2[100];
    strcpy(location2,"");
    strcat(location2,currentData.name);
}

```

```

strcat(location2, "/");
strcat(location2, mazename);
strcat(location2, "sol.txt");

if(FILE *f=fopen(location2, "r")){
    cout<<"THIS MAZE HAS ALREADY BEEN SOLVED...";
    fclose(f);
    return;
}

FILE *file = fopen(location, "r");
if(!file){
    cout<<"NO SUCH MAZE EXISTS...";
    getch();
    return;
}

int count=0;
char line2[256];
char line3[256];
while(fgets(line2, sizeof(line2), file))
{
    if(count==0){
        strcpy(line3, line2);
    }
    count++;
}
int count2=0;

for(int i=0; line3[i]!='\0'; i++){
    if(line3[i]!=' '){
        count2++;
    }
}

rownumber=count;
columnnumber=count2-1;
file = fopen(location, "r");

char **ar=new char*[rownumber];
for(int i=0; i<rownumber; i++)
{
    ar[i]=new char[columnnumber];
}

```

```

        char **result=new char*[rownumber];
for(int i=0;i<rownumber;i++)
{
    result[i]=new char[columnnumber];
}

char line[256];
int index = 0;
int sourcerow;
int sourcecolumn;
int destinationrow;
int destinationcolumn;

while (fgets(line, sizeof(line), file)) {
    int res=parse(line,index,ar,&sourcerow,&sourcecolumn,&destinationrow,&destinationcolumn);
    if(res==-1){
        cout<<"\n Your maze is not in the proper format";
        return;
    }
    index++;
}

fclose(file);
cout<<"\n Your Maze is\n";
for(int i=0;i<rownumber;i++){
    for(int j=0;j<columnnumber;j++){
        cout<<ar[i][j];
        cout<<" ";
    }
    cout<<"\n";
}
for(int i=0;i<rownumber;i++){
    for(int j=0;j<columnnumber;j++){
        result[i][j]=' ';
    }
}

auto start = high_resolution_clock::now();
int res=shortestpath2(ar,sourcerow,sourcecolumn,destinationrow,destinationcolumn,result);
auto stop = high_resolution_clock::now();
auto duration = duration_cast<nanoseconds>(stop - start);

ofstream rankfile;
rankfile.open("rank.txt",std::ios_base::app);
rankfile<<currentData.name;

```



```

rankfile<<" ";
rankfile<<mazename;
rankfile<<" ";
rankfile<<duration.count();
rankfile<<"\n";
rankfile.close();
getch();

if(res==0){
    cout<<"The Maze Doesn't have any solution";
    getch();
}
if(res==1){

cout<<"\n The fastest solution to the specified maze is:-\n";
    result[sourcerow][sourcecolumn]='S';
    result[destinationrow][destinationcolumn]='D';
    for(int i=0;i<rownumber;i++){
        for(int j=0;j<columnnumber;j++){
            cout<<result[i][j];
            cout<<" ";
        }
        cout<<"\n";
    }

    ofstream file2;

    file2.open (location2);
    if(!file2){

    }
    for(int i=0;i<rownumber;i++){
        for(int j=0;j<columnnumber;j++){
            file2<<result[i][j];
            file2<<" ";
        }

        file2<<"\n";
    }

    file2.close();
    cout<<"\n Time taken to solve your maze is :";
    cout << duration.count() << endl;
return;
}

```

```
}
```

```
Ranksystem parse2(char* line)
```

```
{
```

```
    char *data;
```

```
    Ranksystem temp;
```

```
        data = strtok(line, " ");
```

```
        strcpy(temp.name,data);
```

```
        data=strtok(NULL," ");
```

```
        strcpy(temp.mazename,data);
```

```
        data=strtok(NULL," ");
```

```
        temp.time=atof(data);
```

```
        return temp;
```

```
}
```

```
void ranking(){
```

```
    char line[256];
```

```
    vector<Ranksystem> v;
```

```
    FILE *file = fopen("rank.txt", "r");
```

```
    if(!file){
```

```
        cout<<"cannot open file";
```

```
        getch();
```

```
        return;
```

```
    }
```

```
    while (fgets(line, sizeof(line), file)) {
```

```
        v.push_back(parse2(line));
```

```
    }
```

```
    fclose(file);
```

```
    sort(v.begin(),v.end(),compareTime);
```

```
        system("CLS");
```

```
        cout<<"HARDEST MAZES TO SOLVE ACCORDING TO TIME COMPLEXITY \n";
```

```
        int index=1;
```

```
        for (auto x : v){
```

```
            cout<<index << " ";
```

```
            cout << x.name <<" ";
```

```
            cout<< x.mazename <<" ";
```

```
            cout<<x.time<<"\n";
```

```
            index++;
```

```
        }
```

```
}
```

```
int main(){
```

```
    loader();
```

```
    system("CLS");
```

```
    cout<<"\n \n \n \t \t  CONTENT LOADED SUCCESSFULLY";
```

```
    cout<<"\n\n Press Any Key to Continue";
```

```

    getch();
    int ch;
    do
    {
        system("cls");
        cout<<"1.REGISTER: \n";
        cout<<"2.LOGIN: \n";
        cout<<"3.RANKINGS : \n";
        cout<<"4.HELP:\n";
        cout<<"5.EXIT:\n";
        cout<<"Enter your choice : \n";
        cin>>ch;
        switch(ch)
        {
        case 1:{
            reg();
            break;
        }
        case 2:
        {
            int a=login();
            if(a==1)
            {
                startprogram();

```

```

        getch();
    }
    break;
}

```

```

case 3:
{

    ranking();
    getch();
    break;
}

```

```

case 4:
{
    system("cls");
    cout<<"\t This program is a maze solver which is used to solve any given maze of m x n format. First you have to
register your account";

```

```
cout<<"\n \t and a folder of your name will be created. Then you have to create a maze in a .txt format of M X N and specify a source using S";
```

```
cout<<" \n \t and a destination using D. # MEANS BLOCKAGE AND . means Path. Put space between every character. No other character shall be used";
```

```
cout<<"\n \t Once you login , computer will ask you to enter the name of the maze you want to solve and computer will solve it and create a";
```

```
cout<<" \n \t solution.txt file of that particular maze. A Sample maze has been given in the folder.";
```

```
cout<<getch();
```

```
break;
```

```
}
```

```
case 5:{
```

```
break;
```

```
}
```

```
}
```

```
}
```

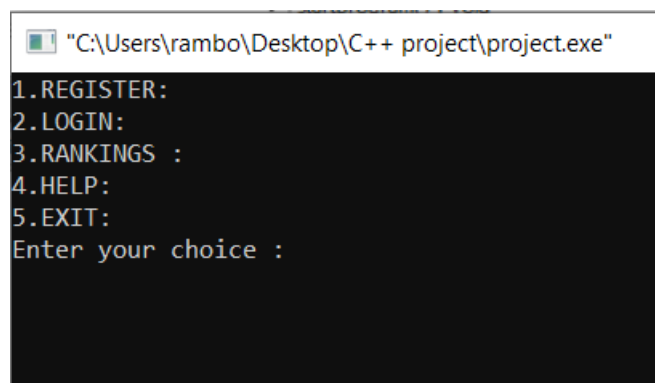
```
while(ch!=5);
```

```
return 0;
```

```
}
```

OUTPUT SCREENS

```
Enter Username:tarush
Enter Password less than 8 characters:*****
Account Successfully Registered
Press any key to continue_
```



```
"C:\Users\rambo\Desktop\C++ project\project.exe"
1.REGISTER:
2.LOGIN:
3.RANKINGS :
4.HELP:
5.EXIT:
Enter your choice :
```

```
Enter your username:tarush
Enter your Password:*****
Login Successful
```

```
Enter your username:tarush
Enter your Password:*****
Login Successful
```

```
Enter your username:tarush
Enter your Password:*****
Login Successful
```

```
ENTER NAME OF THE MAZE YOU WANT TO SOLVE :maze4

Your Maze is
# # # # # # #
S . . # # . . .
# # . # # . # .
# . . # # . . .
# . # # # . # .
# . # # # . # .
# . . . . # D

The fastest solution to the specified maze is:-

S . .
.
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
D
```

```
ENTER NAME OF THE MAZE YOU WANT TO SOLVE :maze4

Your Maze is
# # # # # # #
S . . # # . . .
# # . # # . # .
# . . # # . . .
# . # # # . # .
# . # # # . # .
# . . . . # D

The fastest solution to the specified maze is:-

S . .
.
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
D
```

```
ENTER NAME OF THE MAZE YOU WANT TO SOLVE :maze4

Your Maze is
# # # # # # #
S . . # # . . .
# # . # # . # .
# . . # # . . .
# . # # # . # .
# . # # # . # .
# . . . . # D

The fastest solution to the specified maze is:-

S . .
.
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
D
```

```
ENTER NAME OF THE MAZE YOU WANT TO SOLVE :maze4

Your Maze is
# # # # # # #
S . . # # . . .
# # . # # . # .
# . . # # . . .
# . # # # . # .
# . # # # . # .
# . . . . # D

The fastest solution to the specified maze is:-

S . .
.
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
D
```

```
ENTER NAME OF THE MAZE YOU WANT TO SOLVE :maze4

Your Maze is
# # # # # # #
S . . # # . . .
# # . # # . # .
# . . # # . . .
# . # # # . # .
# . # # # . # .
# . . . . # D

The fastest solution to the specified maze is:-

S . .
.
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
D
```

```
ENTER NAME OF THE MAZE YOU WANT TO SOLVE :maze4

Your Maze is
# # # # # # #
S . . # # . . .
# # . # # . # .
# . . # # . . .
# . # # # . # .
# . # # # . # .
# . . . . # D

The fastest solution to the specified maze is:-

S . .
.
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
D
```





"C:\Users\rambo\Desktop\C++ project\project.exe"










This program is a maze solver which is used to solve any given maze of m x n format. First you have to register your account and a folder of your name will be created. Then you have to create a maze in a .txt format of M X N and specify a source using S and a destination using D. # MEANS BLOCKAGE AND . means Path. Put space between every character. No other character shall be used. Once you login, computer will ask you to enter the name of the maze you want to solve and computer will solve it and create a solution.txt file of that particular maze. A Sample maze has been given in the folder.■

"C:\Users\rambo\Desktop\C++ project\project.exe"

HARDEST MAZES TO SOLVE ACCORDING TO TIME COMPLEXITY

```
1 tarush maze4 996000
2 harsh maze1 0
3 harsh maze2 0
4 harsh maze2 0
5 harsh maze2 0
6 harsh maze4 0
7 harsh maze4 0
```

Name	Date modified	Type	Size
 maze1	11/27/2019 6:52 PM	Text Document	1 KB
 maze2	11/28/2019 10:23 AM	Text Document	1 KB
 maze2sol	11/28/2019 10:36 AM	Text Document	1 KB
 maze4	11/28/2019 10:39 AM	Text Document	1 KB

Name	Date modified	Type	Size
 Samplemaze	11/27/2019 6:04 PM	Text Document	1 KB
 rank	11/28/2019 10:42 AM	Text Document	1 KB
 project.o	11/28/2019 10:43 AM	O File	153 KB
 project	11/28/2019 10:43 AM	Application	1,632 KB
 project	11/27/2019 6:54 PM	CPP File	17 KB
 maze3	11/28/2019 10:29 AM	Text Document	1 KB
 DATA	11/28/2019 10:42 AM	Text Document	1 KB
 tarush	11/28/2019 10:42 AM	File folder	
 harsh	11/28/2019 10:40 AM	File folder	

CONCLUSION

Through this project ,we learnt key concepts used in the industry such as Graphs, STL, Shortest Path Algorithms etc. We were able to implement these concepts and get a thorough idea of their various uses. This project allows us to solve any maze of any size or dimensions and find the most efficient and time-friendly solution there is.

