A picture containing text, clipart

Description automatically generated

RIGA TECHNICAL UNIVERSITY

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

INSTITUTE OF APPLIED COMPUTER SYSTEMS

Practical Assignment 1 for Artificial Intelligence.

Software code’s Google Drive Link: <https://drive.google.com/file/d/1TEbAsYa4skhsZntL30uan4yXKQpuHwHB/view?usp=sharing>

Please put the link in google chrome, download the zip file in this link, and extract it, and then open the GameApp.sln to play the game.

Submitted By,

Harjith Shiva Anbalagan.

201ADB040.

Table of Contents………….

1. Description of the String-Take-it-out-Game which has been developed by me
2. The Programming language chosen, the framework and the GUI application chosen for the game
3. Description of the Algorithm for State Space Graph for the game developed
4. Description of the Algorithm of Minimax for the state space Graph Created
5. Description of the development of the game Logic and Optimal move using of the minimax algorithm for the state space graph created
6. Description of the GUI implementation for the working of game logic developed and embedding it into the GUI developed
7. The Software code in text format for the developed Game
8. References for the algorithms and codes

**1. Description of the String-Take-it-out-Game which has been developed by me:**

The String-Take-it-out-game starts with the string “1112” and it prompts the user whether he/she wishes to the first player/or not and if the player chooses to be the first player then he/she has to take out one number from the current number displayed on the GUI screen and add it to their current score, then the control is passed on to the computer to chooses the most suitable umber displayed on the screen which will make the probability of winning of the computer high and then adds the chosen number to the current score of the computer.

Like this the players makes moves sequentially and the game ends when the string is empty and at last which player has the highest score wins the game and the other one loses the game.

**2.The Programming language chose, the framework and the GUI application chosen for the game:**

I have used the **C#** along with the **.NET** frameworks as it is very easy for me to create an GUI using **C#.NET** with the **Windows Forms App** for the creation of **GUI**.

**3. Description of the Algorithm for State Space Graph for the game developed:**

Basically, I have Create a list of the type node for storing all the nodes of the state space graph created by the end of the algorithm.

I have created a class called node which ha the following encapsulated members which characterizes the parameters for the game:

class node : common //node of a game tree

{

public string id;

public string numbers;

public int p1;

public int p2;

public int depth;

public int minimax\_val;

}

**Id** is the unique identifier for each nodes in the state space graph.

**Numbers** contains the current string which each nodes in the state space graph contains.

**P1** represents the current score of the player 1.

**P2** represents the current score of the player 2.

**Depth** identifies the depth of all the nodes in the state space graph with the root node of depth 1.

**Minimax\_val** represents the minimax value for all the nodes after applying the minimax algorithm on the created game tree.

And also of a list of type node and a dictionary of lists which has all the nodes of the state space graph and children of a particular parent for eaxh node.

public static List<node> nodes = new List<node>();

public static IDictionary<string, List<string>> numberNames = new Dictionary<string, List<string>>();

The implementation for the creation of the state space graph starts with various classes and inheritance of these classes to make the function that we want to call publicly available so that there are no restrictions on the access modifiers and to run the program without any errors.

Initially all the functions are called by assigning the string of the game to the root node of the list and then it will call recursively until the end of the string and finally all the nodes along with the encapsulated members and for the checking purpose I have used the display() function to be sure that I’m on the right track.

The method moves calculate all the possible moves of the game and then store it on the list and dictionary that we have created.

public void moves(string move\_made, List<node> nodes\_generated, node current\_node) // the game moves logic.

{

if (move\_made == "1")

number\_used = "1"; // the number used in the string

else if (move\_made == "2")

number\_used = "2"; // the number used in the string

if (current\_node.numbers.Contains(number\_used))

{

string new\_id = String.Concat("A", j.ToString());

++j;

string new\_string = current\_node.numbers;

int new\_p1 = 0;

int new\_p2 = 0;

if (number\_used == "2")

{

int pos = new\_string.IndexOf(number\_used);

new\_string = new\_string.Remove(pos, 1);

}

else

{

int pos = new\_string.IndexOf(number\_used);

new\_string = new\_string.Remove(pos, 1);

}

if (move\_made == "1" || move\_made == "2")

{

if (current\_node.depth % 2 == 0)

{

new\_p1 = current\_node.p1;

new\_p2 = current\_node.p2 + Int16.Parse(number\_used);

}

else

{

new\_p1 = current\_node.p1 + Int16.Parse(number\_used);

new\_p2 = current\_node.p2;

}

}

int new\_depth = current\_node.depth + 1;

node new\_node = new node();

new\_node.id = new\_id;

new\_node.numbers = new\_string;

new\_node.p1 = new\_p1;

new\_node.p2 = new\_p2;

new\_node.depth = new\_depth;

bool check\_redundant = false;

int i = 0;

while ((!check\_redundant) && (i <= (nodes.Count) - 1)) // while loop for checking whether the node is redundant or not.

{

node check\_node = nodes[i];

if ((check\_node.numbers == new\_node.numbers) && (check\_node.p1 == new\_node.p1) && (check\_node.p2 == new\_node.p2) && (check\_node.depth == new\_node.depth))

{

check\_redundant = true;

--j;

}

else

++i;

}

if (!check\_redundant) // if it is not redundant it is added both as anew node and arcs for it.

{

move new\_move = new move();

new\_move.add\_node(new\_node);

new\_move.add\_arc(current\_node.id, nodes[i].id);

nodes\_generated.Add(new\_node);

}

if (check\_redundant) // if it is redundant it is just added with the existing node's arcs, no new node creation.

{

move new\_move = new move();

node duplicate\_node = nodes[i];

new\_move.add\_arc(current\_node.id, duplicate\_node.id);

}

}

And after the completion of this function I did call a display() function to ensure that whether the algorithm works fine or not, I’m attaching the console output after calling the moves algorithm and before the calculation of the minimax values to the graph.

Graphical user interface, text

Description automatically generated

As one can see above that the redundant nodes has been removes as well as the arcs and one can see that the last member minimax value has all zeros as I have not implemented the minimax algorithm on the nodes that have been created.

**4. Description of the Algorithm of Minimax for the state space Graph Created:**

I have a function calls the assign\_minimax() which will assign the minimax values manually for the nodes at the last level and for the inner level nodes it assigns based on the node’s minimax values and checking whether the node is at the minimum level or at the maximum level.

If the nodes is at the minimum level then it will take minimum values of all it’s descendants and if it is a t the maximum level it takes the maximum value among all it’s descendants.

public void assign\_minimax()

{

//int i = (nodes.Count);

int height = find\_max\_depth();

for (int i = (nodes.Count - 1); i > 0; i--)

{

if (nodes[i].depth == height)

{

if (nodes[i].p1 > nodes[i].p2)

{

nodes[i].minimax\_val = 1;

}

else

{

nodes[i].minimax\_val = -1;

}

}

else

{

foreach (var parent in numberNames.Keys)

{

if (parent == nodes[i].id)

{

int child = numberNames[parent].Count;

if (child == 1)

{

string child\_1\_id = numberNames[parent][0];

for (int k = nodes.Count - 1; k >= 0; k--)

{

if (child\_1\_id == nodes[k].id)

{

nodes[i].minimax\_val = nodes[k].minimax\_val;

}

else

continue;

}

}

else

{

string child\_1\_id = numberNames[parent][0];

string child\_2\_id = numberNames[parent][1];

int child\_1\_heuristic = 0;

int child\_2\_heuristic = 0;

for (int k = nodes.Count - 1; k >= 0; k--)

{

if (child\_1\_id == nodes[k].id)

{

child\_1\_heuristic = nodes[k].minimax\_val;

}

else

continue;

}

for (int k = nodes.Count - 1; k >= 0; k--)

{

if (child\_2\_id == nodes[k].id)

{

child\_2\_heuristic = nodes[k].minimax\_val;

}

else

continue;

}

if (nodes[i].depth % 2 == 0)

{

nodes[i].minimax\_val = Math.Min(child\_1\_heuristic, child\_2\_heuristic);

}

else

{

nodes[i].minimax\_val = Math.Max(child\_1\_heuristic, child\_2\_heuristic);

}

}

}

else

continue;

}

}

}

}

And after he completion of this function the minimax algorithm will be assigned to each and every node of the state space graph, and I’m adding the console output after the minimax algorithm has been applied to all the nodes in the state space graph.

Graphical user interface, text

Description automatically generated

As one can see that all the nodes have the corresponding minimax values after the completion of the algorithm.

So as of now the state space graph along with the minimax values have been created and stored in the lit and the dictionary that we have created before.

**5.Despcrption of the development of the game Logic and Optimal move using of the minimax algorithm for the state space graph created:**

In the Game\_logic.cs I have created two functions named as computer\_play() and the human\_play() both the function takes the current node which is passed to these function and these values are retained as these are passed by the variables as static which retains the last assigned vales.

The game logic first asks whether the player is wishing to start the game or not and if the human decides to play first then he is allowed to choose a number which is displayed as a string by passing the root node of the state space graph, then the chosen number is deleted from the current node’s string and then

The child nodes of the corresponding parents are found using the dictionary and the list available, and the algorithm checks whether if the current node’s values after choosing a value by the human corresponds to which child node by traversing the child nodes list which makes the computation time less as it need not traverse the entire nodes in the state space graph.

Now the matched node’s content is modes as the current node and it’s contents are displayed in the GUI, the next turn is of the computer hence it chooses the optimal node among the child nodes considered and compares their minimax values and chooses the best so that it could lead the human and maps the content of the chosen node’s contents to the GUI.

Likewise, the process alternates until the string is empty and displays who is the winner based on checking their scores.

And if the human doesn’t want to be the first node then the first time control is given to the computer and the same process happens alternating until the string becomes empty and finally the winner is displayed by comparing their scores.

**6.Description of the GUI implementation for the working of game logic developed and embedding it into the GUI developed:**

The GUI has the option to choose who can be the first player whether the human or the computer and for the human to play it displays the number we can choose from the string that is being displayed on the screen and for the computer play it pops a dialog box stating that it is the computer play and the player has to click ok before starting the next move.

For retaining the last assigned values to the node and the current node I have created a HelperVariables class for the variables which needs to be retaining the last assigned value.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace GameApp

{

public class HelperVariables

{

public static int user\_choice = 0;

public static string stringnumber = string.Empty;

public static int humanscore = 0, human\_score = 0;

public static int computerscore = 0,computer\_score =0;

public static int humanplay = 0;

public static int initialstart = 0;

public static string parentnode = String.Empty;

}

}

At the end of the game, it also states that the GAME IS OVER! And it asks whether if we want to start a new game and if we choose yes we start he game from the beginning until the human chooses he don’t want a new game to be started.

The logic for each button clicks, labels, textboxes and dialog boxes are written corresponding to the GUI in the Form1.cs.

Now I will Include the one Iteration of the entire game by choosing not to start the game

Graphical user interface, application

Description automatically generated

If we choose NO then the computer makes the first move

Chart, waterfall chart

Description automatically generated

The computer chooses the number 2 as by choosing it, has the highest possibility of winning and the next move is made by the human.

Graphical user interface, text, application

Description automatically generated

After choosing 1 the player score and the computer score are updated according to the changes made by us and the minimax algorithm, the next move is made by the computer.

Graphical user interface, application

Description automatically generated

The computer chooses 1 as it is the only number left in the string and it is now the chance of the human to choose the number.

Graphical user interface, text, application

Description automatically generated

It says that the GAME IS OVER and the scores of computer and player are displayed on the GUI.

Graphical user interface, application

Description automatically generated

It says that the winner is the computer and it asks whether if we want to start a new game. If we choose YES then the game starts from the beginning else

The GUI is closed and visual studio stops running the program.

Graphical user interface, application

Description automatically generated

**7.The Software code in text format for the developed Game:**

**Form1.cs**

**namespace GameApp**

**{**

**public partial class Form1 : Form**

**{**

**public Form1()**

**{**

**InitializeComponent();**

**}**

**private void Form1\_Load(object sender, EventArgs e)**

**{**

**Game\_logic gmlc = new Game\_logic();**

**label1.Text = gmlc.root\_node.numbers;**

**label9.Visible = false;**

**}**

**private void label1\_Click(object sender, EventArgs e)**

**{**

**}**

**private void button2\_Click(object sender, EventArgs e)**

**{**

**string startstring = label1.Text;**

**if (startstring != string.Empty)**

**{**

**if (!startstring.Contains(button2.Text))**

**{**

**DialogResult result = MessageBox.Show(this,"The chosen number is no longer avilable... Choose another number", "String Game", MessageBoxButtons.OK);**

**if (result == DialogResult.OK)**

**{**

**button1.Focus();**

**return;**

**}**

**}**

**}**

**HelperVariables.humanscore = Convert.ToInt16(label4.Text);**

**HelperVariables.computerscore = Convert.ToInt16(label5.Text);**

**Font LargeFont = new Font("Arial", 25);**

**Game\_logic gmlc = new Game\_logic();**

**if (HelperVariables.humanplay == 1 && label1.Text != String.Empty)**

**{**

**HelperVariables.user\_choice = Convert.ToInt32(button2.Text);**

**gmlc.human\_play(label1.Text, HelperVariables.humanplay, HelperVariables.parentnode);**

**label1.Text = HelperVariables.stringnumber;**

**label4.Text = HelperVariables.humanscore.ToString();**

**}**

**if (HelperVariables.humanplay == 0 && label1.Text != String.Empty)**

**{**

**label8.Visible = false;**

**button1.Visible = false;**

**button2.Visible = false;**

**DialogResult result = MessageBox.Show(this,"Note your score in the score window and now Computer is playing... Click OK", "String Game", MessageBoxButtons.OK);**

**if (result == DialogResult.OK)**

**{**

**HelperVariables.humanscore = Convert.ToInt16(label4.Text);**

**gmlc.computer\_play(label1.Text, HelperVariables.humanplay, HelperVariables.parentnode);**

**label1.Text = HelperVariables.stringnumber;**

**label5.Text = HelperVariables.computerscore.ToString();**

**}**

**if (HelperVariables.humanplay == 1 && label1.Text != String.Empty)**

**{**

**DialogResult result1 = MessageBox.Show(this,"Note Scores in the score window and now it is your turn... Choose the Number", "String Game", MessageBoxButtons.OK);**

**if (result == DialogResult.OK)**

**{**

**label8.Visible = true;**

**button1.Visible = true;**

**button2.Visible = true;**

**}**

**}**

**}**

**if (label1.Text.Equals(string.Empty))**

**{**

**label8.Visible = false;**

**button1.Visible = false;**

**button2.Visible = false;**

**DialogResult result1 = MessageBox.Show(this,"Game Over ... ", "String Game", MessageBoxButtons.OK);**

**if (result1 == DialogResult.OK)**

**{**

**if (HelperVariables.humanscore > HelperVariables.computerscore)**

**label9.Text = "WINNER IS " + "PLAYER";**

**else if (HelperVariables.humanscore == HelperVariables.computerscore)**

**label9.Text = "IT'S A DRAW...";**

**else if (HelperVariables.humanscore < HelperVariables.computerscore)**

**label9.Text = "WINNER IS " + "COMPUTER";**

**}**

**label9.Visible = true;**

**label9.Font = LargeFont;**

**label1.Visible = false;**

**label7.Visible = false;**

**DialogResult result11 = MessageBox.Show(this,"Do you want to Play Again?", "String Game", MessageBoxButtons.YesNo);**

**Form1 frm1 = new Form1();**

**if (result11 == DialogResult.Yes)**

**{**

**this.Hide();**

**frm1.Show();**

**HelperVariables.user\_choice = 0;**

**HelperVariables.stringnumber = string.Empty;**

**HelperVariables.humanscore = 0; HelperVariables.human\_score = 0;**

**HelperVariables.computerscore = 0; HelperVariables.computer\_score = 0;**

**HelperVariables.humanplay = 0;**

**HelperVariables.initialstart = 0;**

**HelperVariables.parentnode = String.Empty;**

**}**

**else**

**{**

**MessageBox.Show("Thank You... Catch You Another Time...Bye", "String Game");**

**frm1.Close();**

**this.Close();**

**}**

**}**

**}**

**private void button1\_Click(object sender, EventArgs e)**

**{**

**string startstring = label1.Text;**

**if (startstring != string.Empty)**

**{**

**if (!startstring.Contains(button1.Text))**

**{**

**DialogResult result = MessageBox.Show("The chosen number is no longer available... Choose another number", "String Game", MessageBoxButtons.OK);**

**if (result == DialogResult.OK)**

**{**

**button2.Focus();**

**return;**

**}**

**}**

**}**

**HelperVariables.humanscore = Convert.ToInt16(label4.Text);**

**Font LargeFont = new Font("Arial", 25);**

**Game\_logic gmlc = new Game\_logic();**

**if (HelperVariables.humanplay == 1 && label1.Text != String.Empty)**

**{**

**HelperVariables.user\_choice = Convert.ToInt32(button1.Text);**

**gmlc.human\_play(label1.Text, HelperVariables.humanplay, HelperVariables.parentnode);**

**label1.Text = HelperVariables.stringnumber;**

**label4.Text = HelperVariables.humanscore.ToString();**

**}**

**if (HelperVariables.humanplay == 0 && label1.Text != String.Empty)**

**{**

**label8.Visible = false;**

**button1.Visible = false;**

**button2.Visible = false;**

**DialogResult result = MessageBox.Show("Note your score in the score window and now Computer is playing... Click OK", "String Game", MessageBoxButtons.OK);**

**if (result == DialogResult.OK)**

**{**

**HelperVariables.computerscore = Convert.ToInt16(label5.Text);**

**gmlc.computer\_play(label1.Text, HelperVariables.humanplay, HelperVariables.parentnode);**

**label1.Text = HelperVariables.stringnumber;**

**label5.Text = HelperVariables.computerscore.ToString();**

**}**

**if (HelperVariables.humanplay == 1 && label1.Text != String.Empty)**

**{**

**DialogResult result1 = MessageBox.Show("Note Scores in the score window and now it is your turn... Choose the Number", "String Game", MessageBoxButtons.OK);**

**if (result == DialogResult.OK)**

**{**

**label8.Visible = true;**

**button1.Visible = true;**

**button2.Visible = true;**

**}**

**}**

**}**

**if (label1.Text.Equals(string.Empty))**

**{**

**label8.Visible = false;**

**button1.Visible = false;**

**button2.Visible = false;**

**DialogResult result1 = MessageBox.Show("Game Over ... ", "String Game", MessageBoxButtons.OK);**

**if (result1 == DialogResult.OK)**

**{**

**if (HelperVariables.humanscore > HelperVariables.computerscore)**

**label9.Text = "WINNER IS " + "PLAYER";**

**else if (HelperVariables.humanscore == HelperVariables.computerscore)**

**label9.Text = "IT'S A DRAW...";**

**else if (HelperVariables.humanscore < HelperVariables.computerscore)**

**label9.Text = "WINNER IS " + "COMPUTER";**

**}**

**label9.Visible = true;**

**label9.Font = LargeFont;**

**label1.Visible = false;**

**label7.Visible = false;**

**}**

**}**

**private void button3\_Click(object sender, EventArgs e)**

**{**

**HelperVariables.parentnode = "A1";**

**HelperVariables.initialstart = 1;**

**label8.Visible = true;**

**button1.Visible = true;**

**button2.Visible = true;**

**HelperVariables.humanplay = 1;**

**button3.Visible = false;**

**button4.Visible = false;**

**label6.Visible = false;**

**}**

**private void button4\_Click(object sender, EventArgs e)**

**{**

**HelperVariables.parentnode = "A1";**

**HelperVariables.initialstart = 0;**

**label8.Visible = true;**

**button1.Visible = true;**

**button2.Visible = true;**

**HelperVariables.humanplay = 0;**

**button3.Visible = false;**

**button4.Visible = false;**

**label6.Visible = false;**

**HelperVariables.humanscore = Convert.ToInt16(label4.Text);**

**Game\_logic gmlc = new Game\_logic();**

**if (HelperVariables.humanplay == 0 && label1.Text != String.Empty)**

**{**

**DialogResult result = MessageBox.Show("Computer starts playing... Click OK", "String Game", MessageBoxButtons.OK);**

**if (result == DialogResult.OK)**

**{**

**HelperVariables.computerscore = Convert.ToInt16(label5.Text);**

**gmlc.computer\_play(label1.Text, HelperVariables.humanplay, HelperVariables.parentnode);**

**label1.Text = HelperVariables.stringnumber;**

**label5.Text = HelperVariables.computerscore.ToString();**

**}**

**}**

**}**

**}**

**}**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Program.cs:**

**namespace GameApp**

**{**

**class common //the global members used through-out.**

**{**

**public static List<node> nodes\_generated = new List<node>();**

**public static int j = 2;**

**public static string number\_used;**

**public static node current\_node;**

**public static List<node> nodes = new List<node>();**

**public static IDictionary<string, List<string>> numberNames = new Dictionary<string, List<string>>();**

**public static List<node> heuristic\_nodes = new List<node>();**

**public static List<string> level\_nodes = new List<string>();**

**}**

**class node : common //node of a game tree**

**{**

**public string id;**

**public string numbers;**

**public int p1;**

**public int p2;**

**public int depth;**

**public int minimax\_val;**

**}**

**class string\_tree : node // adding nodes and arcs function implementation.**

**{**

**/\*public string\_tree()**

**{**

**List<node> nodes = new List<node>();**

**IDictionary<string, List<string>> numberNames = new Dictionary<string, List<string>>();**

**}\*/**

**public void add\_node(node n)**

**{**

**nodes.Add(n);**

**}**

**public void add\_arc(string start\_node\_id, string end\_node\_id)**

**{**

**if (!numberNames.ContainsKey(start\_node\_id))**

**{**

**numberNames.Add(start\_node\_id, new List<string>());**

**}**

**numberNames[start\_node\_id].Add(end\_node\_id);**

**}**

**}**

**class move : string\_tree**

**{**

**public void moves(string move\_made, List<node> nodes\_generated, node current\_node) // the game moves logic.**

**{**

**if (move\_made == "1")**

**number\_used = "1"; // the number used in the string**

**else if (move\_made == "2")**

**number\_used = "2"; // the number used in the string**

**if (current\_node.numbers.Contains(number\_used))**

**{**

**string new\_id = String.Concat("A", j.ToString());**

**++j;**

**string new\_string = current\_node.numbers;**

**int new\_p1 = 0;**

**int new\_p2 = 0;**

**if (number\_used == "2")**

**{**

**int pos = new\_string.IndexOf(number\_used);**

**new\_string = new\_string.Remove(pos, 1);**

**}**

**else**

**{**

**int pos = new\_string.IndexOf(number\_used);**

**new\_string = new\_string.Remove(pos, 1);**

**}**

**if (move\_made == "1" || move\_made == "2")**

**{**

**if (current\_node.depth % 2 == 0)**

**{**

**new\_p1 = current\_node.p1;**

**new\_p2 = current\_node.p2 + Int16.Parse(number\_used);**

**}**

**else**

**{**

**new\_p1 = current\_node.p1 + Int16.Parse(number\_used);**

**new\_p2 = current\_node.p2;**

**}**

**}**

**int new\_depth = current\_node.depth + 1;**

**node new\_node = new node();**

**new\_node.id = new\_id;**

**new\_node.numbers = new\_string;**

**new\_node.p1 = new\_p1;**

**new\_node.p2 = new\_p2;**

**new\_node.depth = new\_depth;**

**bool check\_redundant = false;**

**int i = 0;**

**while ((!check\_redundant) && (i <= (nodes.Count) - 1)) // while loop for checking whether the node is redundant or not.**

**{**

**node check\_node = nodes[i];**

**if ((check\_node.numbers == new\_node.numbers) && (check\_node.p1 == new\_node.p1) && (check\_node.p2 == new\_node.p2) && (check\_node.depth == new\_node.depth))**

**{**

**check\_redundant = true;**

**--j;**

**}**

**else**

**++i;**

**}**

**if (!check\_redundant) // if it is not redundant it is added both as anew node and arcs for it.**

**{**

**move new\_move = new move();**

**new\_move.add\_node(new\_node);**

**new\_move.add\_arc(current\_node.id, nodes[i].id);**

**nodes\_generated.Add(new\_node);**

**}**

**if (check\_redundant) // if it is redundant it is just added with the existing node's arcs, no new node creation.**

**{**

**move new\_move = new move();**

**node duplicate\_node = nodes[i];**

**new\_move.add\_arc(current\_node.id, duplicate\_node.id);**

**}**

**}**

**}**

**public void generate\_nodes() // making the moves for a particular node.**

**{**

**while (nodes\_generated.Count > 0)**

**{**

**current\_node = nodes\_generated[0];**

**move m1 = new move();**

**m1.moves("1", nodes\_generated, current\_node);**

**m1.moves("2", nodes\_generated, current\_node);**

**nodes\_generated.RemoveAt(0);**

**}**

**}**

**public void display()**

**{**

**for (int i = 0; i < nodes.Count; i++)**

**{**

**Console.WriteLine("node id:" + nodes[i].id + "--------" + "present string: " + nodes[i].numbers**

**+ "--------" + "player1(score): " + nodes[i].p1 + "-------" + "player2(score): " + nodes[i].p2 + "---------minimax\_value: " + nodes[i].minimax\_val);**

**}**

**foreach (var parent in numberNames.Keys)**

**{**

**var child = numberNames[parent].Count;**

**if (child == 1)**

**Console.WriteLine(parent + " only child: " + numberNames[parent][0]);**

**if (child == 2)**

**{**

**Console.WriteLine(parent + " child 1: " + numberNames[parent][0]);**

**Console.WriteLine(parent + " child 2: " + numberNames[parent][1]);**

**}**

**}**

**}**

**}**

**class tree\_values : move**

**{**

**public int find\_max\_depth()**

**{**

**int max\_depth = nodes[0].depth;**

**for (int i = 1; i <= (nodes.Count) - 1; i++)**

**{**

**if (nodes[i].depth > max\_depth)**

**max\_depth = nodes[i].depth;**

**}**

**return max\_depth;**

**}**

**public void assign\_minimax()**

**{**

**//int i = (nodes.Count);**

**int height = find\_max\_depth();**

**for (int i = (nodes.Count - 1); i > 0; i--)**

**{**

**if (nodes[i].depth == height)**

**{**

**if (nodes[i].p1 > nodes[i].p2)**

**{**

**nodes[i].minimax\_val = 1;**

**}**

**else**

**{**

**nodes[i].minimax\_val = -1;**

**}**

**}**

**else**

**{**

**foreach (var parent in numberNames.Keys)**

**{**

**if (parent == nodes[i].id)**

**{**

**int child = numberNames[parent].Count;**

**if (child == 1)**

**{**

**string child\_1\_id = numberNames[parent][0];**

**for (int k = nodes.Count - 1; k >= 0; k--)**

**{**

**if (child\_1\_id == nodes[k].id)**

**{**

**nodes[i].minimax\_val = nodes[k].minimax\_val;**

**}**

**else**

**continue;**

**}**

**}**

**else**

**{**

**string child\_1\_id = numberNames[parent][0];**

**string child\_2\_id = numberNames[parent][1];**

**int child\_1\_heuristic = 0;**

**int child\_2\_heuristic = 0;**

**for (int k = nodes.Count - 1; k >= 0; k--)**

**{**

**if (child\_1\_id == nodes[k].id)**

**{**

**child\_1\_heuristic = nodes[k].minimax\_val;**

**}**

**else**

**continue;**

**}**

**for (int k = nodes.Count - 1; k >= 0; k--)**

**{**

**if (child\_2\_id == nodes[k].id)**

**{**

**child\_2\_heuristic = nodes[k].minimax\_val;**

**}**

**else**

**continue;**

**}**

**if (nodes[i].depth % 2 == 0)**

**{**

**nodes[i].minimax\_val = Math.Min(child\_1\_heuristic, child\_2\_heuristic);**

**}**

**else**

**{**

**nodes[i].minimax\_val = Math.Max(child\_1\_heuristic, child\_2\_heuristic);**

**}**

**}**

**}**

**else**

**continue;**

**}**

**}**

**}**

**}**

**internal static class Program**

**{**

**/// <summary>**

**/// The main entry point for the application.**

**/// </summary>**

**[STAThread]**

**static void Main()**

**{**

**// To customize application configuration such as set high DPI settings or default font,**

**// see https://aka.ms/applicationconfiguration.**

**ApplicationConfiguration.Initialize();**

**node first\_node = new node();**

**string\_tree root = new string\_tree();**

**move move\_gen = new move();**

**tree\_values T = new tree\_values();**

**first\_node.id = "A1";**

**first\_node.numbers = "1112";**

**number\_used = first\_node.numbers;**

**first\_node.p1 = 0;**

**first\_node.p2 = 0;**

**first\_node.depth = 1;**

**nodes\_generated.Add(first\_node); // adding the first initial input of string and player's scores.**

**root.add\_node(first\_node);**

**move\_gen.generate\_nodes(); // the function to create children for the current node selected.**

**T.assign\_minimax();**

**// move\_gen.display();**

**Application.Run(new Form1());**

**}**

**}**

**}**

**}**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Game\_logic.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace GameApp

{

class Game\_logic : HelperVariables

{

public node root\_node = common.nodes[0];

public void human\_play(string rootnode, int humanind,string parent\_node)

{

string rn = rootnode,referencenode;

int childcount = 0;

string Childnode;

string NodeId = String.Empty, stnumber1 = String.Empty, stnumber2 = String.Empty;

if (humanind == 1)

{

int n = rn.IndexOf(user\_choice.ToString());

rn = rn.Remove(n, 1);

stringnumber = rn;

humanscore += user\_choice;

HelperVariables.human\_score += user\_choice;

for (int k = 0; k <= common.nodes.Count - 1; k++)

{

if (stringnumber.Equals(common.nodes[k].numbers))

{

if(initialstart == 1)

{

if (HelperVariables.human\_score.Equals(common.nodes[k].p1))

{

HelperVariables.parentnode = common.nodes[k].id;

stringnumber = common.nodes[k].numbers;

}

}

else

{

if (HelperVariables.human\_score.Equals(common.nodes[k].p2))

{

parentnode = common.nodes[k].id;

stringnumber = common.nodes[k].numbers;

}

}

}

}

}

HelperVariables.humanplay = 0;

}

public void computer\_play(string rootnode, int humanind,string parent\_node)

{

string rn = rootnode;

int childcount = 0;

string Childnode;

string NodeId = String.Empty, stnumber1 = String.Empty, stnumber2 = String.Empty;

int Minmaxval1 = 0, Minmaxval2 = 0, score1 =0,score2 =0;

if (humanind == 0)

{

stringnumber = rn;

foreach (var parent in common.numberNames.Keys)

{

if (parent.Equals(parent\_node))

{

childcount = common.numberNames[parent].Count;

if (childcount == 1)

{

Childnode = common.numberNames[parent][0];

for (int k = 0; k <= common.nodes.Count - 1; k++)

{

if (Childnode.Equals(common.nodes[k].id))

{

HelperVariables.parentnode = common.nodes[k].id;

if (initialstart == 0)

{

stringnumber = common.nodes[k].numbers;

computerscore = common.nodes[k].p1;

break;

}

else

{

stringnumber = common.nodes[k].numbers;

computerscore = common.nodes[k].p2;

break;

}

}

}

}

if (childcount == 2)

{

var child1 = common.numberNames[parent][0];

var child2 = common.numberNames[parent][1];

for (int k = common.nodes.Count - 1; k >= 0; k--)

{

if (common.nodes[k].id == child1)

{

Minmaxval1 = common.nodes[k].minimax\_val;

stnumber1 = common.nodes[k].numbers;

HelperVariables.parentnode = common.nodes[k].id;

if (initialstart == 0)

{

score1 = common.nodes[k].p1;

}

else

{

score1 = common.nodes[k].p2;

}

}

}

for (int k = common.nodes.Count - 1; k >= 0; k--)

{

if (common.nodes[k].id == child2)

{

Minmaxval2 = common.nodes[k].minimax\_val;

stnumber2 = common.nodes[k].numbers;

HelperVariables.parentnode = common.nodes[k].id;

if (initialstart == 0)

{

score2 = common.nodes[k].p1;

}

else

{

score2 = common.nodes[k].p2;

}

}

}

if (initialstart == 0)

{

if (Minmaxval1 > Minmaxval2)

{

stringnumber = stnumber1;

computerscore = score1;

}

else

{

stringnumber = stnumber2;

computerscore = score2;

}

}

else if (initialstart == 1)

{

if (Minmaxval1 < Minmaxval2)

{

stringnumber = stnumber1;

computerscore = score1;

}

else

{

stringnumber = stnumber2;

computerscore = score2;

}

}

}

break;

}

}

}

HelperVariables.humanplay = 1;

}

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**HelperVariables.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace GameApp

{

public class HelperVariables

{

public static int user\_choice = 0;

public static string stringnumber = string.Empty;

public static int humanscore = 0, human\_score = 0;

public static int computerscore = 0,computer\_score =0;

public static int humanplay = 0;

public static int initialstart = 0;

public static string parentnode = String.Empty;

}

}

**Form1.Designer.cs:**

**namespace GameApp**

**{**

**partial class Form1**

**{**

**/// <summary>**

**/// Required designer variable.**

**/// </summary>**

**private System.ComponentModel.IContainer components = null;**

**/// <summary>**

**/// Clean up any resources being used.**

**/// </summary>**

**/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>**

**protected override void Dispose(bool disposing)**

**{**

**if (disposing && (components != null))**

**{**

**components.Dispose();**

**}**

**base.Dispose(disposing);**

**}**

**#region Windows Form Designer generated code**

**/// <summary>**

**/// Required method for Designer support - do not modify**

**/// the contents of this method with the code editor.**

**/// </summary>**

**private void InitializeComponent()**

**{**

**this.label1 = new System.Windows.Forms.Label();**

**this.button1 = new System.Windows.Forms.Button();**

**this.button2 = new System.Windows.Forms.Button();**

**this.label2 = new System.Windows.Forms.Label();**

**this.label3 = new System.Windows.Forms.Label();**

**this.label4 = new System.Windows.Forms.Label();**

**this.label5 = new System.Windows.Forms.Label();**

**this.label6 = new System.Windows.Forms.Label();**

**this.label7 = new System.Windows.Forms.Label();**

**this.label8 = new System.Windows.Forms.Label();**

**this.button3 = new System.Windows.Forms.Button();**

**this.button4 = new System.Windows.Forms.Button();**

**this.label9 = new System.Windows.Forms.Label();**

**this.label10 = new System.Windows.Forms.Label();**

**this.label11 = new System.Windows.Forms.Label();**

**this.label12 = new System.Windows.Forms.Label();**

**this.SuspendLayout();**

**//**

**// label1**

**//**

**this.label1.AutoSize = true;**

**this.label1.Font = new System.Drawing.Font("Roboto", 14F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);**

**this.label1.Location = new System.Drawing.Point(354, 66);**

**this.label1.Name = "label1";**

**this.label1.Size = new System.Drawing.Size(68, 23);**

**this.label1.TabIndex = 0;**

**this.label1.Text = "label1";**

**this.label1.Click += new System.EventHandler(this.label1\_Click);**

**//**

**// button1**

**//**

**this.button1.Font = new System.Drawing.Font("Segoe UI", 18F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);**

**this.button1.Location = new System.Drawing.Point(371, 161);**

**this.button1.Name = "button1";**

**this.button1.Size = new System.Drawing.Size(84, 50);**

**this.button1.TabIndex = 1;**

**this.button1.Text = "2";**

**this.button1.UseVisualStyleBackColor = true;**

**this.button1.Visible = false;**

**this.button1.Click += new System.EventHandler(this.button1\_Click);**

**//**

**// button2**

**//**

**this.button2.Font = new System.Drawing.Font("Segoe UI", 18F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);**

**this.button2.Location = new System.Drawing.Point(276, 161);**

**this.button2.Name = "button2";**

**this.button2.Size = new System.Drawing.Size(78, 50);**

**this.button2.TabIndex = 2;**

**this.button2.Text = "1";**

**this.button2.UseVisualStyleBackColor = true;**

**this.button2.Visible = false;**

**this.button2.Click += new System.EventHandler(this.button2\_Click);**

**//**

**// label2**

**//**

**this.label2.AutoSize = true;**

**this.label2.Font = new System.Drawing.Font("Roboto", 14F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);**

**this.label2.Location = new System.Drawing.Point(39, 289);**

**this.label2.Name = "label2";**

**this.label2.Size = new System.Drawing.Size(133, 23);**

**this.label2.TabIndex = 3;**

**this.label2.Text = "Player Score:";**

**//**

**// label3**

**//**

**this.label3.AutoSize = true;**

**this.label3.Font = new System.Drawing.Font("Roboto", 14F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);**

**this.label3.Location = new System.Drawing.Point(6, 326);**

**this.label3.Name = "label3";**

**this.label3.Size = new System.Drawing.Size(166, 23);**

**this.label3.TabIndex = 4;**

**this.label3.Text = "Computer Score:";**

**//**

**// label4**

**//**

**this.label4.AutoSize = true;**

**this.label4.Font = new System.Drawing.Font("Roboto", 14F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);**

**this.label4.Location = new System.Drawing.Point(178, 289);**

**this.label4.Name = "label4";**

**this.label4.Size = new System.Drawing.Size(22, 23);**

**this.label4.TabIndex = 5;**

**this.label4.Text = "0";**

**//**

**// label5**

**//**

**this.label5.AutoSize = true;**

**this.label5.Font = new System.Drawing.Font("Roboto", 14F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);**

**this.label5.Location = new System.Drawing.Point(178, 326);**

**this.label5.Name = "label5";**

**this.label5.Size = new System.Drawing.Size(22, 23);**

**this.label5.TabIndex = 6;**

**this.label5.Text = "0";**

**//**

**// label6**

**//**

**this.label6.AutoSize = true;**

**this.label6.Font = new System.Drawing.Font("Roboto", 14F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);**

**this.label6.Location = new System.Drawing.Point(196, 123);**

**this.label6.Name = "label6";**

**this.label6.Size = new System.Drawing.Size(301, 23);**

**this.label6.TabIndex = 7;**

**this.label6.Text = "Do you want to start the game?";**

**//**

**// label7**

**//**

**this.label7.AutoSize = true;**

**this.label7.Font = new System.Drawing.Font("Roboto", 14F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);**

**this.label7.Location = new System.Drawing.Point(227, 66);**

**this.label7.Name = "label7";**

**this.label7.Size = new System.Drawing.Size(121, 23);**

**this.label7.TabIndex = 8;**

**this.label7.Text = "Start String:";**

**//**

**// label8**

**//**

**this.label8.AutoSize = true;**

**this.label8.Font = new System.Drawing.Font("Segoe UI", 14F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);**

**this.label8.Location = new System.Drawing.Point(177, 176);**

**this.label8.Name = "label8";**

**this.label8.Size = new System.Drawing.Size(93, 25);**

**this.label8.TabIndex = 9;**

**this.label8.Text = "CHOOSE:";**

**this.label8.Visible = false;**

**//**

**// button3**

**//**

**this.button3.Font = new System.Drawing.Font("Segoe UI", 18F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);**

**this.button3.Location = new System.Drawing.Point(199, 154);**

**this.button3.Name = "button3";**

**this.button3.Size = new System.Drawing.Size(130, 47);**

**this.button3.TabIndex = 10;**

**this.button3.Text = "YES";**

**this.button3.UseVisualStyleBackColor = true;**

**this.button3.Click += new System.EventHandler(this.button3\_Click);**

**//**

**// button4**

**//**

**this.button4.Font = new System.Drawing.Font("Segoe UI", 18F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);**

**this.button4.Location = new System.Drawing.Point(354, 154);**

**this.button4.Name = "button4";**

**this.button4.Size = new System.Drawing.Size(137, 47);**

**this.button4.TabIndex = 11;**

**this.button4.Text = "NO";**

**this.button4.UseVisualStyleBackColor = true;**

**this.button4.Click += new System.EventHandler(this.button4\_Click);**

**//**

**// label9**

**//**

**this.label9.AutoSize = true;**

**this.label9.Location = new System.Drawing.Point(39, 123);**

**this.label9.Name = "label9";**

**this.label9.Size = new System.Drawing.Size(38, 15);**

**this.label9.TabIndex = 12;**

**this.label9.Text = "label9";**

**//**

**// label10**

**//**

**this.label10.AutoSize = true;**

**this.label10.Font = new System.Drawing.Font("Segoe UI", 14F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);**

**this.label10.ForeColor = System.Drawing.SystemColors.Highlight;**

**this.label10.Location = new System.Drawing.Point(275, 15);**

**this.label10.Name = "label10";**

**this.label10.Size = new System.Drawing.Size(141, 25);**

**this.label10.TabIndex = 13;**

**this.label10.Text = "STRING GAME";**

**//**

**// label11**

**//**

**this.label11.AutoSize = true;**

**this.label11.Font = new System.Drawing.Font("Calibri", 12F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point);**

**this.label11.ForeColor = System.Drawing.SystemColors.HighlightText;**

**this.label11.Location = new System.Drawing.Point(529, 0);**

**this.label11.Name = "label11";**

**this.label11.Size = new System.Drawing.Size(272, 57);**

**this.label11.TabIndex = 14;**

**this.label11.Text = "Developed By: Harith Shiva Anbalagan\r\n 201ADB040\r\n ";**

**//**

**// label12**

**//**

**this.label12.AutoSize = true;**

**this.label12.Font = new System.Drawing.Font("Segoe UI", 12F, ((System.Drawing.FontStyle)(((System.Drawing.FontStyle.Bold | System.Drawing.FontStyle.Italic)**

**| System.Drawing.FontStyle.Underline))), System.Drawing.GraphicsUnit.Point);**

**this.label12.Location = new System.Drawing.Point(6, 265);**

**this.label12.Name = "label12";**

**this.label12.Size = new System.Drawing.Size(119, 21);**

**this.label12.TabIndex = 15;**

**this.label12.Text = "Score Window:";**

**//**

**// Form1**

**//**

**this.AutoScaleDimensions = new System.Drawing.SizeF(7F, 15F);**

**this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;**

**this.BackColor = System.Drawing.SystemColors.ControlDark;**

**this.ClientSize = new System.Drawing.Size(800, 450);**

**this.Controls.Add(this.label12);**

**this.Controls.Add(this.label11);**

**this.Controls.Add(this.label10);**

**this.Controls.Add(this.label9);**

**this.Controls.Add(this.button4);**

**this.Controls.Add(this.button3);**

**this.Controls.Add(this.label8);**

**this.Controls.Add(this.label7);**

**this.Controls.Add(this.label6);**

**this.Controls.Add(this.label5);**

**this.Controls.Add(this.label4);**

**this.Controls.Add(this.label3);**

**this.Controls.Add(this.label2);**

**this.Controls.Add(this.button2);**

**this.Controls.Add(this.button1);**

**this.Controls.Add(this.label1);**

**this.Name = "Form1";**

**this.Text = "GAMEAPP";**

**this.Load += new System.EventHandler(this.Form1\_Load);**

**this.ResumeLayout(false);**

**this.PerformLayout();**

**}**

**#endregion**

**private Label label1;**

**private Button button1;**

**private Button button2;**

**private Label label2;**

**private Label label3;**

**private Label label4;**

**private Label label5;**

**private Label label6;**

**private Label label7;**

**private Label label8;**

**private Button button3;**

**private Button button4;**

**private Label label9;**

**private Label label10;**

**private Label label11;**

**private Label label12;**

**}**

**}**

**8.References:**

[**https://www.programiz.com/csharp-programming**](https://www.programiz.com/csharp-programming)

[**https://www.javatpoint.com/mini-max-algorithm-in-ai**](https://www.javatpoint.com/mini-max-algorithm-in-ai)

[**https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-1-introduction/**](https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-1-introduction/)

[**https://github.com/in-op/GameAI**](https://github.com/in-op/GameAI)

[**https://www.geeksforgeeks.org/search-algorithms-in-ai/**](https://www.geeksforgeeks.org/search-algorithms-in-ai/)

[**https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/constructors**](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/constructors)

[**https://estudijas.rtu.lv/mod/resource/view.php?id=1967141**](https://estudijas.rtu.lv/mod/resource/view.php?id=1967141)