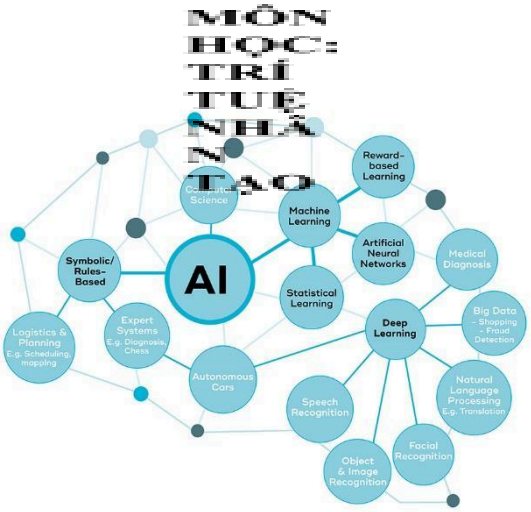
BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC NHA TRANG
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO BÀI TẬP MÔN HỌC: TRÍ TUỆ NHÂN TẠO



Giảng viên bộ môn : **Nguyễn Đình Cường**
Sinh viên thực hiện : **Nguyễn Văn Hải Long**
Lớp : **60.CNTT-1**
MSSV : **60136035**

Tháng 8 năm 2021

PHỤ LỤC

BÁO CÁO BÀI TẬP

1. Programming Heuristic

1.1. Thuật toán tìm kiếm A*(A star)

A* là một thuật toán tìm kiếm trong đồ thị. Thuật toán sẽ tìm đường từ 1 nút ban đầu đến 1 nút đích cho trước sao cho chi phí là tốt nhất (thấp nhất) và số bước duyệt là ít nhất (mang lại sự tối ưu cho bài toán).

A* là thuật toán cải thiện hiệu năng từ thuật toán Greedy Best-First Search. Khi Greedy Best-First Search chỉ tìm đường dựa trên hàm ước lượng từ điểm kế tiếp đến đích mà không tính độ dài quãng đường mà nó đã đi vì có thể ước lượng là nhỏ nhưng đường đã đi lạimang giá trị quá lớn.

Ý tưởng : Tránh việc xét các nhánh tìm kiếm đã xác định cho đến thời điểm hiện tại là có chi phí cao.

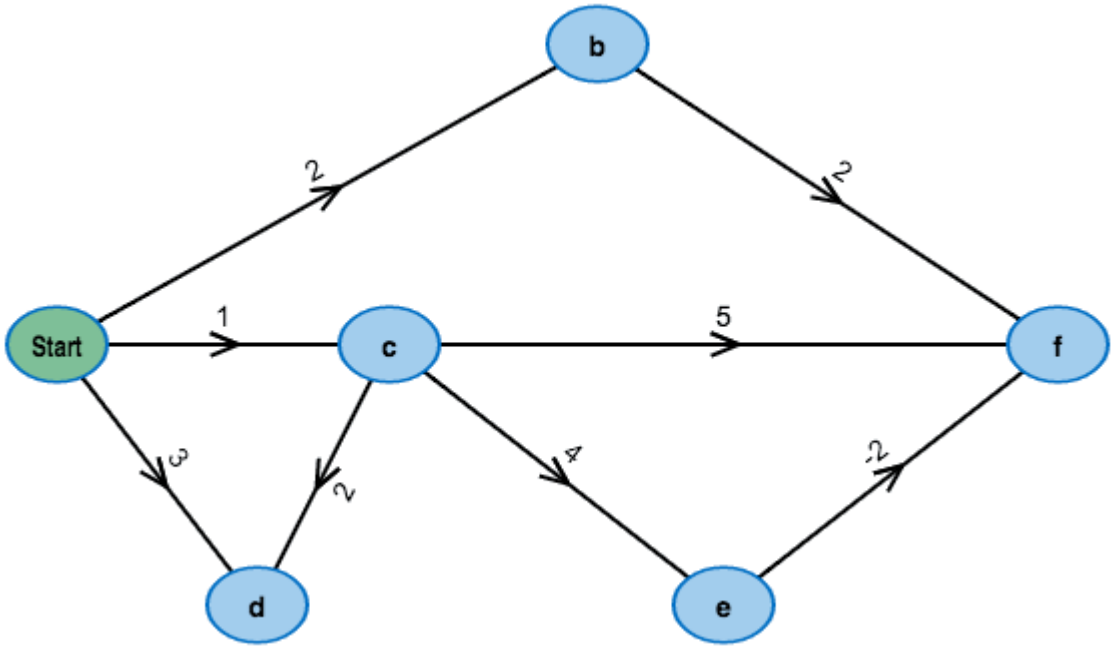
Sử dụng hàm đánh giá f(x) = h(x) + g(x).

h(x) : là ước lượng từ nút hiện tại tới nút đích

g(x) : chi phí từ nút gốc tới nút hiện tại n

f(x) : là chi phí tổng thể ước lượng của đường đi qua nút hiện tại n đến đích.

Thuật toán A* sử dụng f(x) làm tiêu chí chọn đỉnh tiếp theo trong đồ thị.



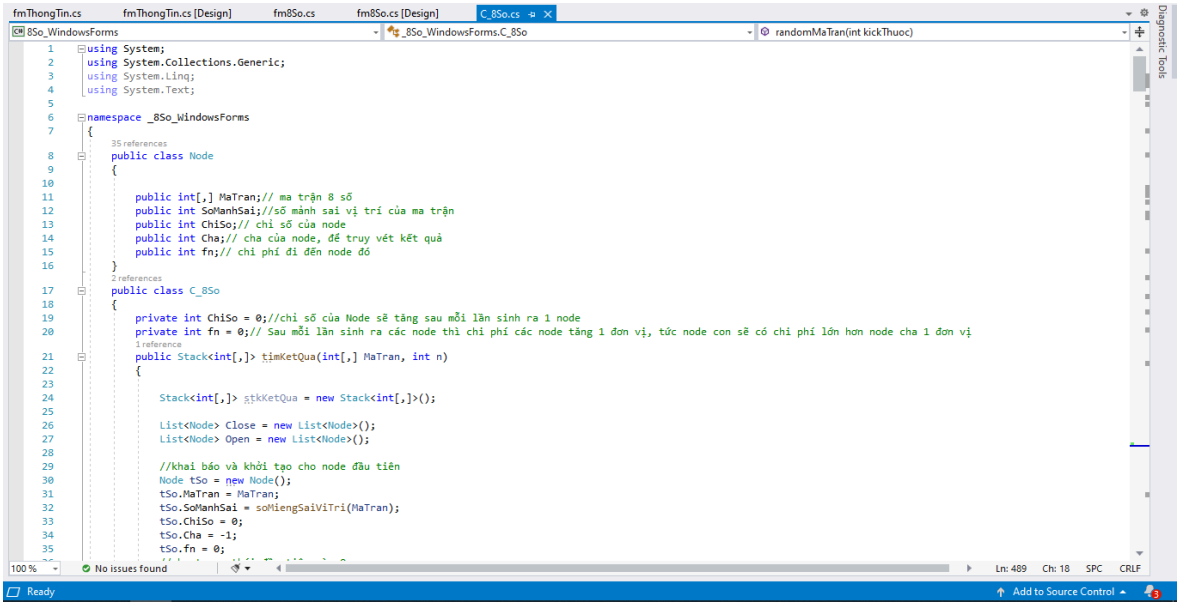
1.2. Chương trình demo: Puzzle 8 sử dụng thuật toán A*

Ngôn ngữ sử dụng: C#

Giao diện: Winform

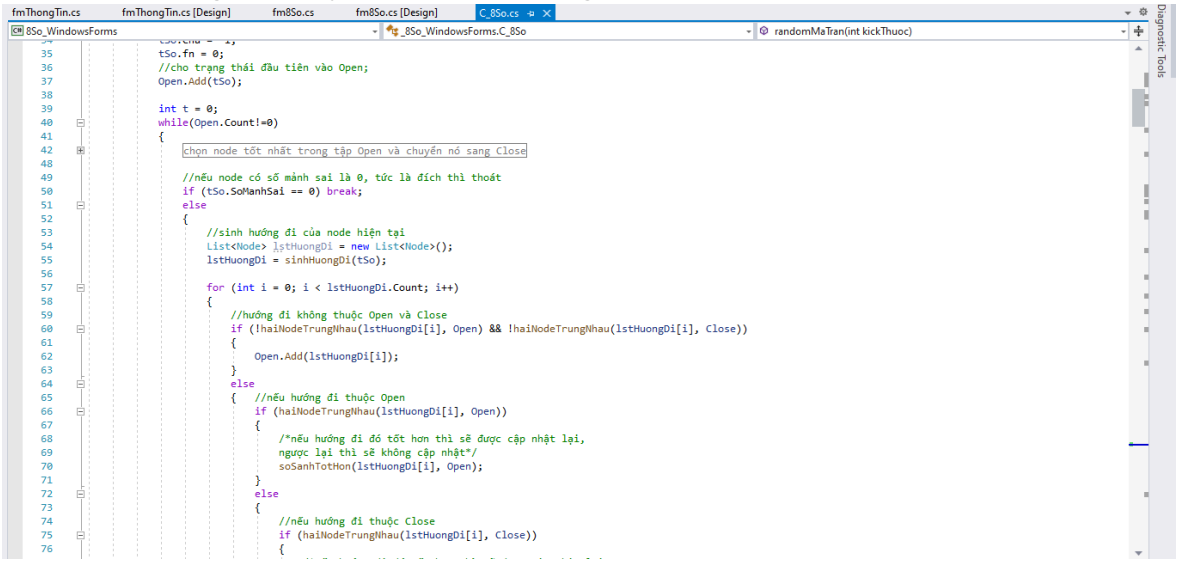
Một số hình ảnh code:

+ Khởi tạo các giá trị đối tượng tính toán:

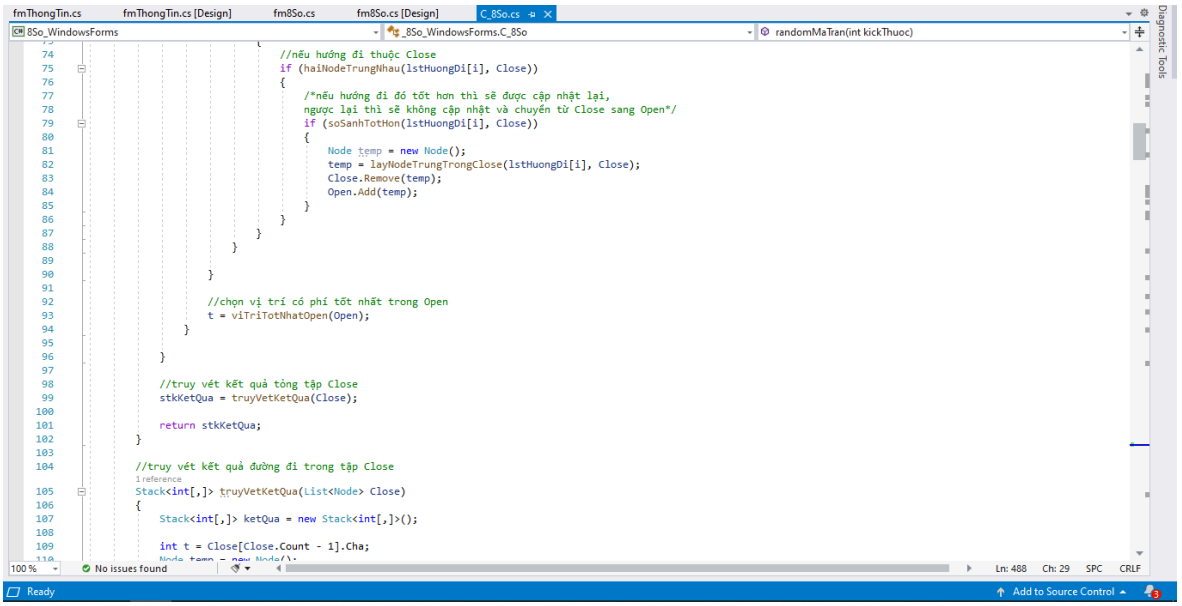


```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace _8So_WindowsForms
{
    35 references
    public class Node
    {

        public int[,] MaTran;// ma trận 8 số
        public int SoManhSai;//số mảnh sai vị trí của ma trận
        public int ChiSo;// chỉ số của node
        public int Cha;// cha của node, để truy vết kết quả
        public int fn;// chi phí đi đến node đó
    }
    2 references
    public class C_8So
    {
        private int ChiSo = 0;//chỉ số của Node sẽ tăng sau mỗi lần sinh ra 1 node
        private int fn = 0;// Sau mỗi lần sinh ra các node thì chi phí các node tăng 1 đơn vị, tức node con sẽ có chi phí lớn hơn node cha 1 đơn vị
        1 reference
        public Stack<int[,]> timKetQua(int[,] MaTran, int n)
        {

            Stack<int[,]> stkKetQua = new Stack<int[,]>();

            List<Node> Close = new List<Node>();
            List<Node> Open = new List<Node>();

            //khai báo và khởi tạo cho node đầu tiên
            Node tSo = new Node();
            tSo.MaTran = MaTran;
            tSo.SoManhSai = soMiengSaiViTri(MaTran);
            tSo.ChiSo = 0;
            tSo.Cha = -1;
            tSo.fn = 0;
```

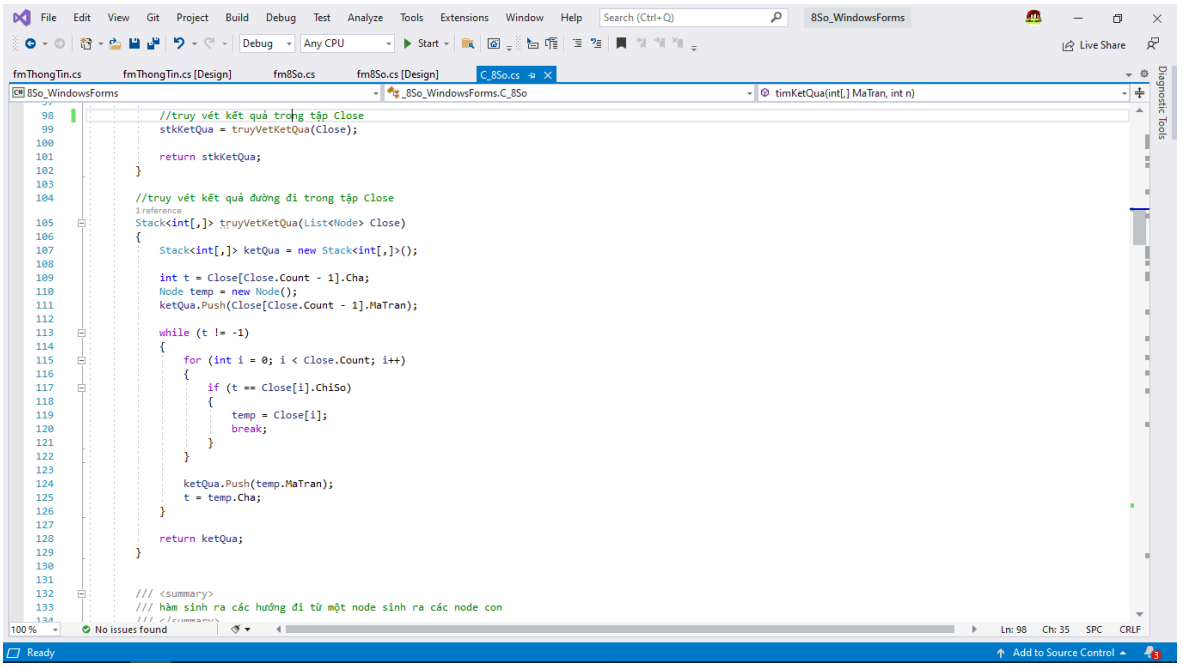
+ Điều chỉnh hướng di chuyển của đối tượng:



```
            tSo.fn = 0;
            //cho trạng thái đầu tiên vào Open;
            Open.Add(tSo);

            int t = 0;
            while(Open.Count!=0)
            {
                chọn node tốt nhất trong tập Open và chuyển nó sang Close

                //nếu node có số mảnh sai là 0, tức là đích thì thoát
                if (tSo.SoManhSai == 0) break;
                else
                {
                    //sinh hướng đi của node hiện tại
                    List<Node> lstHuongDi = new List<Node>();
                    lstHuongDi = sinhHuongDi(tSo);

                    for (int i = 0; i < lstHuongDi.Count; i++)
                    {
                        //hướng đi không thuộc Open và Close
                        if (!haiNodeTrungNhau(lstHuongDi[i], Open) && !haiNodeTrungNhau(lstHuongDi[i], Close))
                        {
                            Open.Add(lstHuongDi[i]);
                        }
                        else
                        {   //nếu hướng đi thuộc Open
                            if (haiNodeTrungNhau(lstHuongDi[i], Open))
                            {
                                /*nếu hướng đi đó tốt hơn thì sẽ được cập nhật lại,
                                ngược lại thì sẽ không cập nhật*/
                                soSanhTotHon(lstHuongDi[i], Open);
                            }
                            else
                            {
                                //nếu hướng đi thuộc Close
                                if (haiNodeTrungNhau(lstHuongDi[i], Close))
                                {
```



```
                    //nếu hướng đi thuộc Close
                    if (haiNodeTrungNhau(lstHuongDi[i], Close))
                    {
                        /*nếu hướng đi đó tốt hơn thì sẽ được cập nhật lại,
                        ngược lại thì sẽ không cập nhật và chuyển từ Close sang Open*/
                        if (soSanhTotHon(lstHuongDi[i], Close))
                        {
                            Node temp = new Node();
                            temp = layNodeTrungTrongClose(lstHuongDi[i], Close);
                            Close.Remove(temp);
                            Open.Add(temp);
                        }
                    }
                }
            }

        }

        //chọn vị trí có phí tốt nhất trong Open
        t = viTriTotNhatOpen(Open);
    }

}

//truy vét kết quả tòng tập Close
stkKetQua = truyVetKetQua(Close);

return stkKetQua;
}

//truy vét kết quả đường đi trong tập Close
1 reference
Stack<int[,]> truyVetKetQua(List<Node> Close)
{
    Stack<int[,]> ketQua = new Stack<int[,]>();

    int t = Close[Close.Count - 1].Cha;
```



```
//truy vét kết quả trong tập Close
stkKetQua = truyVetKetQua(Close);

return stkKetQua;
}

//truy vét kết quả đường đi trong tập Close
1 reference
Stack<int[,]> truyVetKetQua(List<Node> Close)
{
    Stack<int[,]> ketQua = new Stack<int[,]>();

    int t = Close[Close.Count - 1].Cha;
    Node temp = new Node();
    ketQua.Push(Close[Close.Count - 1].MaTran);

    while (t != -1)
    {
        for (int i = 0; i < Close.Count; i++)
        {
            if (t == Close[i].ChiSo)
            {
                temp = Close[i];
                break;
            }
        }

        ketQua.Push(temp.MaTran);
        t = temp.Cha;
    }

    return ketQua;
}


/// <summary>
/// hàm sinh ra các hướng đi từ một node sinh ra các node con
```

+ Chọn vị trí chi phí tốt nhất:

```
/// Chọn vị trí có chi phí tốt nhất trong Open
/// </summary>
/// <param name="Open">Tập Open</param>
/// <returns>Vị trí tốt nhất</returns>
1 reference
int viTriTotNhatOpen(List<Node> Open)
{
    if (Open.Count != 0)
    {
        Node min = new Node();
        min = Open[0];
        int vt = 0;

        for (int i = 1; i < Open.Count; i++)
            if (min.SoManhSai > Open[i].SoManhSai)
            {
                min = Open[i];
                vt = i;
            }
            else
            {
                if (min.SoManhSai == Open[i].SoManhSai)
                {
                    if (min.fn > Open[i].fn)
                    {
                        min = Open[i];
                        vt = i;
                    }
                }
            }
        return vt;
    }

    return 0;
}

/// <summary>
```

+ So sánh vị trí Node:

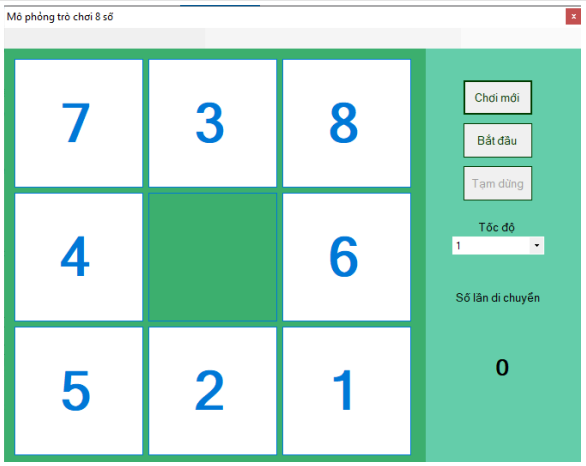
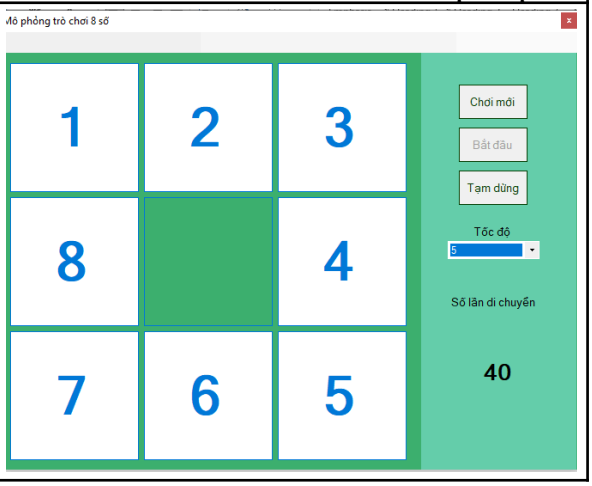
```
/// <summary>
/// So sánh chi phí của hai node
/// </summary>
/// <param name="TamSo">Node cần so sánh</param>
/// <param name="lst8So">Tập Open hoặc Close</param>
/// <returns>trả về true nếu tốt hơn và cập nhật lại cha và chi phí cho node, ngược lại không làm gì và trả về false </returns>
2 references
bool soSanhTotHon(Node tSo, List<Node> lst8So)
{
    for (int i = 0; i < lst8So.Count; i++)
        if (haiMaTranBangNhau(tSo.MaTran, lst8So[i].MaTran))
        {
            if (tSo.fn < lst8So[i].fn)
            {
                //vì 2 ma trận bằng nhau lên số mảnh sai vị trị là như nhau lên ta không cần cập nhật
                lst8So[i].Cha = tSo.Cha;// cập nhật lại cha của hướng đi
                lst8So[i].fn = tSo.fn;// cập nhật lại chi phí đường đi

                return true;
            }
            else return false;
        }

    return false;
}
```

+ Trả về vị trí mảnh nằm sai:

```
/// <summary>
/// trả về số mảnh nằm sai vị trí
/// </summary>
/// <param name="MaTran">Ma trận</param>
/// <returns>Trả về số miếng sai vị trí</returns>
5 references
int soMiengSaiViTri(int[,] MaTran)
{
    int dung = 0;
    int t = 0;
    for (int i = 0; i < MaTran.GetLength(0); i++)
    {
        if (i == 0)
            for (int j = 0; j < MaTran.GetLength(0); j++)
            {
                t++;
                if (MaTran[i, j] == t)
                    dung++;
            }
        else
        {
            if (MaTran[1, 2] == 4)
                dung++;
            if (MaTran[2, 2] == 5)
                dung++;
            if (MaTran[2, 1] == 6)
                dung++;
            if (MaTran[2, 0] == 7)
                dung++;
            if (MaTran[1, 0] == 8)
                dung++;

            break;
        }
    }
    return 8-dung;
}
```

Chú thích đã được thêm vào từng vùng code giảng viên có thể xem trong chương trình Demo.

Hình ảnh Demo:

Màn hình khởi đầu	Màn hình khi hoàn thành sắp xếp
	

Có kèm video Demo.

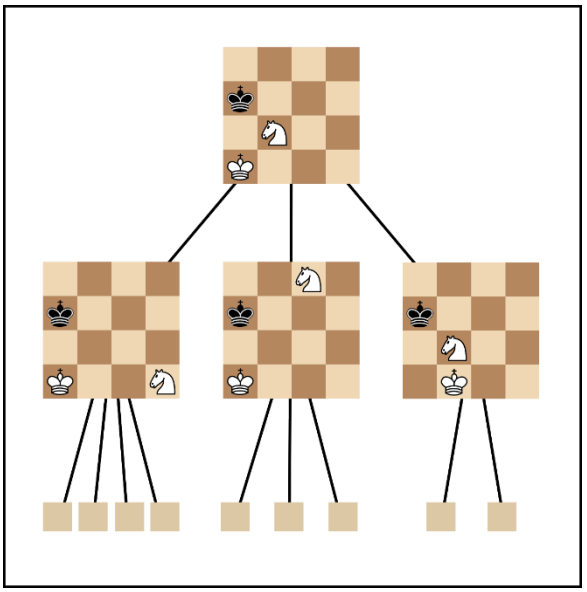
2. Theory of game (Lý thuyết về trò chơi)

2.1. Thuật toán Minimax

Minimax là giải thuật là một thuật toán đệ quy lựa chọn bước đi kế tiếp trong một trò chơi có hai người bằng cách định giá trị cho các Node trên cây trò chơi sau đó tìm Node có giá trị phù hợp để đi bước tiếp theo.

Quá trình chơi cờ là quá trình quân trắng và đen thay phiên nhau đưa ra quyết định, thực hiện một trong số các nước đi hợp lệ. Trên cây trò chơi, quá trình đó sẽ tạo ra đường đi từ gốc đến lá. Giả sử tới một thời điểm nào đó, đường đi đã dẫn tới đỉnh u. Nếu u là đỉnh trắng (đen) thì trắng (đen) cần chọn đi tới một trong các đỉnh đen (trắng) v là con của u. Tại đỉnh đen (trắng) v mà trắng (đen) vừa chọn, đen (trắng) sẽ phải chọn đi tới một trong các đỉnh trắng (đen) w là con của v. Quá trình trên sẽ dừng lại khi đạt tới một đỉnh là lá của cây.

Giả sử quân trắng cần tìm nước đi tại đỉnh u. Nước đi tối ưu cho quân trắng là nước đi dần tới đỉnh con của v là đỉnh tốt nhất(cho trắng) trong số các đỉnh con của u. Ta cần giả thiết rằng, đến lượt đối thủ chọn nước đi từ v, đen cũng sẽ chọn nước đi tốt nhất cho a ta. Như vậy, để chọn nước đi tối ưu cho trắng tại đỉnh u, ta cần phải xác định giá trị các đỉnh của trò chơi gốc u. Giá trị của các đỉnh lá là giá trị của hàm kết cuộc. Đỉnh có giá trị càng lớn càng tốt cho trắng, đỉnh có giá trị càng nhỏ càng tốt cho đen, Để xác định giá trị các đỉnh của cây trò chơi gốc u, ta đi từ mức thấp nhất lên gốc u. Giả sử v là đỉnh trong của cây và giá trị các đỉnh con của nó đã được xác định. Khi đó nếu v là đỉnh trắng thì giá trị của nó được xác định là giá trị lớn nhất trong các giá trị của các đỉnh con. Còn nếu v là đỉnh đen thì giá trị nhỏ nhất trong các giá trị của các đỉnh con.



Để cài đặt kĩ thuật Minimax, việc gán giá trị cho các đỉnh được thực hiện bởi các hàm đệ quy Maxval và Minval. Hàm MaxVal xác định giá trị cho các đỉnh trắng, hàm MinVal xác định giá trị cho các đỉnh đen.

2.2. Tinh chỉnh Alpha-beta (cắt tỉa Alpha- beta)

Việc tinh chỉnh Alpha-beta là một phương pháp tối ưu hóa thuật toán Minimax cho phép chúng ta bỏ qua một số hướng trong tất cả hướng đi có thể. Điều này giúp ta dự đoán hướng bằng minimax hiệu quả, trong khi sử dụng cùng một thuật toán.

Việc giảm thiểu alpha-beta dựa trên tình huống mà chúng ta có thể ngừng đưa ra hướng đi nếu chúng ta thấy hướng đi đó dẫn đến một kết quả không mong đợi. Việc điều chỉnh alpha-beta làm cho thuật toán nhanh hơn.

2.3. Chương trình demo: Cờ vua dùng thuật toán Minimax

Ngôn ngữ sử dụng: Javascript

Giao diện: Website (HTML+CSS)

Một số hình ảnh code:

Triển khai thuật toán Minimax: kèm theo các giá trị liên quan như alpha, beta (hỗ trợ cắt nhánh dư thừa) - file nguồn script.js.

```
/*The "AI" part starts here */

var minimaxRoot =function(depth, game, isMaximisingPlayer) {

    var newGameMoves = game.ugly_moves();
    var bestMove = -9999;
    var bestMoveFound;

    for(var i = 0; i < newGameMoves.length; i++) {
        var newGameMove = newGameMoves[i]
        game.ugly_move(newGameMove);
        var value = minimax(depth - 1, game, -10000, 10000, !isMaximisingPlayer);
        game.undo();
        if(value >= bestMove) {
            bestMove = value;
            bestMoveFound = newGameMove;
        }
    }
```

```
    }
    return bestMoveFound;
};

var minimax = function (depth, game, alpha, beta, isMaximisingPlayer) {
    positionCount++;
    if (depth === 0) {
        return -evaluateBoard(game.board());
    }

    var newGameMoves = game.ugly_moves();

    if (isMaximisingPlayer) {
        var bestMove = -9999;
        for (var i = 0; i < newGameMoves.length; i++) {
            game.ugly_move(newGameMoves[i]);
            bestMove = Math.max(bestMove, minimax(depth - 1, game, alpha, beta, !isMaximisingPlayer));
            game.undo();
            alpha = Math.max(alpha, bestMove);
            if (beta <= alpha) {
                return bestMove;
            }
        }
        return bestMove;
    } else {
        var bestMove = 9999;
        for (var i = 0; i < newGameMoves.length; i++) {
            game.ugly_move(newGameMoves[i]);
            bestMove = Math.min(bestMove, minimax(depth - 1, game, alpha, beta, !isMaximisingPlayer));
            game.undo();
            beta = Math.min(beta, bestMove);
            if (beta <= alpha) {
                return bestMove;
            }
        }
        return bestMove;
    }
};

var evaluateBoard = function (board) {
```

Khai báo các đối tượng trên bàn cờ: màu cờ (trắng-đen), loại cờ (tốt, xe, mã, tượng(tịnh), hậu, vua) – file nguồn chess.js.

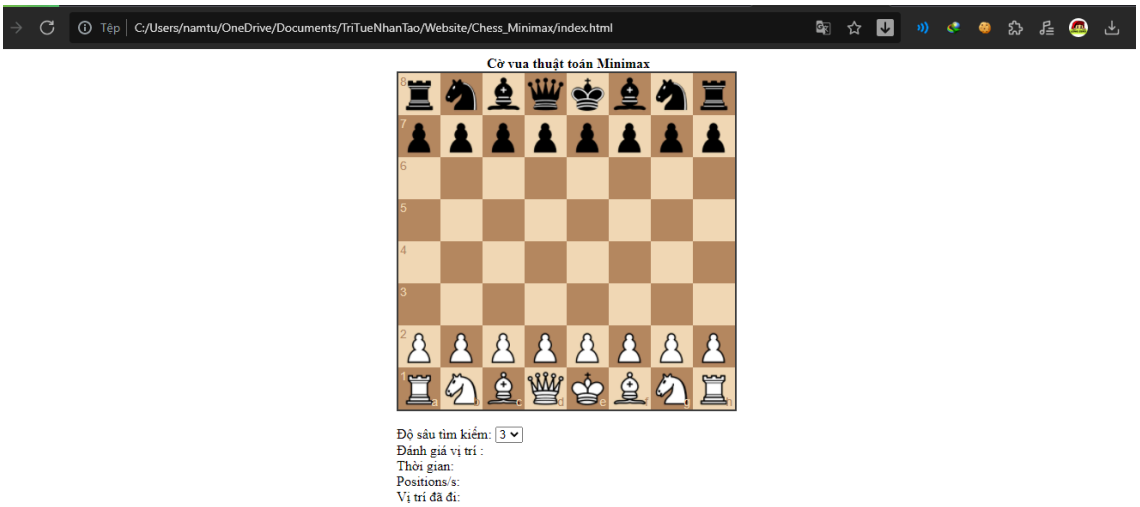
```
var Chess = function(fen) {

    /* jshint indent: false */

    var BLACK = 'b';
    var WHITE = 'w';

    var EMPTY = -1;

    var PAWN = 'p';
    var KNIGHT = 'n';
    var BISHOP = 'b';
    var ROOK = 'r';
    var QUEEN = 'q';
    var KING = 'k';

    var SYMBOLS = 'pnbrqkPNBRQK';

    var DEFAULT_POSITION = 'rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1';

    var POSSIBLE_RESULTS = ['1-0', '0-1', '1/2-1/2', '*'];

    var PAWN_OFFSETS = {
        b: [16, 32, 17, 15],
        w: [-16, -32, -17, -15]
    };

    var PIECE_OFFSETS = {
        n: [-18, -33, -31, -14,  18, 33, 31,  14],
        b: [-17, -15,  17,  15],
        r: [-16,   1,  16,  -1],
        q: [-17, -16, -15,   1,  17, 16, 15,  -1],
        k: [-17, -16, -15,   1,  17, 16, 15,  -1]
    };

    var ATTACKS = [
        20, 0, 0, 0, 0, 0, 0, 24,  0, 0, 0, 0, 0, 0,20, 0,
        0,20, 0, 0, 0, 0, 0, 24,  0, 0, 0, 0, 0,20, 0, 0,
```

JavaScript file | length : 53.108 lines : 1.640 | Ln : 4 Col : 1 Pos : 31 | Unix (LF) | UTF-8 | INS

Lưu những vị trí tối ưu nhất (theo Wikipedia) dưới dạng mảng nhằm đạt được hiệu quả cao nhất (những vị trí nằm giữa bàn cờ sẽ cho nhiều lựa chọn – đường đi tối ưu hơn) – file nguồn script.js.

```
var knightEval =
    [
        [-5.0, -4.0, -3.0, -3.0, -3.0, -3.0, -4.0, -5.0],
        [-4.0, -2.0,  0.0,  0.0,  0.0,  0.0, -2.0, -4.0],
        [-3.0,  0.0,  1.0,  1.5,  1.5,  1.0,  0.0, -3.0],
        [-3.0,  0.5,  1.5,  2.0,  2.0,  1.5,  0.5, -3.0],
        [-3.0,  0.0,  1.5,  2.0,  2.0,  1.5,  0.0, -3.0],
        [-3.0,  0.5,  1.0,  1.5,  1.5,  1.0,  0.5, -3.0],
        [-4.0, -2.0,  0.0,  0.5,  0.5,  0.0, -2.0, -4.0],
        [-5.0, -4.0, -3.0, -3.0, -3.0, -3.0, -4.0, -5.0]
    ];

var bishopEvalWhite = [
    [ -2.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -2.0],
    [ -1.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0, -1.0],
    [ -1.0,  0.0,  0.5,  1.0,  1.0,  0.5,  0.0, -1.0],
    [ -1.0,  0.5,  0.5,  1.0,  1.0,  0.5,  0.5, -1.0],
    [ -1.0,  0.0,  1.0,  1.0,  1.0,  1.0,  0.0, -1.0],
    [ -1.0,  1.0,  1.0,  1.0,  1.0,  1.0,  1.0, -1.0],
    [ -1.0,  0.5,  0.0,  0.0,  0.0,  0.0,  0.5, -1.0],
    [ -2.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -2.0]
];

var bishopEvalBlack = reverseArray(bishopEvalWhite);

var rookEvalWhite = [
    [  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0],
    [  0.5,  1.0,  1.0,  1.0,  1.0,  1.0,  1.0,  0.5],
    [ -0.5,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0, -0.5],
    [ -0.5,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0, -0.5],
    [ -0.5,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0, -0.5],
    [ -0.5,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0, -0.5],
    [ -0.5,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0, -0.5],
    [  0.0,  0.0,  0.0,  0.5,  0.5,  0.0,  0.0,  0.0]
];
```

Hình ảnh Demo:



Video Demo được lưu trong thư mục Video.

3. Logic and Semantic (Logic và ngữ nghĩa)

3.1. Giải thuật Vương Hạo

Thuật giải Vương Hạo là một thuật giải khác để chứng minh việc suy dẫn mệnh đề (có cùng mục đích với Robinson):

Bước 1: Đưa bài toán cần chứng minh về dạng chuẩn:

$$GT_1, GT_2, ..., GT_n \Rightarrow KL_1, KL_2, ..., KL_m$$

Trong đó các GT_i và $_j$ KL là các câu chỉ gồm các phép $\wedge$, $\vee$, $\neg$, không chứa phép $\Rightarrow$ hay $\Leftrightarrow$. Lưu ý: dấu phẩy (,) ở vế trái tương đương với $\wedge$, ở vế phải tương đương với $\vee$.

Bước 2: Nếu tồn tại một câu có phép $\neg$ ở đầu thì chuyển về câu và loại bỏ phép $\neg$

Bước 3: Thay các dấu $\wedge$ ở vế trái và các dấu $\vee$ ở vế phải bằng dấu phẩy (,). Khi đó vế trái chỉ còn dấu $\vee$ và $\neg$, về phải chỉ còn dấu $\wedge$ và $\neg$.

Bước 4:

Nếu dòng hiện hành có dạng:

$$GT_1,...,GT_k,A \vee B,GT_{k+1},...,GT_n \Rightarrow KL_1,KL_2,...,KL_m$$

thì thay bằng hai dòng:

$$GT_1,...,GT_k,A,GT_{k+1},...,GT_n \Rightarrow KL_1,KL_2,...,KL_m$$

và

$$GT_1,...,GT_k,B,GT_{k+1},...,GT_n \Rightarrow KL_1,KL_2,...,KL_m$$

Nếu dòng hiện hành có dạng:

$$GT_1,GT_2,...,GT_n \Rightarrow KL_1,...,KL_k,A \wedge B,KL_{k+1},...,KL_m$$

thì thay bằng hai dòng:

$$GT_1,GT_2,...,GT_n \Rightarrow KL_1,...,KL_k,A,KL_{k+1},...,KL_m$$

và

$$GT_1,GT_2,...,GT_n \Rightarrow KL_1,...,KL_k,B,KL_{k+1},...,KL_m$$

Bước 5: Một dòng được chứng minh nếu tồn tại một mệnh đề ở cả hai vế.

Bước 6: Một dòng không thể tách cũng không thể chuyển vế dấu ¬ mà không có biến mệnh đề chung ở cả hai vế thì không được chứng minh.

Lặp lại bước 2 đến 6 cho tới khi mọi dòng được chứng minh hay tồn tại một dòng không được chứng minh. Bài toán ban đầu được chứng minh nếu mọi dòng tách ra từ nó được chứng minh.

3.2. Chương trình demo: Mô phỏng thuật toán Vương Hạo trong Prolog

Ngôn ngữ sử dụng: Prolog

Giao diện: SWI-Prolog.

Cách sử dụng:

Một số kí tự trong mệnh đề khá khó khăn để sử dụng nên phần mềm chỉ nhận những kí tự tương đương trong bảng:

Phép tính mệnh đề	Kí hiệu toán học	Kí hiệu chương trình
Phép phủ định (negation)	¬	!
Phép hội (conjunction)	∧	&
Phép tuyển (disjunction)	∨	v
Phép kéo theo (implication)	→	->
Phép tương đương (equivalence)	↔	<->
Phép tuần tự (sequent)	⊢	|=

Một số hình ảnh code:

+ Xác định các toán tử

```
:-op(700,xfy,<->)
:-op(700,xfy,->)
:-op(600,xfy,v)
:-op(600,xfy,&)
:-op(500,fy,!)
```

+ Thực thi 7 luật suy diễn trong thuật toán Vương Hạo.

vuonghao.pl

File Edit Browse Compile Prolog Pce Help

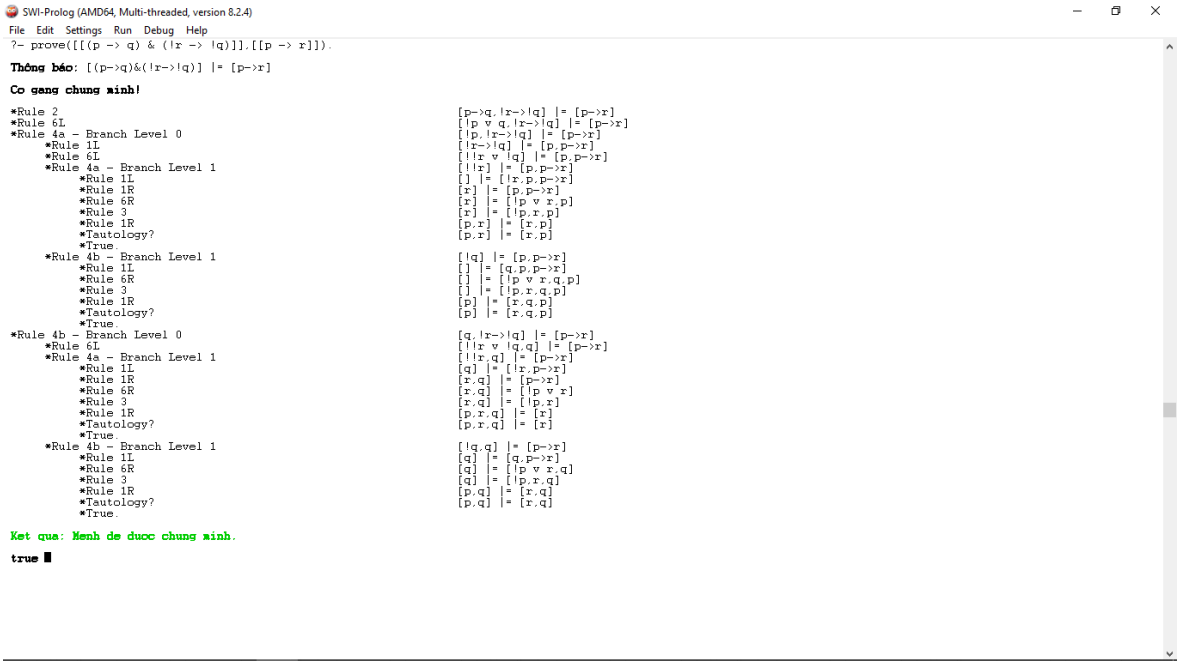
vuonghao.pl

```
% Thuat toan Vuong Hao.
% Xac dinh toan tu.
:-op(700,xfy,<->).
:-op(700,xfy,->).
:-op(600,xfy,v).
:-op(600,xfy,&).
:-op(500,fy,!).

% Lenh goi chinh cho chuong trinh.
prove([],[]).
prove([L|P],[R|A]):-
    nl, ansi_format([bold],'Statement: ',[]), write(L), write(' |= '), write(R), nl, nl,
    ansi_format([bold],'Attempting proof!',[]), nl, nl,
    wang(L,R),
    prove(P,A).

% Thu tuc thuat toan de chung minh menh de.
wang(L,R):-
    rules(L,R,[],0), nl,
    ansi_format([bold,fg(green)], 'Result: The given theorem is true.',[]), nl, nl;
    ansi_format([bold, fg(red)],'Proof failed for current step!',[]), nl, nl,
    ansi_format([bold,fg(red)], 'Result: The given theorem is false.',[]), nl, nl.

% Di chuyen phu dinh tu trai sang phai.
rules(L,R,S,T):-
    member(!X,L),
    delete(L,!X,Ld),
    append(S,[[['*Rule 1L                  '],[Ld,' |= ',[X|R]],T]],Sa),
    rules(Ld,[X|R],Sa,T).

% Di chuyen phu dinh tu phai qua trai.
rules(L,R,S,T):-
    member(!X,R),
    delete(R,!X,Rd),
    append(S,[[['*Rule 1R                  '],[[X|L],' |= ',Rd],T]],Sa),
    rules([X|L],Rd,Sa,T).

% Thay the phep keo theo ben trai.
rules(L,R,S,T):-
    member(X -> Y,L),
    delete(L,X -> Y,Ld),
    append(S,[[['*Rule 6L                  '],[[!X v Y|Ld],' |= ',R],T]],Sa),
    rules([!X v Y|Ld],R,Sa,T).

% Thay the phep keo theo ben phai.
rules(L,R,S,T):-
    member(X -> Y,R),
    delete(R,X -> Y,Rd),
  append(S,[[['*Rule 6R                '],[L,' |= ',[!X v Y|Rd]],T]],Sa),
    rules(L,[!X v Y|Rd],Sa,T).

% Thay the phep tuong duong ben trai.
rules(L,R,S,T):-
    member(X <-> Y,L),
    delete(L,X <-> Y,Ld),
    append(S,[[['*Rule 7L                  '],[[(X -> Y) & (Y -> X)|Ld],' |= ',R],T]],Sa),
    rules([(X -> Y) & (Y -> X)|Ld],R,Sa,T).

% Thay the phep tuong duong ben phai.
rules(L,R,S,T):-
    member(X <-> Y,R),
    delete(R,X <-> Y,Rd),
    append(S,[[['*Rule 7R                  '],[L,' |= ',[(X -> Y) & (Y -> X)|Rd]],T]],Sa),
    rules(L,[(X -> Y) & (Y -> X)|Rd],Sa,T).

% So sánh hai ve menh de.
rules(L,R,S,T):-
    append(S,[[['*Tautology?               '],[L,' |= ',R], T]],Sa),
    member(X,L),
    member(X,R),
    append(Sa,[[['*True.                    '],[],T]],Sb),
    printprove(Sb).

% Thay the phep hoi bang du phay o ben trai.
rules(L,R,S,T):-
    member(X & Y,L),
    delete(L,X & Y,Ld),
    append(S,[[['*Rule 2                   '],[[X,Y|Ld],' |= ',R],T]],Sa),
    rules([X,Y|Ld],R,Sa,T).

% Thay the phep tuyen bang dau phay o ben phai.
rules(L,R,S,T):-
    member(X v Y,R),
    delete(R,X v Y,Rd),
    append(S,[[['*Rule 3                   '],[L,' |= ',[X,Y|Rd]],T]],Sa),
    rules(L,[X,Y|Rd],Sa,T).

% Tach phep tuyen ben trai.
rules(L,R,S,T):-
    member(X v Y,L),
    delete(L,X v Y,Ld),
    Ta is T + 1,
    append(S,[[['*Rule 4a - Branch Level ',T],[[X|Ld],' |= ',R],T]],Sa),
    rules([X|Ld],R,Sa,Ta),
    Tb is T + 1,
    append([],[[['*Rule 4b - Branch Level ',T],[[Y|Ld],' |= ',R],T]],Sb),
    rules([Y|Ld],R,Sb,Tb).

% Tach phep hoi ben phai.
rules(L,R,S,T):-
    member(X & Y,R),
    delete(R,X & Y, Rd),
    append(S,[[['*Rule 5a - Branch Level ',T],[L,' |= ',[X|Rd]],T]],Sa),
    Ta is T + 1,
    rules(L,[X|Rd],Sa,Ta),
    Tb is T + 1,
    append([],[[['*Rule 5b - Branch Level ',T],[L,' |= ',[Y|Rd]],T]],Sb),
    rules(L,[Y|Rd],Sb,Tb).
```

```
% Neu menh de dung thi in ra dieu phai chung minh.
printprove([]).
printprove([[P,Q,T]|S]):-
  tab(T*5), printlist(P), tab(40 - T*5), printlist(Q), nl,
    printprove(S).

% In danh sach theo de quy.
printlist([]).
printlist([H|T]):-
    write(H),
    printlist(T).
```

Line: 108

Hình ảnh Demo:



Video thực hiện thử một số mệnh đề được lưu trong thư mục Video.

4. Simulation (Mô phỏng)

4.1. Genetic Algorithm applied to optimization (Thuật toán di truyền áp dụng để tối ưu hoá)

Thuật toán di truyền (GA) là thuật toán tìm kiếm dựa trên các khái niệm về chọn lọc tự nhiên và di truyền. GAs là một tập con của một nhánh tính toán lớn hơn nhiều được gọi là Tính toán tiến hóa.

Trong GAs, ta có một nhóm các giải pháp khả thi cho vấn đề đã cho. Những dung dịch này sau đó trải qua quá trình tái tổ hợp và đột biến (giống như trong di truyền tự nhiên), tạo ra những đứa trẻ mới và quá trình này được lặp lại trong nhiều thế hệ khác nhau. Mỗi cá thể (hoặc giải pháp ứng cử viên) được chỉ định một giá trị phù hợp (dựa trên giá trị hàm mục tiêu của nó) và các cá thể lai tạo có cơ hội cao hơn để giao phối và sinh ra các cá thể phù hợp. Điều này phù hợp với Học thuyết của Darwin về sự sống còn của những cá thể khoẻ mạnh nhất.

Do đó, nó tiếp tục phát triển các cá nhân hoặc giải pháp tốt hơn qua nhiều thế hệ, cho đến khi đạt được tiêu chí dừng. Thuật toán di truyền về bản chất là đủ ngẫu nhiên, nhưng chúng hoạt động tốt hơn nhiều so với tìm kiếm cục bộ ngẫu nhiên (nơi ta chỉ thử các giải pháp ngẫu nhiên, theo dõi các giải pháp tốt nhất cho đến nay), vì chúng cũng khai thác thông tin lịch sử.

4.2. Chương trình demo: Tìm đường đi sử dụng thuật toán di truyền

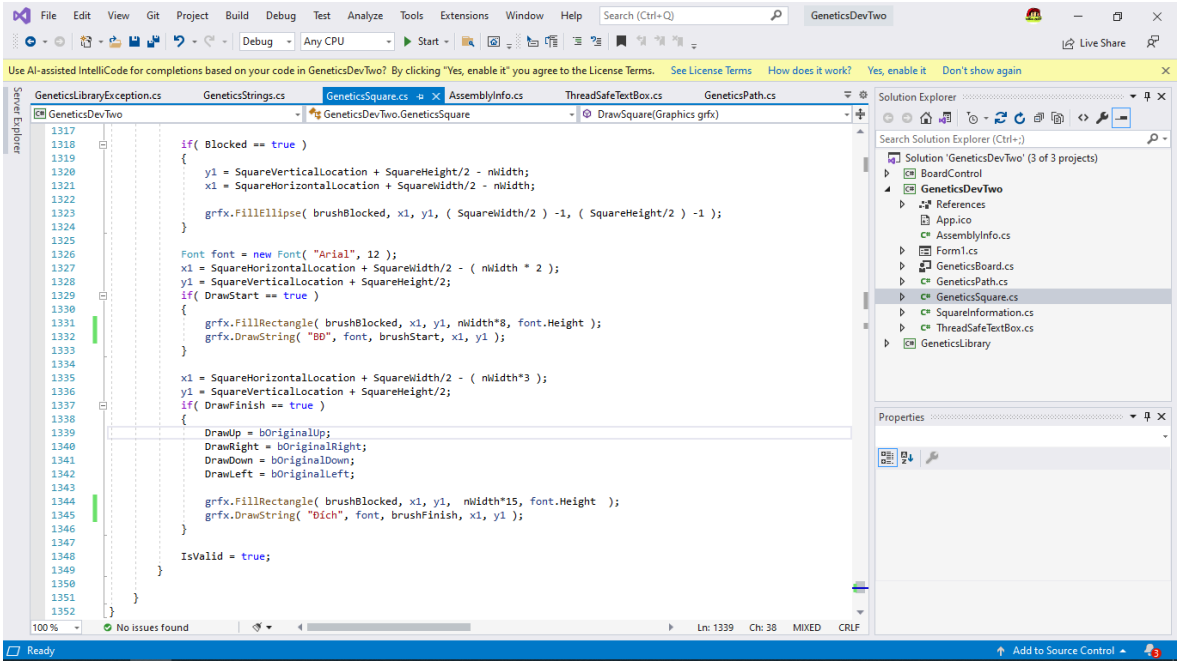
Ngôn ngữ sử dụng: C#

Giao diện: Winform

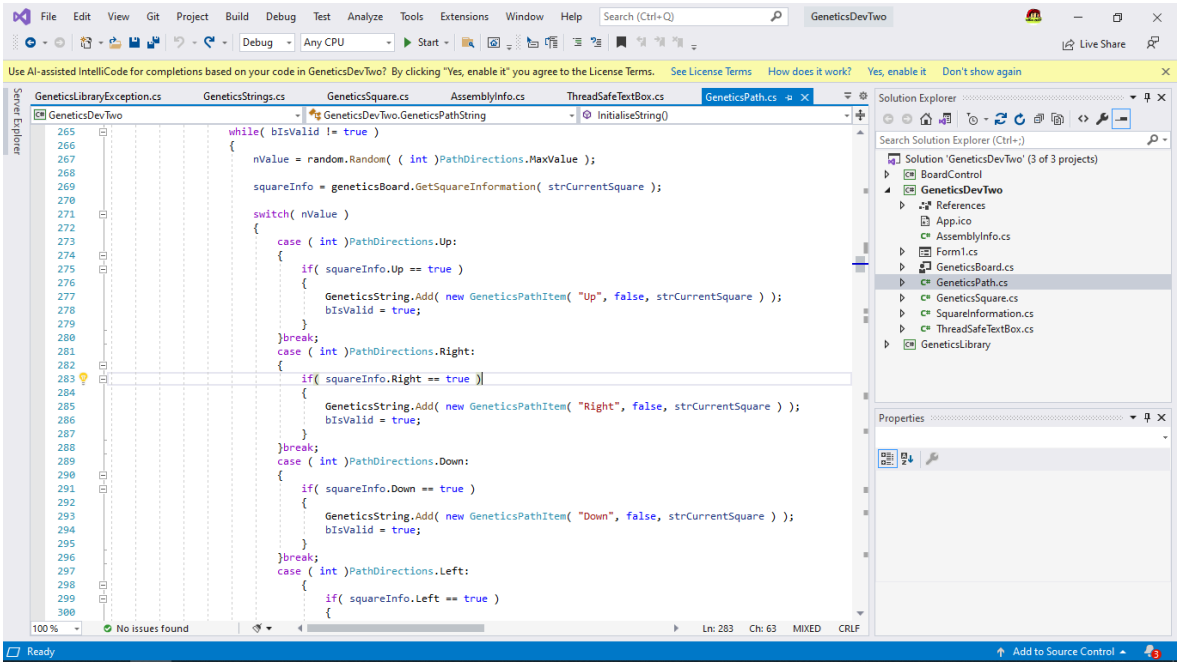
Mô tả: Project gồm nhiều file chức năng khác nhau phân là 3 mục chính: Control (điều khiển), GeneticsDevTwo (Chương trình chính) và Library (Thư viện chứa các chức năng của thuật toán).

Một số hình ảnh code:

+ Khai báo thông số cho điểm Bắt đầu (BĐ), điểm Đích cùng một vài thông số về vị trí – file GeneticsSquare.cs.



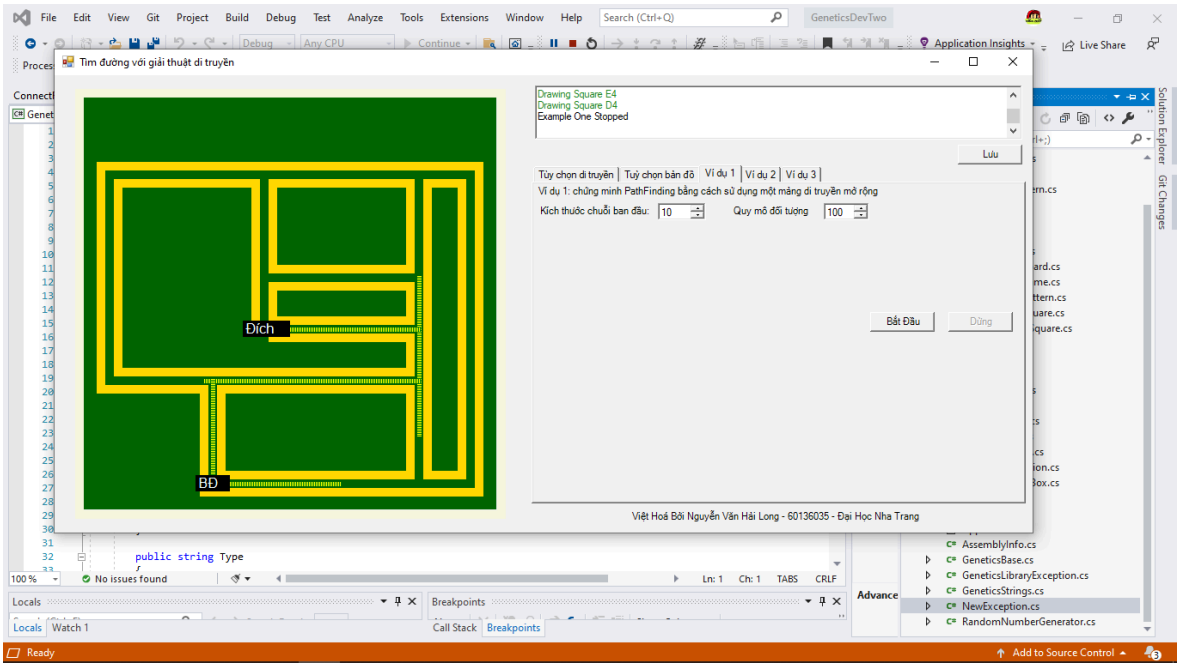
+ Thực thi các đường đi một cách ngẫu nhiên theo 4 hướng (tràn ra 4 hướng theo quy mô đối tượng) – file GeneticsPath.cs.



+ Các file thực thi khác:

- ThreadSafeTextBox.cs : TextBox chứ thông tin tìm đường (hướng đi).
- SquareInformation.cs: Nơi khai báo các biến mang giá trị Boolean.

Hình ảnh Demo:



Video demo được lưu trữ trong thư mục Video.

5. IoT and Robotics applied in with AI (IoT và Robotics được ứng dụng với AI)

5.1. Sử dụng Arduino để chế tạo cánh tay robot

Do tình hình dịch phức tạp nên em không có sử dụng được demo bằng phần cứng nên em gửi các file liên quan đến về chủ đề.

Các thiết bị cần thiết:

- Arduino UNO

- Module Bluetooth HC05
- Động cơ RC Servo MG996
- Động cơ RC Servo 9G
- Điện trở 1K Ohm
- Điện trở 2K Ohm
- Adapter 5V - 2A
- Jack cắm DC cái
- Breadboard 400 chân
- Dây bus đực - đực

Phần mềm yêu cầu:

- **Arduino IDE:** Arduino IDE là một phần mềm lập trình dựa trên ngôn ngữ lập trình C. Bằng cách viết những câu lệnh có cấu trúc tương tự cấu trúc câu lệnh trong C, chúng ta có thể điều khiển hoặc thu nhận các dữ liệu từ các module, cảm biến thông qua vi điều khiển arduino.
- **MIT App Inventor:** là một ứng dụng web nguồn mở ban đầu được cung cấp bởi Google và hiện tại được duy trì bởi Viện Công nghệ Massachusetts (MIT). Nền tảng cho phép nhà lập trình tạo ra các ứng dụng phần mềm cho hệ điều hành Android (OS). Bằng cách sử dụng giao diện đồ họa, nền tảng cho phép người dùng kéo và thả các khối mã (blocks) để tạo ra các ứng dụng có thể chạy trên thiết bị Android. Nguồn trang: http://ai2.appinventor.mit.edu.

5.2. Chương trình demo: Cánh tay robot

Lập trình với Arduino IDE:

Đầu tiên ta khai báo thư viện SoftwareSerial cho giao tiếp nối tiếp của mô-đun Bluetooth cũng như thư viện servo. Cả hai thư viện này đều được tích hợp sẵn với Arduino IDE nên ta không cần phải cài đặt chúng bên ngoài. Sau đó, ta cần xác định sáu servos, mô-đun Bluetooth HC-05 và một số biến để lưu trữ vị trí hiện tại và trước đó của servos, cũng như các mảng để lưu trữ các vị trí hoặc các bước cho chế độ tự động.

```
#include <SoftwareSerial.h>
#include <Servo.h>

Servo servo01;
Servo servo02;
Servo servo03;
Servo servo04;
Servo servo05;
Servo servo06;

SoftwareSerial Bluetooth(3, 4); // Arduino(RX, TX) - HC-05 Bluetooth (TX, RX)

int servo1Pos, servo2Pos, servo3Pos, servo4Pos, servo5Pos, servo6Pos;// Vị trsi hiện tại
int servo1PPos, servo2PPos, servo3PPos, servo4PPos, servo5PPos, servo6PPos; // Vị trí trước đó
int servo01SP[50], servo02SP[50], servo03SP[50], servo04SP[50], servo05SP[50], servo06SP[50];// để lưu trữ các vị trí / bước
int speedDelay = 20;
int index = 0;
String dataIn = "";
```

Trong phần thiết lập, ta cần khởi tạo servos và mô-đun Bluetooth và di chuyển cánh tay robot về vị trí ban đầu của nó. Sử dụng hàm write () di chuyển servo đến bất kỳ vị trí nào từ 0 đến 180 độ.

```
void setup() {
  servo01.attach(5);
  servo02.attach(6);
  servo03.attach(7);
  servo04.attach(8);
  servo05.attach(9);
  servo06.attach(10);
  Bluetooth.begin(38400);// Tốc độ truyền mặc định của mô-đun Bluetooth
  Bluetooth.setTimeout(1);
  delay(20);
  // Vị trí ban đầu của cánh tay robot
  servo1PPos = 90;
  servo01.write(servo1PPos);
  servo2PPos = 150;
  servo02.write(servo2PPos);
  servo3PPos = 35;
  servo03.write(servo3PPos);
  servo4PPos = 140;
  servo04.write(servo4PPos);
  servo5PPos = 85;
  servo05.write(servo5PPos);
  servo6PPos = 80;
  servo06.write(servo6PPos);
}
```

Tiếp đó khởi tạo vòng lặp, bằng cách sử dụng chức năng Bluetooth.available (),chương trình sẽ liên tục kiểm tra xem có bất kỳ dữ liệu nào đến từ Điện thoại thông minh hay không. Nếu đúng, sử dụng hàm readString (), chương trình đọc dữ liệu dưới dạng chuỗi và lưu trữ nó vào biến dataIn. Tùy thuộc vào dữ liệu đã đến, chương trình sẽ điều khiển cho cánh tay robot biết phải làm gì.

```
//Kiểm tra dữ liệu đến
  if (Bluetooth.available() > 0) {
    dataIn = Bluetooth.readString(); // Đọc dữ liệu dưới dạng chuỗi
```

Sử dụng hàm startedWith (), chương trình kiểm tra tiền tố của mỗi dữ liệu đến và vì vậy nó biết phải làm gì tiếp theo.

```
// Nếu thanh trượt "Waist" đã thay đổi giá trị - Di chuyển Servo 1 đến vị trí

    if (dataIn.startsWith("s1")) {
      String dataInS = dataIn.substring(2, dataIn.length());// Chỉ trích xuất số
      servo1Pos = dataInS.toInt(); // Chuyển chuỗi sáng số
```

Gọi hàm write () và servo sẽ đi đến vị trí đó, nhưng theo cách đó, servo sẽ chạy ở tốc độ tối đa, quá sức đối với cánh tay robot. Thay vào đó, ta cần kiểm soát tốc độ của servo, vì vậy nên sử dụng một số vòng lặp FOR để dần dần di chuyển servo từ vị trí trước đó đến vị trí hiện tại bằng cách triển khai thời gian trễ giữa mỗi lần lặp. Bằng cách thay đổi thời gian trễ, ta có thể thay đổi tốc độ của servo.

```
// Chúng tôi sử dụng vòng lặp for để chúng tôi có thể kiểm soát tốc độ của servo

     // Nếu vị trí trước đó lớn hơn thì vị trí hiện tại
      if (servo1PPos > servo1Pos) {
        for ( int j = servo1PPos; j >= servo1Pos; j--) {  // Di chuyển servo xuống
          servo01.write(j);
          delay(20);    // defines the speed at which the servo rotates
        }
      }
     // Nếu vị trí trước đó nhỏ hơn thì vị trí hiện tại
      if (servo1PPos < servo1Pos) {
        for ( int j = servo1PPos; j <= servo1Pos; j++) {  // Di chuyển servo lên
          servo01.write(j);
          delay(20);
        }
      }
      servo1PPos = servo1Pos;  // Đặt vị trí hiện tại là vị trí trước đó
    }
```

Thiết đặt lưu thông tin. Nếu chúng ta nhấn nút SAVE, vị trí của mỗi động cơ servo được lưu trong một mảng. Với mỗi lần nhấn, chỉ số tăng lên vì vậy mảng được lấp đầy từng bước.

```
// Nếu nút "SAVE" được nhấn
    if (dataIn.startsWith("SAVE")) {
      servo01SP[index] = servo1PPos; // lưu vị trí vào mảng
      servo02SP[index] = servo2PPos;
      servo03SP[index] = servo3PPos;
      servo04SP[index] = servo4PPos;
      servo05SP[index] = servo5PPos;
      servo06SP[index] = servo6PPos;
      index++;                       // Tăng chỉ số mảng
    }
```

Khai báo nút RUN và thao tác thực thi, gọi hàm tùy chỉnh runningervo () chạy các bước được lưu trữ. Ở đây chương trình chạy đi chạy lại các bước đã lưu trữ cho đến khi chúng tôi nhấn nút RESET. Sử dụng vòng lặp FOR, chạy qua tất cả các vị trí được lưu trữ trong các mảng và đồng thời hàm kiểm tra xem có bất kỳ dữ liệu đến từ điện thoại thông minh hay không. Dữ liệu này có thể là nút RUN / PAUSE, nút này sẽ tạm dừng rô bốt và nếu được nhấp lại sẽ tiếp tục với các chuyển động tự động. Ngoài ra nếu điều khiển tốc độ, chương trình sẽ sử dụng giá trị đó để thay đổi thời gian trễ giữa mỗi lần lặp trong vòng lặp FOR bên dưới, điều khiển tốc độ của động cơ servo.

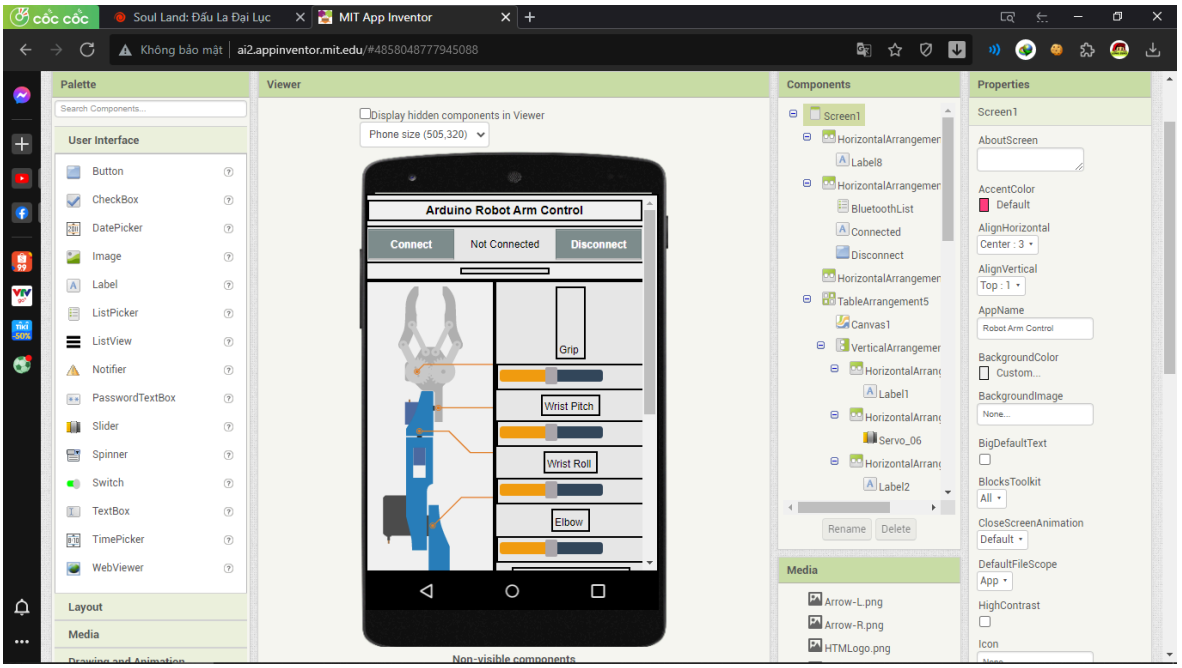
```
// Chức năng tùy chỉnh chế độ tự động - chạy các bước đã lưu
void runservo() {
  while (dataIn != "RESET") {  // Chạy đi chạy lại các bước cho đến khi nhấn nút "RESET"
    for (int i = 0; i <= index - 2; i++) { // Chạy qua tất cả các bước (chỉ mục)
      if (Bluetooth.available() > 0) {     // Kiểm tra dữ liệu nhập
        dataIn = Bluetooth.readString();
        if ( dataIn == "PAUSE") {          // Nếu nút "PAUSE" được nhấn
          while (dataIn != "RUN") {        // Chờ cho đến khi "RUN" được nhấn lại
            if (Bluetooth.available() > 0) {
              dataIn = Bluetooth.readString();
              if ( dataIn == "RESET") {
                break;
              }
            }
          }
        }
       // Nếu thanh trượt SPEED được thay đổi
        if (dataIn.startsWith("ss")) {
          String dataInS = dataIn.substring(2, dataIn.length());
          speedDelay = dataInS.toInt();// Thay đổi tốc độ servo (thời gian trễ)
        }
      }
      // Servo 1
      if (servo01SP[i] == servo01SP[i + 1]) {
      }
      if (servo01SP[i] > servo01SP[i + 1]) {
        for ( int j = servo01SP[i]; j >= servo01SP[i + 1]; j--) {
          servo01.write(j);
          delay(speedDelay);
        }
      }
      if (servo01SP[i] < servo01SP[i + 1]) {
        for ( int j = servo01SP[i]; j <= servo01SP[i + 1]; j++) {
          servo01.write(j);
          delay(speedDelay);
        }
```

Di chuyển các servos đến vị trí tiếp theo của chúng. Sau cùng nếu nhấn nút RESET, chương trình sẽ xóa tất cả dữ liệu từ các mảng và cũng đặt lại chỉ mục về 0 để ta có thể thực hiện các chuyển động mới.

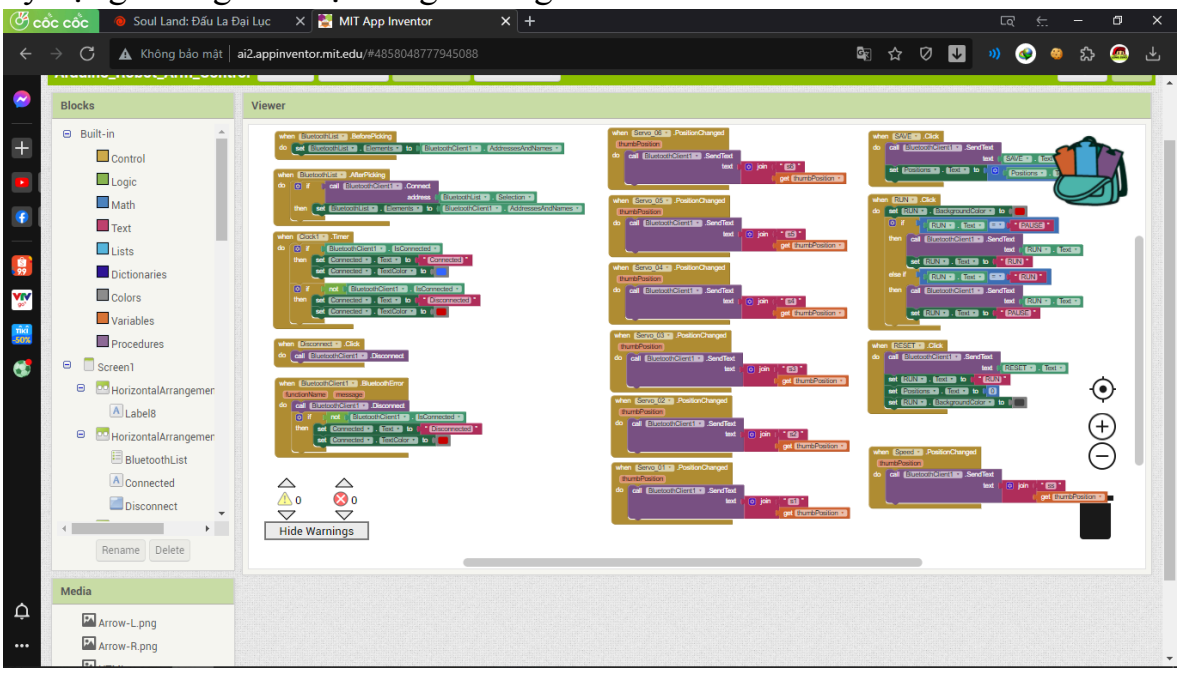
```
// Nếu nút "RESET" được nhấn
    if ( dataIn == "RESET") {
      memset(servo01SP, 0, sizeof(servo01SP));// Xóa dữ liệu mảng thành 0
      memset(servo02SP, 0, sizeof(servo02SP));
      memset(servo03SP, 0, sizeof(servo03SP));
      memset(servo04SP, 0, sizeof(servo04SP));
      memset(servo05SP, 0, sizeof(servo05SP));
      memset(servo06SP, 0, sizeof(servo06SP));
      index = 0; // Chỉ mục thành 0
    }
```

Lập trình MIT App Inventor:

Sử dụng biểu tượng bằng cách kéo thả để tạo giao diện



Xây dựng những khối lệnh logic trong Blocks



File thực thi được lưu trong thư mục B5