**TRƯỜNG ĐẠI HỌC THỦ DẦU MỘT**

**VIỆN KỸ THUẬT - CÔNG NGHỆ**

****

**TIỂU LUẬN MÔN HỌC**

**TRÍ TUỆ NHÂN TẠO**

**ĐỀ TÀI**

**ỨNG DỤNG THUẬT TOÁN**

**VÀO TRÒ CHƠI GHÉP TRANH**

**GVHD: Th.S NGUYỄN LÊ HIỀN DUYÊN**

**SVTH: LÊ THANH HIỆP**

**MSSV: 1824801030217**

**LỚP: D18PM04**

**SVTH: NGUYỄN QUANG TÚ**

**MSSV: 1824801030214**

**LỚP: D18PM04**

**BÌNH DƯƠNG – 3/2021**

**TRƯỜNG ĐẠI HỌC THỦ DẦU MỘT**

**VIỆN KỸ THUẬT - CÔNG NGHỆ**

****

**TIỂU LUẬN MÔN HỌC**

**TRÍ TUỆ NHÂN TẠO**

**ĐỀ TÀI**

**ỨNG DỤNG THUẬT TOÁN**

**VÀO TRÒ CHƠI GHÉP TRANH**

**GVHD: Th.S NGUYỄN LÊ HIỀN DUYÊN**

**SVTH: LÊ THANH HIỆP**

**MSSV: 1824801030217**

**LỚP: D18PM04**

**SVTH: NGUYỄN QUANG TÚ**

**MSSV: 1824801030214**

**LỚP: D18PM04**

**BÌNH DƯƠNG – 3/2021**

**LỜI NHẬN XÉT CỦA GIÁO VIÊN**

MỤC LỤC

[MỤC LỤC ii](#_Toc1711470)

[DANH MỤC HÌNH iii](#_Toc1711471)

[DANH MỤC BẢNG iv](#_Toc1711472)

[MỞ ĐẦU 1](#_Toc1711473)

[CHƯƠNG 1. KHẢO SÁT VÀ PHÂN TÍCH BÀI TOÁN 2](#_Toc1711474)

[1.1. ... 2](#_Toc1711475)

[1.1.1. … 2](#_Toc1711476)

[CHƯƠNG 2. THIẾT KẾ GIAO DIỆN 3](#_Toc1711477)

[2.1. ... 3](#_Toc1711478)

[CHƯƠNG 3. THIẾT KẾ CHƯƠNG TRÌNH 4](#_Toc1711479)

[3.1. ... 4](#_Toc1711480)

[KẾT LUẬN 5](#_Toc1711481)

[TÀI LIỆU THAM KHẢO 6](#_Toc1711482)

DANH MỤC HÌNH

[**Hình 2.1:** 7](#_Toc488530488)

[**Hình 2.2:** 8](#_Toc488530489)

DANH MỤC BẢNG

[**Bảng 3.1:** Mô tả bảng cấu hình hệ thống 19](#_Toc488530548)

[**Bảng 3.2:** 20](#_Toc488530549)

**TỔNG QUAN**

Thông tin nhóm

|  |  |  |  |
| --- | --- | --- | --- |
| MSSV | Họ và tên | Thông tin liên lạc | Vai trò |
| 1824801030217 | LÊ THANH HIỆP | 0901913820 | Leader |
| 1824801030214 | NGUYỄN QUANG TÚ | 0901913820 | **Developer** |

Bảng phân công công việc

|  |  |  |
| --- | --- | --- |
| STT | Người thực hiện | Nội dung thực hiện |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

MỞ ĐẦU

Ngày nay, chúng ta đã bước vào thế kỷ 21, kỷ nguyên của Công nghệ thông tin, đặc biệt là trí tuệ nhân là yếu tố quan trọng nhất quyết định sự thành công của mỗi ngành hay mỗi quốc gia. Trí tuệ nhân tạo đã và đang làm thay đổi cuộc sống của chúng ta, với sự phát triển mạnh mẽ của việc áp dụng các nghiên cứu về trí tuệ nhân tạo áp dụng cho cuộc sống. Tất cả các ngành như: Quân đội, y tế, giáo dục, kinh tế thương mại, tài chính,… Đều có thể áp dụng trí tuệ nhân một cách rộng rãi, Việc áp dụng trí tuệ nhân tạo để giải quyết các vấn đề trong xã hội và việc phát triển kinh tế đang được nhà nước khuyến khích và đầu tư rất lớn.

Trên thế giới cũng như Việt Nam, CNTT có ảnh hưởng rất mạnh mẽ đến sự phát triển của đất nước và thế giới đặc biệt là trí tuệ nhân tạo. Nó chở thành một yếu tố không thể thiếu và có tính quyết định đến sự thành công hay thất bại của nhiều ngành ở nước ta, CNTT đang phát triển với tốc độ khá mạnh mẽ và được ứng dụng rộng rãi trong tất cả các lĩnh vực, đặc biệt là trong công tác ứng dụng công nghệ vào trong cuộc sống.

Như chúng ta đã biết, sức mạnh của một nền kinh tế phụ thuộc rất lớn vào các hoạt động trong nước của các doanh nghiệp, vì vậy sự thành công trong kinh doanh của doanh nghiệp không những là mục tiêu của riêng doanh nghiệp, mà nó còn là nhân tố quyết định vị thế của đất nước trên trường quốc tế. Việc đưa AI vào áp dụng cho các doanh nghiệp và cả các ngành như y tế, công nghiệp nặng đang được ưu tiên và phát triển mạnh mẽ.

Tại Việt Nam, Nhà nước đang đi vào phát triển dịch vụ, và đầu tư mạnh mẽ vào trí tuệ nhân tạo hay còn gọi là AI. Vì thế, việc đó đầu nó và phát triển nó đang là một xu thế rất hot và rất được ưu chuộng hiện này.

Chính vì vậy thông qua việc học môn trí tuệ nhân tạo (AI) nhóm em đã nghĩ ra một ý tưởng nhỏ đó là áp dụng thuật toán mình đã học để làm ra một Game xếp hình 8 ô số. Phục vụ cho việc chứng minh áp dụng trí tuệ nhân tạo mang lại lợi ích tối ưu về không gian và thời gian cho con người.

**LỜI CẢM ƠN**

Trong thời đại công nghệ 4.0 đang ngày càng phát triển, ứng dụng trí tuệ nhân tạo vào đời sống được xem là một trong những xu thế phát triển mạnh mẽ hiện nay. Việc tự động hóa các hành vi thông minh đang trở thành một cuộc cách mạng trong ngành CNTT.

Vì vậy chúng em đã thực hiện đồ án trí tuệ nhân tạo áp dụng thuật toán “Breadth First Search” và “Best First Search” để làm ra trò chơi ghép tranh 8 ô số, phục vụ cho việc chứng minh áp dụng trí tuệ nhân tạo mang lại lợi ích tối ưu về không gian và thời gian cho con người.

Với sự hướng dẫn, động viên tận tình của cô **Nguyễn Lê Hiền Duyên**, chúng em đã hiểu được cách thức hoạt động của thuật toán và hoàn thành bài báo cáo đồ án này. Do chưa có nhiều kinh nghiệm nghiên cứu, thực hành nên chúng em cũng không tránh khỏi những thiếu sót. Chúng em rất mong nhận được sự thông cảm và góp ý của thầy để đề tài của chúng em được hoàn thiện hơn.

Chúng em xin chân thành cảm ơn!

1. GIỚI THIỆU VÀ MÔ TẢ BÀI TOÁN THỰC TẾ ĐƯỢC GIẢI QUYẾT
   1. Tên đề tài:

Game ghép tranh 8 ô số.

* 1. Lý do chọn đề tài:

Mô tả sự khác nhau, thời gian tìm kiếm, độ tối ưu giữa hai thuật toán “Breadth First Search” và “Best First Search” thông qua trò chơi Ghép Tranh 8 ô số.

* 1. Mô tả bài toán thực tế được giải quyết:
* Vị trí của các hình trong trò chơi sẽ nằm ngẫu nhiên trộn lẫn trong 9 ô, trong đó có 1 ô đen để người dùng dịch chuyển đi từng bước. Mỗi lần di chuyển người dùng chỉ có thể đi 1 bước theo chiều qua trái, qua phải, đi lên hoặc đi xuống để ghép thành 1 hình hoàn chỉnh theo hình mẫu đã cho theo đó. Người dùng không được đi chéo.
* Trong quá trình chơi có thể có trường hợp người dùng không thể đi đến trạng thái hoàn chỉnh của hình. Vì vậy chúng ta áp dụng trí tuệ nhân tạo vào trong trò chơi này, và cụ thể là chúng em đã áp dụng 2 thuật toán Breadth-First-Search và Best-First-Search trong game này để đưa người chơi đi đến trạng thái hoàn chỉnh (giải ra được đường đi đến trạng thái hoàn chỉnh).
* Vậy bài toán thực tế ở đây là tìm ra đường đi đến trạng thái hoàn thành của game ghép tranh 8 ô số, cũng như đưa một trạng thái ngẫu nhiên chưa hoàn chỉnh của bức tranh về một bức tranh hoàn chỉnh áp dụng 2 thuật toán trong trí tuệ nhân tạo.

2. CÁC CHI TIẾT CỦA PHƯƠNG PHÁP DÙNG ĐỂ GIẢI QUYẾT BÀI TOÁN
   1. Thuật toán Breadth First Search:
      * Đây là thuật toán tìm đường đi từ đỉnh xuất phát đến đỉnh kết thúc bằng các duyệt theo chiều rộng.
      * Đây là thuật toán nằm trong nhóm thuật toán tìm kiếm mù, thuật toán không quan tâm đến trọng số trên đường đi mà chỉ duyệt theo những đỉnh kề liên tiếp nó.
      * Xuất phát tử một đỉnh và đi tới các đỉnh kề nó, tiếp tục cho đến khi không còn đỉnh nào để đi.
      * Trong quá trình đi đến đỉnh kề, tiến hành lưu lại đỉnh kề để khi đi ngược lại từ đỉnh kết thúc đến đỉnh xuất phát ta có được đường đi ngắn nhất.
      * **Mô tả thuật toán:** Cách đỉnh đã được xét thì không thể xét lại lần 2 nữa. Cơ chế lưu lại đỉnh kề sẽ là lưu các đỉnh kề thành 1 danh sách và lấy từ từ danh sách các đỉnh kề ra để xét, khi mà một đỉnh kề của một đỉnh đang xét được thêm vào danh sách thì nó sẽ được thêm vào cuối của danh sách hay còn gọi là cơ chế Queue(hàng đợi). Cơ chết này có nghĩa khi thêm một phần tử thì phần tử đó sẽ được thêm ở cuối danh sách hàng đợi, còn lấy phần tử ra thì sẽ lấy ra ở đầu danh sách hàng đợi. Đó chính là điểm nổi bật để phân biệt thuật toán này với các thuật toán khác.
      * ***Ưu điểm***

* Dễ cài đặt.
* Nếu số đỉnh là hữu hạn, thuật toán chắc chắn tìm ra kết quả.
  + - ***Khuyết điểm***
* Mang tính chất vét cạn, không nên áp dụng nếu duyệt số đỉnh quá lớn.
* Mang tính chất mù quáng, duyệt tất cả đỉnh, không chú ý đến thông tin trong các đỉnh để duyệt hiệu quả, dẫn đến duyệt qua các đỉnh không cần thiết
* Chiếm thời gian và không gian bộ nhớ khi số đình duyệt nhiều.
  + - **Chi tiết chạy từng bước của thuật toán BFS:**

Breadth\_First\_Search (BFS)

**Begin** Bước 1: **Khởi tạo** danh sách **L** chỉ chứa trạng thái ban đầu

Bước 2: **Loop do**

2.1 **If** L rỗng **then**

{Thông báo tìm kiếm thất bại; Stop};

2.2 Loại trạng thái **u** ở đầu danh sách L;

2.3 **If u** là trạng thái kết thúc **then**

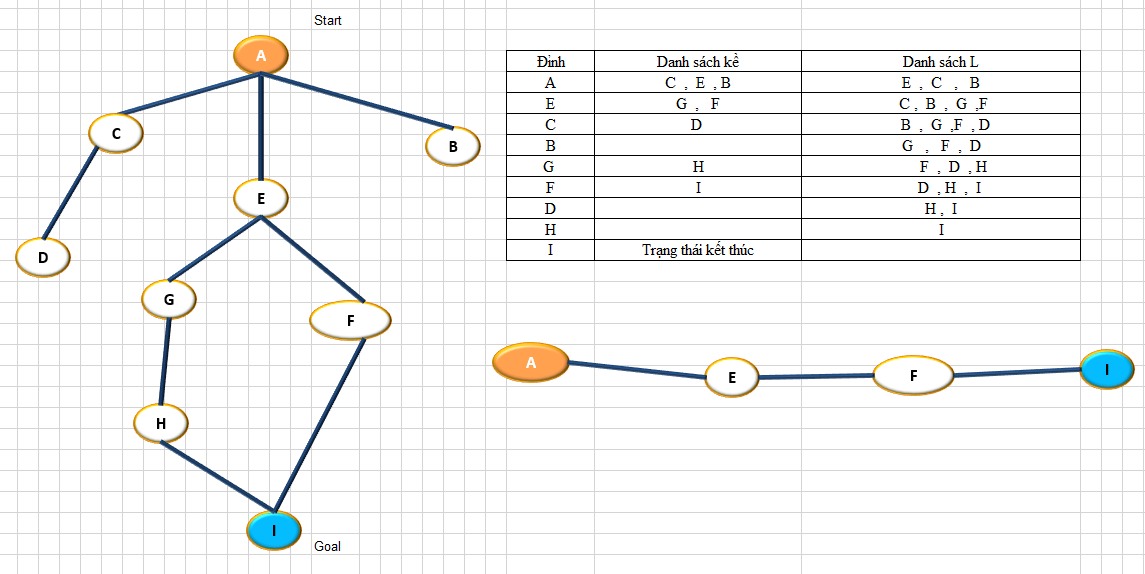
{Thông báo tìm kiếm thành công; Stop};

2.4 **for** mỗi trạng thái **v** kề với trạng thái **u do**

{Đặt v vào cuối danh sách L;

father(v) = u};

Ví dụ 1:



Trong ví dụ trên, các trạng thái trong danh sách kề sẽ được thêm lần lượt vào cuối trong danh sách L(Queue hàng đợi), mỗi lần đỉnh duyệt sẽ lấy ra trạng thái đầu tiên trong danh sách L này để duyệt tiếp theo. Cứ tiếp tục như vậy cho đến khi đi đến đỉnh kết thúc cần tìm hoặc cho đến khi Danh sách L (Queue hàng đợi ) đã rỗng.

* 1. Thuật toán Best First Search
     + Tìm kiếm theo bề rộng được hướng dẫn bởi hàm đánh giá hay còn gọi là hàm Heuristic. Nằm trong nhóm thuật toán tìm kiếm kinh nghiệm.
     + Hàm đánh giá được tính theo 2 phương thức: đếm số ô sai vị trí so với trạng thái người dùng mong muốn và tính theo khoảng cách Mahattan. Trong trò chơi này hàm đánh giá (Heuristic) được tính theo đếm số ô sai vị trí so với trạng thái đích mong muốn.
     + Mô tả thuật toán: Cũng tương tự như thuật toán Breadth First Search. Nhưng Best First Search khác ở chổ: Đỉnh được chon để phát triển là đỉnh tốt nhất được xác định bởi hàm đánh giá (đỉnh có giá trị hàm đánh giá nhỏ nhất). Có nghĩa là mỗi lần thêm vào hàng đợi danh sách các trạng thái sẽ được sắp xếp theo thứ tự tăng dần dựa theo hàm đánh giá, hàm đánh giá càng thấp thì tương đương số ô bị sai vị trí là ít nhất. Từ đó ta chọn được trạng thái tốt nhất để đi tiếp.
     + Ưu điểm của thuật toán này duyệt qua ít đỉnh hơn vì có hàm đánh giá ở trong nên khi duyệt việc sắp xếp hàm đánh giá sẽ giúp tìm tới đỉnh cuối cùng nhanh hơn và ít tốn thời gian hơn.
     + ***Ưu điểm***
* Khá dễ cài đặt.
* Nếu số đỉnh là hữu hạn, thuật toán chắc chắn tìm ra kết quả.
* Có xét thêm hàm Heuristic nên duyệt ít trạng thái hơn Breadth First Search => nhanh hơn, ít tốn không gian hơn.
  + - ***Khuyết điểm***
* Chiếm thời gian và không gian bộ nhớ nếu số đỉnh duyệt quá lớn.
  + - **Chi tiết chạy từng bước của thuật toán** Best First Search**:**

**Begin** Bước 1: **Khởi tạo** danh sách **L** chỉ chứa trạng thái ban đầu

Bước 2: **Loop do**

2.1 **If** L rỗng **then**

{Thông báo tìm kiếm thất bại; Stop};

2.2 Loại trạng thái **u** ở đầu danh sách L;

2.3 **If u** là trạng thái kết thúc **then**

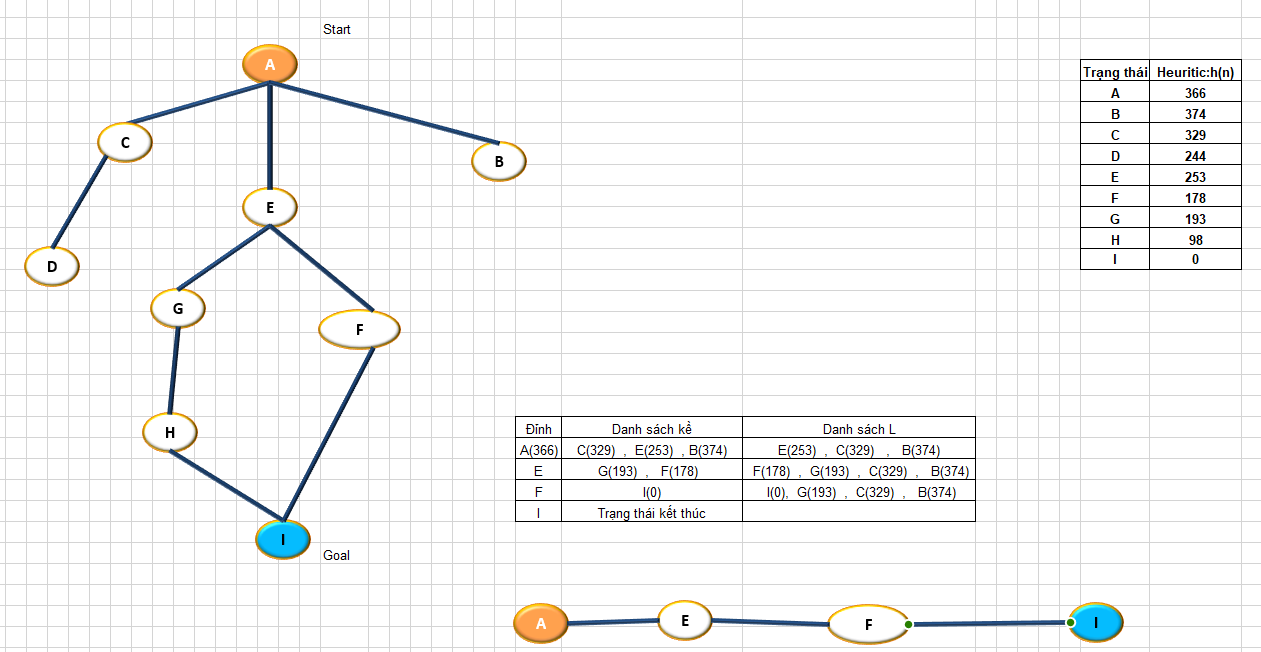
{Thông báo tìm kiếm thành công; Stop};

2.4 **for** mỗi trạng thái **v** kề với trạng thái **u do**

Xen v vào danh sách L sao cho L được sắp theo thứ tự tăng dần của hàm đánh giá;

End;

**VÍ DỤ 2:**

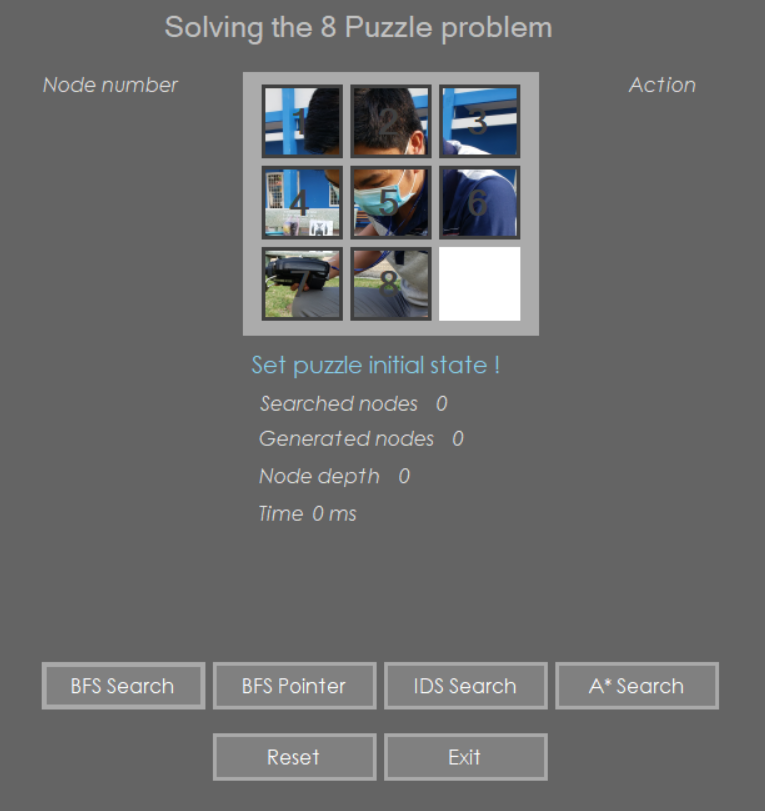


Trong ví dụ 2 này ta thấy đường đi cuối cùng vẫn là A->E->F->I như là ví dụ 1. Nhưng số trạng thái mà thuật toán phải duyệt chỉ có 4 đỉnh là đã tìm ra được đường đi.

Do trong danh sách L (Queue hàng đợi) lúc này đã đưa được trạng thái tốt nhất về đầu danh sách để lấy ra duyệt đi tiếp.

1. QUÁ TRÌNH THỰC HIỆN
   1. Thiết kế giao diện

-Thiết kế 1 panel chứa 9 Button tương đương với 9 mảnh hình và các Button chức năng cho game phía dưới. Đặt id Button theo thứ tự btn1,btn2…btn9. Cắt hình thành 8 mảnh ghép đánh số theo thứ tự 1-8 và một mảnh trắng tên kk



* 1. Lập trình chức năng di chuyển các hình cho người chơi

static public void MoveBTN(Button btn1, Button btn2,Button btn3,Color clr)

{

if (btn2.Text == "")

{

btn2.Text = btn1.Text;

btn2.BackColor = btn1.BackColor;

btn2.ForeColor = btn1.ForeColor;

btn2.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\" + btn1.Text.ToString() + ".JPG");

btn2.FlatAppearance.BorderSize = 3;

btn1.Text = "";

btn1.BackColor = clr;

btn1.ForeColor = clr;

btn1.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\kk.png");

btn1.FlatAppearance.BorderSize = 0;

btn1.FlatAppearance.MouseDownBackColor = clr;

btn1.FlatAppearance.MouseOverBackColor = clr;

}

else if (btn3.Text == "")

{

btn3.Text = btn1.Text;

btn3.BackColor = btn1.BackColor;

btn3.ForeColor = btn1.ForeColor;

btn3.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\" + btn1.Text.ToString() + ".JPG");

btn3.FlatAppearance.BorderSize = 3;

btn1.Text = "";

btn1.BackColor = clr;

btn1.ForeColor = clr;

btn1.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\kk.png");

btn1.FlatAppearance.BorderSize = 0;

btn1.FlatAppearance.MouseDownBackColor = clr;

btn1.FlatAppearance.MouseOverBackColor = clr;

}

}

static public void MoveBTN(Button btn1, Button btn2, Button btn3,Button btn4, Color clr)

{

if (btn2.Text == "")

{

btn2.Text = btn1.Text;

btn2.BackColor = btn1.BackColor;

btn2.ForeColor = btn1.ForeColor;

btn2.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\" + btn1.Text.ToString() + ".JPG");

btn2.FlatAppearance.BorderSize = 3;

btn1.Text = "";

btn1.BackColor = clr;

btn1.ForeColor = clr;

btn1.FlatAppearance.BorderSize = 0;

btn1.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\kk.png");

btn1.FlatAppearance.MouseDownBackColor = clr;

btn1.FlatAppearance.MouseOverBackColor = clr;

}

else if (btn3.Text == "")

{

btn3.Text = btn1.Text;

btn3.BackColor = btn1.BackColor;

btn3.ForeColor = btn1.ForeColor;

btn3.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\" + btn1.Text.ToString() + ".JPG");

btn3.FlatAppearance.BorderSize = 3;

btn1.Text = "";

btn1.BackColor = clr;

btn1.ForeColor = clr;

btn1.FlatAppearance.BorderSize = 0;

btn1.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\kk.png");

btn1.FlatAppearance.MouseDownBackColor = clr;

btn1.FlatAppearance.MouseOverBackColor = clr;

}

else if (btn4.Text == "")

{

btn4.Text = btn1.Text;

btn4.BackColor = btn1.BackColor;

btn4.ForeColor = btn1.ForeColor;

btn4.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\" + btn1.Text.ToString() + ".JPG");

btn4.FlatAppearance.BorderSize = 3;

btn1.Text = "";

btn1.BackColor = clr;

btn1.ForeColor = clr;

btn1.FlatAppearance.BorderSize = 0;

btn1.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\kk.png");

btn1.FlatAppearance.MouseDownBackColor = clr;

btn1.FlatAppearance.MouseOverBackColor = clr;

}

}

static public void MoveBTN(Button btn1, Button btn2, Button btn3, Button btn4,Button btn5, Color clr)

{

if (btn2.Text == "")

{

btn2.Text = btn1.Text;

btn2.BackColor = btn1.BackColor;

btn2.ForeColor = btn1.ForeColor;

btn2.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\" + btn1.Text.ToString() + ".JPG");

btn2.BackgroundImageLayout = ImageLayout.Stretch;

btn2.FlatAppearance.BorderSize = 3;

btn1.Text = "";

btn1.BackColor = clr;

btn1.ForeColor = clr;

btn1.FlatAppearance.BorderSize = 0;

btn1.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\kk.png");

btn1.BackgroundImageLayout = ImageLayout.Stretch;

btn1.FlatAppearance.MouseDownBackColor = clr;

btn1.FlatAppearance.MouseOverBackColor = clr;

}

else if (btn3.Text == "")

{

btn3.Text = btn1.Text;

btn3.BackColor = btn1.BackColor;

btn3.ForeColor = btn1.ForeColor;

btn3.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\" + btn1.Text.ToString() + ".JPG");

btn3.BackgroundImageLayout = ImageLayout.Stretch;

btn3.FlatAppearance.BorderSize = 3;

btn1.Text = "";

btn1.BackColor = clr;

btn1.ForeColor = clr;

btn1.FlatAppearance.BorderSize = 0;

btn1.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\kk.png");

btn1.BackgroundImageLayout = ImageLayout.Stretch;

btn1.FlatAppearance.MouseDownBackColor = clr;

btn1.FlatAppearance.MouseOverBackColor = clr;

}

else if (btn4.Text == "")

{

btn4.Text = btn1.Text;

btn4.BackColor = btn1.BackColor;

btn4.ForeColor = btn1.ForeColor;

btn4.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\" + btn1.Text.ToString() + ".JPG");

btn4.BackgroundImageLayout = ImageLayout.Stretch;

btn4.FlatAppearance.BorderSize = 3;

btn1.Text = "";

btn1.BackColor = clr;

btn1.ForeColor = clr;

btn1.FlatAppearance.BorderSize = 0;

btn1.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\kk.png");

btn1.BackgroundImageLayout = ImageLayout.Stretch;

btn1.FlatAppearance.MouseDownBackColor = clr;

btn1.FlatAppearance.MouseOverBackColor = clr;

}

else if (btn5.Text == "")

{

btn5.Text = btn1.Text;

btn5.BackColor = btn1.BackColor;

btn5.ForeColor = btn1.ForeColor;

btn5.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\" + btn1.Text.ToString() + ".JPG");

btn5.BackgroundImageLayout = ImageLayout.Stretch;

btn5.FlatAppearance.BorderSize = 3;

btn1.Text = "";

btn1.BackColor = clr;

btn1.ForeColor = clr;

btn1.FlatAppearance.BorderSize = 0;

btn1.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\kk.png");

btn1.BackgroundImageLayout = ImageLayout.Stretch;

btn1.FlatAppearance.MouseDownBackColor = clr;

btn1.FlatAppearance.MouseOverBackColor = clr;

}

}

}

Btn thứ nhất là vị trí được chọn. Btn thứ hai và btn thứ ba là vị trí được xét

Nếu btn hai rỗng trao đổi btn1 và btn2

Ngược lại trao đồi btn1 và btn3

* 1. Hàm xuất thông báo và gọi hàm thực thi khi người chơi chọn chức năng sắp xếp BFS

List<int> temp;

label3\_Info.ForeColor = Color.DarkTurquoise;

label3\_Info.Text = "Searching ...";

sw.Start();

label3\_Info.Text = bfs.BfsSearchFunction();

sw.Stop();

lbl\_time\_ans.Text = sw.Elapsed.TotalMilliseconds.ToString() + " ms";

MessageBox.Show(label3\_Info.Text);

sw.Reset();

temp = bfs.PathFinder(bfs.i);

foreach (var item in temp)

{

listBox1.Items.Add(item.ToString());

}

foreach (int item in temp)

{

listBox2.Items.Add(bfs.queueArray.ElementAt(item).ActionStruct);

path.Add(bfs.queueArray.ElementAt(item).ActionStruct);

}

lbl\_Snodes\_ans.Text = bfs.i.ToString();

lbl\_Gnodes\_ans.Text = bfs.queueArray.Count.ToString();

lbl\_depth\_ans.Text = bfs.DepthFinder().ToString();

MachineMove.path = path.ToArray();

path.Clear();

temp.Clear();

label3\_Info.ForeColor = Color.LightPink;

label3\_Info.Text = "Puzzle Solved!";

Thread.Sleep(300);

label3\_Info.Text = "";

Thread.Sleep(300);

label3\_Info.Text = "Puzzle Solved!";

Thread.Sleep(300);

label3\_Info.Text = "";

Thread.Sleep(300);

label3\_Info.Text = "Puzzle Solved!";

label3\_Info.ForeColor = Color.Cyan;

label3\_Info.Text = "Showing Solution .";

Thread.Sleep(300);

label3\_Info.Text = "Showing Solution ..";

Thread.Sleep(300);

label3\_Info.Text = "Showing Solution ...";

MachineMove.Action(button1, button2, button3, button4, button5, button6, button7, button8, button9, panel1.BackColor);

label3\_Info.ForeColor = Color.YellowGreen;

label3\_Info.Text = "Machine Win !";

bfs.queueArray.Clear();

bfs.PathList.Clear();

bfs.i = 0;

bfs.j = 0;

trReset = new Thread(new ThreadStart(ThreadReset));

trReset.Start();

* 1. Cài đặt hàm BFS

public string BfsSearchFunction()

{

while (Goal(queueArray[i].PuzzelStruct) == false)

{

if (queueArray[i].PuzzelStruct == null)

{

if (i >= 1)

{

pz.PuzzelStruct = null;

pz.ActionStruct = null;

for (int i = 0; i < 4; i++)

{

queueArray.Add(pz);

}

}

i++;

}

else

{

addNode(queueArray[i].PuzzelStruct);

i++;

}

}

return "Puzzle Solved!";

}

3.5 Cài đặt hàm kiểm tra mục đích

public bool Goal(Int16[,] puzzleArray)

{

if (queueArray[i].PuzzelStruct == null)

{

return false;

}

else if (puzzleArray[0, 0] == 1 &&

puzzleArray[0, 1] == 2 &&

puzzleArray[0, 2] == 3 &&

puzzleArray[1, 0] == 4 &&

puzzleArray[1, 1] == 5 &&

puzzleArray[1, 2] == 6 &&

puzzleArray[2, 0] == 7 &&

puzzleArray[2, 1] == 8 &&

puzzleArray[2, 2] == 0)

{

return true;

}

return false;

}

Return true nếu mảng đã sắp xếp ngược lại return false

3.6 Hàm thêm node vào mảng puzzle

public void addNode (Int16[,] puzzle)

{

puzzeli puzzle8 = new puzzeli();

string id = SearchArray((Int16[,])puzzle);

Int16 temp;

switch (id)

{

case "00":

{

puzzle8.PuzzelStruct = (Int16[,])(queueArray[i].PuzzelStruct).Clone();

temp = puzzle8.PuzzelStruct[1, 0];

puzzle8.PuzzelStruct[1, 0] = puzzle8.PuzzelStruct[0, 0];

puzzle8.PuzzelStruct[0, 0] = temp;

pz.PuzzelStruct= puzzle8.PuzzelStruct;

pz.ActionStruct= "Down";

queueArray.Add(pz);

pz.PuzzelStruct = null;

pz.ActionStruct = null;

queueArray.Add(pz);

pz.PuzzelStruct = null;

pz.ActionStruct = null;

queueArray.Add(pz);

puzzle8.PuzzelStruct = (Int16[,])queueArray[i].PuzzelStruct.Clone();

temp = puzzle8.PuzzelStruct[0,1];

puzzle8.PuzzelStruct[0,1] = puzzle8.PuzzelStruct[0, 0];

puzzle8.PuzzelStruct[0, 0] = temp;

pz.PuzzelStruct = puzzle8.PuzzelStruct;

pz.ActionStruct = "Right";

queueArray.Add(pz);

break;

}

case "01":

{

puzzle8.PuzzelStruct = (Int16[,])(queueArray[i].PuzzelStruct).Clone();

temp = puzzle8.PuzzelStruct[1,1];

puzzle8.PuzzelStruct[1,1] = puzzle8.PuzzelStruct[0,1];

puzzle8.PuzzelStruct[0,1] = temp;

pz.PuzzelStruct = puzzle8.PuzzelStruct;

pz.ActionStruct = "Down";

queueArray.Add(pz);

pz.PuzzelStruct = null;

pz.ActionStruct = null;

queueArray.Add(pz);

puzzle8.PuzzelStruct = (Int16[,])(queueArray[i].PuzzelStruct).Clone();

temp = puzzle8.PuzzelStruct[0,0];

puzzle8.PuzzelStruct[0,0] = puzzle8.PuzzelStruct[0,1];

puzzle8.PuzzelStruct[0,1] = temp;

pz.PuzzelStruct = puzzle8.PuzzelStruct;

pz.ActionStruct = "Left";

queueArray.Add(pz);

puzzle8.PuzzelStruct = (Int16[,])(queueArray[i].PuzzelStruct).Clone();

temp = puzzle8.PuzzelStruct[0, 2];

puzzle8.PuzzelStruct[0, 2] = puzzle8.PuzzelStruct[0, 1];

puzzle8.PuzzelStruct[0, 1] = temp;

pz.PuzzelStruct = puzzle8.PuzzelStruct;

pz.ActionStruct = "Right";

queueArray.Add(pz);

break;

}

case "02":

{

puzzle8.PuzzelStruct = (Int16[,])(queueArray[i].PuzzelStruct).Clone();

temp = puzzle8.PuzzelStruct[1,2];

puzzle8.PuzzelStruct[1,2] = puzzle8.PuzzelStruct[0,2];

puzzle8.PuzzelStruct[0,2] = temp;

pz.PuzzelStruct = puzzle8.PuzzelStruct;

pz.ActionStruct = "Down";

queueArray.Add(pz);

pz.PuzzelStruct = null;

pz.ActionStruct = null;

queueArray.Add(pz);

puzzle8.PuzzelStruct = (Int16[,])(queueArray[i].PuzzelStruct).Clone();

temp = puzzle8.PuzzelStruct[0,1];

puzzle8.PuzzelStruct[0,1] = puzzle8.PuzzelStruct[0,2];

puzzle8.PuzzelStruct[0,2] = temp;

pz.PuzzelStruct = puzzle8.PuzzelStruct;

pz.ActionStruct = "Left";

queueArray.Add(pz);

pz.PuzzelStruct = null;

pz.ActionStruct = null;

queueArray.Add(pz);

break;

}

case "10":

{

puzzle8.PuzzelStruct = (Int16[,])(queueArray[i].PuzzelStruct).Clone();

temp = puzzle8.PuzzelStruct[2,0];

puzzle8.PuzzelStruct[2,0] = puzzle8.PuzzelStruct[1,0];

puzzle8.PuzzelStruct[1,0] = temp;

pz.PuzzelStruct = puzzle8.PuzzelStruct;

pz.ActionStruct = "Down";

queueArray.Add(pz);

puzzle8.PuzzelStruct = (Int16[,])(queueArray[i].PuzzelStruct).Clone();

temp = puzzle8.PuzzelStruct[0,0];

puzzle8.PuzzelStruct[0,0] = puzzle8.PuzzelStruct[1, 0];

puzzle8.PuzzelStruct[1, 0] = temp;

pz.PuzzelStruct = puzzle8.PuzzelStruct;

pz.ActionStruct = "Up";

queueArray.Add(pz);

pz.PuzzelStruct = null;

pz.ActionStruct = null;

queueArray.Add(pz);

puzzle8.PuzzelStruct = (Int16[,])(queueArray[i].PuzzelStruct).Clone();

temp = puzzle8.PuzzelStruct[1,1];

puzzle8.PuzzelStruct[1,1] = puzzle8.PuzzelStruct[1, 0];

puzzle8.PuzzelStruct[1, 0] = temp;

pz.PuzzelStruct = puzzle8.PuzzelStruct;

pz.ActionStruct = "Right";

queueArray.Add(pz);

break;

}

case "11":

{

puzzle8.PuzzelStruct = (Int16[,])(queueArray[i].PuzzelStruct).Clone();

temp = puzzle8.PuzzelStruct[2,1];

puzzle8.PuzzelStruct[2,1] = puzzle8.PuzzelStruct[1,1];

puzzle8.PuzzelStruct[1, 1] = temp;

pz.PuzzelStruct = puzzle8.PuzzelStruct;

pz.ActionStruct = "Down";

queueArray.Add(pz);

puzzle8.PuzzelStruct = (Int16[,])(queueArray[i].PuzzelStruct).Clone();

temp = puzzle8.PuzzelStruct[0,1];

puzzle8.PuzzelStruct[0,1] = puzzle8.PuzzelStruct[1, 1];

puzzle8.PuzzelStruct[1, 1] = temp;

pz.PuzzelStruct = puzzle8.PuzzelStruct;

pz.ActionStruct = "Up";

queueArray.Add(pz);

puzzle8.PuzzelStruct = (Int16[,])(queueArray[i].PuzzelStruct).Clone();

temp = puzzle8.PuzzelStruct[1, 0];

puzzle8.PuzzelStruct[1, 0] = puzzle8.PuzzelStruct[1, 1];

puzzle8.PuzzelStruct[1, 1] = temp;

pz.PuzzelStruct = puzzle8.PuzzelStruct;

pz.ActionStruct = "Left";

queueArray.Add(pz);

puzzle8.PuzzelStruct = (Int16[,])(queueArray[i].PuzzelStruct).Clone();

temp = puzzle8.PuzzelStruct[1,2];

puzzle8.PuzzelStruct[1,2] = puzzle8.PuzzelStruct[1, 1];

puzzle8.PuzzelStruct[1, 1] = temp;

pz.PuzzelStruct = puzzle8.PuzzelStruct;

pz.ActionStruct = "Right";

queueArray.Add(pz);

break;

}

case "12":

{

puzzle8.PuzzelStruct = (Int16[,])(queueArray[i].PuzzelStruct).Clone();

temp = puzzle8.PuzzelStruct[2,2];

puzzle8.PuzzelStruct[2,2] = puzzle8.PuzzelStruct[1,2];

puzzle8.PuzzelStruct[1, 2] = temp;

pz.PuzzelStruct = puzzle8.PuzzelStruct;

pz.ActionStruct = "Down";

queueArray.Add(pz);

puzzle8.PuzzelStruct = (Int16[,])(queueArray[i].PuzzelStruct).Clone();

temp = puzzle8.PuzzelStruct[0,2];

puzzle8.PuzzelStruct[0,2] = puzzle8.PuzzelStruct[1, 2];

puzzle8.PuzzelStruct[1, 2] = temp;

pz.PuzzelStruct = puzzle8.PuzzelStruct;

pz.ActionStruct = "Up";

queueArray.Add(pz);

pz.PuzzelStruct = null;

pz.ActionStruct = null;

queueArray.Add(pz);

puzzle8.PuzzelStruct = (Int16[,])(queueArray[i].PuzzelStruct).Clone();

temp = puzzle8.PuzzelStruct[1,1];

puzzle8.PuzzelStruct[1,1] = puzzle8.PuzzelStruct[1, 2];

puzzle8.PuzzelStruct[1, 2] = temp;

pz.PuzzelStruct = puzzle8.PuzzelStruct;

pz.ActionStruct = "Left";

queueArray.Add(pz);

break;

}

case "20":

{

pz.PuzzelStruct = null;

pz.ActionStruct = null;

queueArray.Add(pz);

puzzle8.PuzzelStruct = (Int16[,])(queueArray[i].PuzzelStruct).Clone();

temp = puzzle8.PuzzelStruct[1,0];

puzzle8.PuzzelStruct[1, 0] = puzzle8.PuzzelStruct[2,0];

puzzle8.PuzzelStruct[2, 0] = temp;

pz.PuzzelStruct = puzzle8.PuzzelStruct;

pz.ActionStruct = "Up";

queueArray.Add(pz);

pz.PuzzelStruct = null;

pz.ActionStruct = null;

queueArray.Add(pz);

puzzle8.PuzzelStruct = (Int16[,])(queueArray[i].PuzzelStruct).Clone();

temp = puzzle8.PuzzelStruct[2,1];

puzzle8.PuzzelStruct[2,1] = puzzle8.PuzzelStruct[2, 0];

puzzle8.PuzzelStruct[2, 0] = temp;

pz.PuzzelStruct = puzzle8.PuzzelStruct;

pz.ActionStruct = "Right";

queueArray.Add(pz);

break;

}

case "21":

{

pz.PuzzelStruct = null;

pz.ActionStruct = null;

queueArray.Add(pz);

puzzle8.PuzzelStruct = (Int16[,])(queueArray[i].PuzzelStruct).Clone();

temp = puzzle8.PuzzelStruct[1,1];

puzzle8.PuzzelStruct[1,1] = puzzle8.PuzzelStruct[2,1];

puzzle8.PuzzelStruct[2, 1] = temp;

pz.PuzzelStruct = puzzle8.PuzzelStruct;

pz.ActionStruct = "Up";

queueArray.Add(pz);

puzzle8.PuzzelStruct = (Int16[,])(queueArray[i].PuzzelStruct).Clone();

temp = puzzle8.PuzzelStruct[2, 0];

puzzle8.PuzzelStruct[2, 0] = puzzle8.PuzzelStruct[2, 1];

puzzle8.PuzzelStruct[2, 1] = temp;

pz.PuzzelStruct = puzzle8.PuzzelStruct;

pz.ActionStruct = "Left";

queueArray.Add(pz);

puzzle8.PuzzelStruct = (Int16[,])(queueArray[i].PuzzelStruct).Clone();

temp = puzzle8.PuzzelStruct[2,2];

puzzle8.PuzzelStruct[2,2] = puzzle8.PuzzelStruct[2, 1];

puzzle8.PuzzelStruct[2, 1] = temp;

pz.PuzzelStruct = puzzle8.PuzzelStruct;

pz.ActionStruct = "Right";

queueArray.Add(pz);

break;

}

case "22":

{

pz.PuzzelStruct = null;

pz.ActionStruct = null;

queueArray.Add(pz);

puzzle8.PuzzelStruct = (Int16[,])(queueArray[i].PuzzelStruct).Clone();

temp = puzzle8.PuzzelStruct[1,2];

puzzle8.PuzzelStruct[1,2] = puzzle8.PuzzelStruct[2,2];

puzzle8.PuzzelStruct[2, 2] = temp;

pz.PuzzelStruct = puzzle8.PuzzelStruct;

pz.ActionStruct = "Up";

queueArray.Add(pz);

puzzle8.PuzzelStruct = (Int16[,])(queueArray[i].PuzzelStruct).Clone();

temp = puzzle8.PuzzelStruct[2, 1];

puzzle8.PuzzelStruct[2, 1] = puzzle8.PuzzelStruct[2, 2];

puzzle8.PuzzelStruct[2, 2] = temp;

pz.PuzzelStruct = puzzle8.PuzzelStruct;

pz.ActionStruct = "Left";

queueArray.Add(pz);

pz.PuzzelStruct = null;

pz.ActionStruct = null;

queueArray.Add(pz);

break;

}

}

}

3.7 Tìm kiếm vị trí trong mảng

public string SearchArray(Int16[,] puzzleSearch)

{

for (int i = 0; i < 3; i++)

{

for (int j = 0; j < 3; j++)

{

if (puzzleSearch[i, j] == 0)

return i.ToString() + j.ToString();

}

}

return "";

}

3.8 Hàm di chuyển button cho máy

public static void Action(Button btn1, Button btn2, Button btn3, Button btn4, Button btn5, Button btn6, Button btn7, Button btn8, Button btn9, Color clr)

{

string item;

for (int i = path.Length-1; i >= 0; i--)

{

item = path[i];

if (btn1.Text == "")

{

switch (item)

{

case "Down":

{

btn1.Text = btn4.Text;

btn1.BackColor = btn4.BackColor;

btn1.ForeColor = btn4.ForeColor;

btn1.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\" + btn4.Text.ToString() + ".JPG");

btn1.FlatAppearance.BorderSize = 3;

btn4.Text = "";

btn4.BackColor = clr;

btn4.ForeColor = clr;

btn4.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\kk.png");

btn4.FlatAppearance.BorderSize = 0;

Thread.Sleep(200);

break;

}

case "Right":

{

btn1.Text = btn2.Text;

btn1.BackColor = btn2.BackColor;

btn1.ForeColor = btn2.ForeColor;

btn1.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\" + btn2.Text.ToString() + ".JPG");

btn1.FlatAppearance.BorderSize = 3;

btn2.Text = "";

btn2.BackColor = clr;

btn2.ForeColor = clr;

btn2.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\kk.png");

btn2.FlatAppearance.BorderSize = 0;

Thread.Sleep(200);

break;

}

}

}

else if (btn2.Text == "")

{

switch (item)

{

case "Down":

{

btn2.Text = btn5.Text;

btn2.BackColor = btn5.BackColor;

btn2.ForeColor = btn5.ForeColor;

btn2.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\" + btn5.Text.ToString() + ".JPG");

btn2.FlatAppearance.BorderSize = 3;

btn5.Text = "";

btn5.BackColor = clr;

btn5.ForeColor = clr;

btn5.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\kk.png");

btn5.FlatAppearance.BorderSize = 0;

Thread.Sleep(200);

break;

}

case "Left":

{

btn2.Text = btn1.Text;

btn2.BackColor = btn1.BackColor;

btn2.ForeColor = btn1.ForeColor;

btn2.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\" + btn1.Text.ToString() + ".JPG");

btn2.FlatAppearance.BorderSize = 3;

btn1.Text = "";

btn1.BackColor = clr;

btn1.ForeColor = clr;

btn1.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\kk.png");

btn1.FlatAppearance.BorderSize = 0;

Thread.Sleep(200);

break;

}

case "Right":

{

btn2.Text = btn3.Text;

btn2.BackColor = btn3.BackColor;

btn2.ForeColor = btn3.ForeColor;

btn2.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\" + btn3.Text.ToString() + ".JPG");

btn2.FlatAppearance.BorderSize = 3;

btn3.Text = "";

btn3.BackColor = clr;

btn3.ForeColor = clr;

btn3.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\kk.png");

btn3.FlatAppearance.BorderSize = 0;

Thread.Sleep(200);

break;

}

}

}

else if (btn3.Text == "")

{

switch (item)

{

case "Down":

{

btn3.Text = btn6.Text;

btn3.BackColor = btn6.BackColor;

btn3.ForeColor = btn6.ForeColor;

btn3.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\" + btn6.Text.ToString() + ".JPG");

btn3.FlatAppearance.BorderSize = 3;

btn6.Text = "";

btn6.BackColor = clr;

btn6.ForeColor = clr;

btn6.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\kk.png");

btn6.FlatAppearance.BorderSize = 0;

Thread.Sleep(200);

break;

}

case "Left":

{

btn3.Text = btn2.Text;

btn3.BackColor = btn2.BackColor;

btn3.ForeColor = btn2.ForeColor;

btn3.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\" + btn2.Text.ToString() + ".JPG");

btn3.FlatAppearance.BorderSize = 3;

btn2.Text = "";

btn2.BackColor = clr;

btn2.ForeColor = clr;

btn2.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\kk.png");

btn2.FlatAppearance.BorderSize = 0;

Thread.Sleep(200);

break;

}

}

}

else if (btn4.Text == "")

{

switch (item)

{

case "Down":

{

btn4.Text = btn7.Text;

btn4.BackColor = btn7.BackColor;

btn4.ForeColor = btn7.ForeColor;

btn4.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\" + btn7.Text.ToString() + ".JPG");

btn4.FlatAppearance.BorderSize = 3;

btn7.Text = "";

btn7.BackColor = clr;

btn7.ForeColor = clr;

btn7.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\kk.png");

btn7.FlatAppearance.BorderSize = 0;

Thread.Sleep(200);

break;

}

case "Up":

{

btn4.Text = btn1.Text;

btn4.BackColor = btn1.BackColor;

btn4.ForeColor = btn1.ForeColor;

btn4.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\" + btn1.Text.ToString() + ".JPG");

btn4.FlatAppearance.BorderSize = 3;

btn1.Text = "";

btn1.BackColor = clr;

btn1.ForeColor = clr;

btn1.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\kk.png");

btn1.FlatAppearance.BorderSize = 0;

Thread.Sleep(200);

break;

}

case "Right":

{

btn4.Text = btn5.Text;

btn4.BackColor = btn5.BackColor;

btn4.ForeColor = btn5.ForeColor;

btn4.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\" + btn5.Text.ToString() + ".JPG");

btn4.FlatAppearance.BorderSize = 3;

btn5.Text = "";

btn5.BackColor = clr;

btn5.ForeColor = clr;

btn5.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\kk.png");

btn5.FlatAppearance.BorderSize = 0;

Thread.Sleep(200);

break;

}

}

}

else if (btn5.Text == "")

{

switch (item)

{

case "Down":

{

btn5.Text = btn8.Text;

btn5.BackColor = btn8.BackColor;

btn5.ForeColor = btn8.ForeColor;

btn5.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\" + btn8.Text.ToString() + ".JPG");

btn5.FlatAppearance.BorderSize = 3;

btn8.Text = "";

btn8.BackColor = clr;

btn8.ForeColor = clr;

btn8.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\kk.png");

btn8.FlatAppearance.BorderSize = 0;

Thread.Sleep(200);

break;

}

case "Up":

{

btn5.Text = btn2.Text;

btn5.BackColor = btn2.BackColor;

btn5.ForeColor = btn2.ForeColor;

btn5.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\" + btn2.Text.ToString() + ".JPG");

btn5.FlatAppearance.BorderSize = 3;

btn2.Text = "";

btn2.BackColor = clr;

btn2.ForeColor = clr;

btn2.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\kk.png");

btn2.FlatAppearance.BorderSize = 0;

Thread.Sleep(200);

break;

}

case "Left":

{

btn5.Text = btn4.Text;

btn5.BackColor = btn4.BackColor;

btn5.ForeColor = btn4.ForeColor;

btn5.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\" + btn4.Text.ToString() + ".JPG");

btn5.FlatAppearance.BorderSize = 3;

btn4.Text = "";

btn4.BackColor = clr;

btn4.ForeColor = clr;

btn4.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\kk.png");

btn4.FlatAppearance.BorderSize = 0;

Thread.Sleep(200);

break;

}

case "Right":

{

btn5.Text = btn6.Text;

btn5.BackColor = btn6.BackColor;

btn5.ForeColor = btn6.ForeColor;

btn5.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\" + btn6.Text.ToString() + ".JPG");

btn5.FlatAppearance.BorderSize = 3;

btn6.Text = "";

btn6.BackColor = clr;

btn6.ForeColor = clr;

btn6.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\kk.png");

btn6.FlatAppearance.BorderSize = 0;

Thread.Sleep(200);

break;

}

}

}

else if (btn6.Text == "")

{

switch (item)

{

case "Down":

{

btn6.Text = btn9.Text;

btn6.BackColor = btn9.BackColor;

btn6.ForeColor = btn9.ForeColor;

btn6.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\" + btn9.Text.ToString() + ".JPG");

btn6.FlatAppearance.BorderSize = 3;

btn9.Text = "";

btn9.BackColor = clr;

btn9.ForeColor = clr;

btn9.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\kk.png");

btn9.FlatAppearance.BorderSize = 0;

Thread.Sleep(200);

break;

}

case "Up":

{

btn6.Text = btn3.Text;

btn6.BackColor = btn3.BackColor;

btn6.ForeColor = btn3.ForeColor;

btn6.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\" + btn3.Text.ToString() + ".JPG");

btn6.FlatAppearance.BorderSize = 3;

btn3.Text = "";

btn3.BackColor = clr;

btn3.ForeColor = clr;

btn3.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\kk.png");

btn3.FlatAppearance.BorderSize = 0;

Thread.Sleep(200);

break;

}

case "Left":

{

btn6.Text = btn5.Text;

btn6.BackColor = btn5.BackColor;

btn6.ForeColor = btn5.ForeColor;

btn6.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\" + btn5.Text.ToString() + ".JPG");

btn6.FlatAppearance.BorderSize = 3;

btn5.Text = "";

btn5.BackColor = clr;

btn5.ForeColor = clr;

btn5.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\kk.png");

btn5.FlatAppearance.BorderSize = 0;

Thread.Sleep(200);

break;

}

}

}

else if (btn7.Text == "")

{

switch (item)

{

case "Up":

{

btn7.Text = btn4.Text;

btn7.BackColor = btn4.BackColor;

btn7.ForeColor = btn4.ForeColor;

btn7.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\" + btn4.Text.ToString() + ".JPG");

btn7.FlatAppearance.BorderSize = 3;

btn4.Text = "";

btn4.BackColor = clr;

btn4.ForeColor = clr;

btn4.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\kk.png");

btn4.FlatAppearance.BorderSize = 0;

Thread.Sleep(200);

break;

}

case "Right":

{

btn7.Text = btn8.Text;

btn7.BackColor = btn8.BackColor;

btn7.ForeColor = btn8.ForeColor;

btn7.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\" + btn8.Text.ToString() + ".JPG");

btn7.FlatAppearance.BorderSize = 3;

btn8.Text = "";

btn8.BackColor = clr;

btn8.ForeColor = clr;

btn8.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\kk.png");

btn8.FlatAppearance.BorderSize = 0;

Thread.Sleep(200);

break;

}

}

}

else if (btn8.Text == "")

{

switch (item)

{

case "Up":

{

btn8.Text = btn5.Text;

btn8.BackColor = btn5.BackColor;

btn8.ForeColor = btn5.ForeColor;

btn8.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\" + btn5.Text.ToString() + ".JPG");

btn8.FlatAppearance.BorderSize = 3;

btn5.Text = "";

btn5.BackColor = clr;

btn5.ForeColor = clr;

btn5.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\kk.png");

btn5.FlatAppearance.BorderSize = 0;

Thread.Sleep(200);

break;

}

case "Left":

{

btn8.Text = btn7.Text;

btn8.BackColor = btn7.BackColor;

btn8.ForeColor = btn7.ForeColor;

btn8.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\" + btn7.Text.ToString() + ".JPG");

btn8.FlatAppearance.BorderSize = 3;

btn7.Text = "";

btn7.BackColor = clr;

btn7.ForeColor = clr;

btn7.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\kk.png");

btn7.FlatAppearance.BorderSize = 0;

Thread.Sleep(200);

break;

}

case "Right":

{

btn8.Text = btn9.Text;

btn8.BackColor = btn9.BackColor;

btn8.ForeColor = btn9.ForeColor;

btn8.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\" + btn9.Text.ToString() + ".JPG");

btn8.FlatAppearance.BorderSize = 3;

btn9.Text = "";

btn9.BackColor = clr;

btn9.ForeColor = clr;

btn9.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\kk.png");

btn9.FlatAppearance.BorderSize = 0;

Thread.Sleep(200);

break;

}

}

}

else if (btn9.Text == "")

{

switch (item)

{

case "Up":

{

btn9.Text = btn6.Text;

btn9.BackColor = btn6.BackColor;

btn9.ForeColor = btn6.ForeColor;

btn9.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\" + btn6.Text.ToString() + ".JPG");

btn9.FlatAppearance.BorderSize = 3;

btn6.Text = "";

btn6.BackColor = clr;

btn6.ForeColor = clr;

btn6.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\kk.png");

btn6.FlatAppearance.BorderSize = 0;

Thread.Sleep(200);

break;

}

case "Left":

{

btn9.Text = btn8.Text;

btn9.BackColor = btn8.BackColor;

btn9.ForeColor = btn8.ForeColor;

btn9.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\" + btn8.Text.ToString() + ".JPG");

btn9.FlatAppearance.BorderSize = 3;

btn8.Text = "";

btn8.BackColor = clr;

btn8.ForeColor = clr;

btn8.BackgroundImage = Bitmap.FromFile(@"E:\taive\8-Puzzle\8-Puzzle\Resources\kk.png");

btn8.FlatAppearance.BorderSize = 0;

Thread.Sleep(200);

break;

}

}

}

}

Array.Clear(MachineMove.path, 0, MachineMove.path.Length);

}

}

3.9 Sự kiện Reset

void ThreadReset()

{

ResetRunCheck = true;

btn\_Reset.Text = "Reset";

while (true)

{

btn\_Reset.FlatAppearance.BorderColor = Color.Silver;

btn\_Reset.BackColor = Color.DarkGray;

btn\_Reset.ForeColor = Color.DarkGreen;

Thread.Sleep(500);

btn\_Reset.FlatAppearance.BorderColor = Color.DarkGray;

btn\_Reset.BackColor = Color.Gray;

btn\_Reset.ForeColor = btn\_Start.ForeColor;

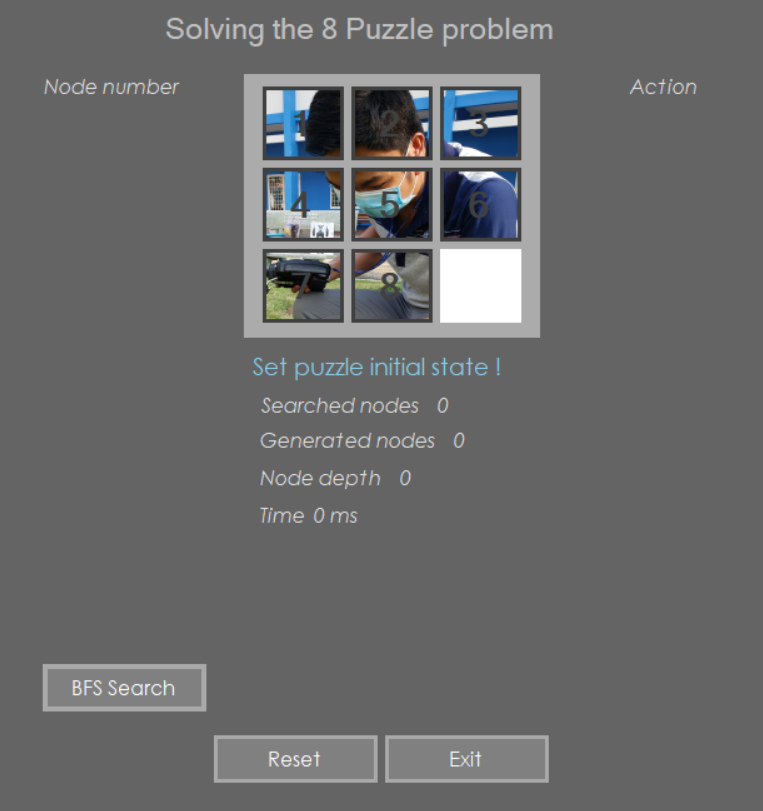
Thread.Sleep(500);

}

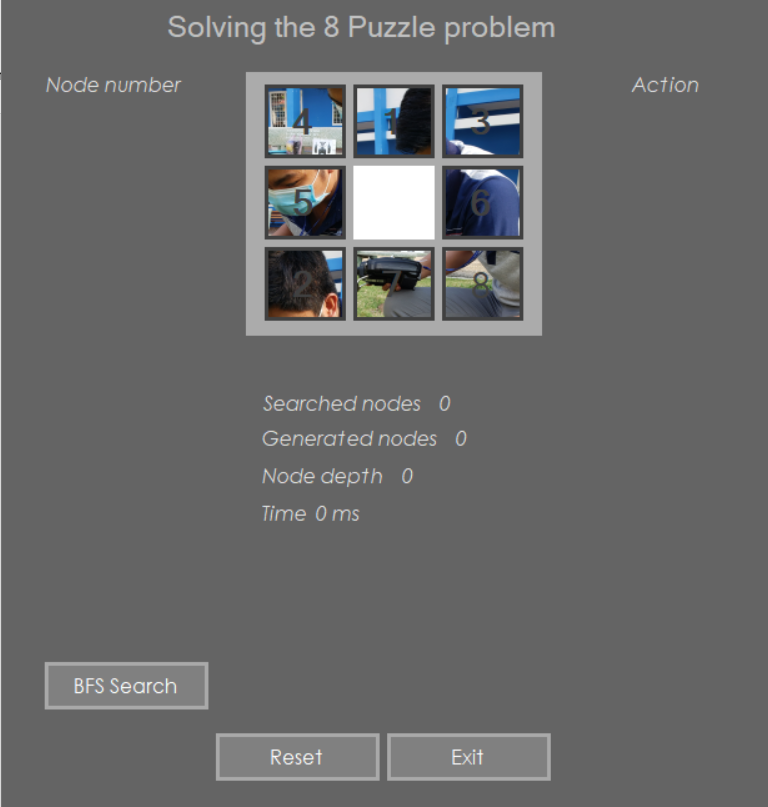
}

4.kết quả

4.1 Màn hình bắt đầu



4.2 Màn hình khi người chơi đã sắp xếp



4.3 Màn hình kết quả khi dùng BFS



1. CÁC VẤN ĐỀ KHÓ KHĂN, TRANH LUẬN VÀ HƯỚNG PHÁT TRIỂN
   1. Vấn đề khó khăn
      * Trường hợp nếu là trạng thái đầu truyền vào là một mảng random thì nếu gặp random ra số vị trí ô sai nhiều thì cả 2 thuật toán Breadth-Frist-Search và Best First Search đều mất quá nhiều thời gian để tìm ra được đường đi.

* Chổ này chúng em khắc phục bằng cách tạo ra 5 trường hợp tương đương 5 testcase có sẵn, 5 trường hợp này đều là các trạng thái mà 2 thuật toán có thể tìm ra được đường đi trong thời gian không quá lâu từ 0.01-> dưới 10 giây. Và cho trạng thái đầu vào chạy random trong 5 testcase này.
  + - Nếu người chơi bấm nút tạm ngừng, sau đó bấm nút chơi lại thì chương trình sẽ bị lỗi.
    - Trò chơi khó có thể mở rộng lên 16 ô, 25 ô..v..v…

1. TÀI LIỆU THAM KHẢO

1.Knuth, Donald E. (1997), [*The Art Of Computer Programming*](http://www-cs-faculty.stanford.edu/~knuth/taocp.html), **1** (ấn bản 3), Boston: Addison-Wesley, [*ISBN*](https://vi.wikipedia.org/wiki/ISBN) [*0-201-89683-4*](https://vi.wikipedia.org/wiki/%C4%90%E1%BA%B7c_bi%E1%BB%87t:Ngu%E1%BB%93n_s%C3%A1ch/0-201-89683-4)

2. <https://notendur.hi.is/mbh6/html/_downloads/Discrete%20Mathematics%20and%20Its%20Applications%20-%20Kenneth%20Rosen%20(2012).pdf>

3. <https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph>

4. <https://www.hackerearth.com/practice/algorithms/graphs/breadth-first-search/tutorial>

5. <https://www.javatpoint.com/breadth-first-search-algorithm>

6.<http://cs.brown.edu/cgc/java3.datastructures.net/handouts/BFS.pdf>

7.https://www.cs.cmu.edu/afs/cs/academic/class/15210-s15/www/lectures/bfs-notes.pdf

* 1. Đề xuất cải tiến game:
     + Đối với trường hợp có quá nhiều ô bị sai thì thuật toán Best First Search hay Breadth First Search vẫn tìm ra đường đi quá lâu, cần áp dụng các thuật toán tối ưu hơn như A\*, Nhánh Cận, AKT để có sự quan sát độ tối ưu, thời gian …v…v… cao hơn.
     + Nên in đường đi là các trạng thái đã duyệt qua lên màn hình giao diện dưới dạng mảng 2 chiều, thay vì là đã in dưới Console.
     + Đề xuất khi bấm nút giải thì label thời gian phải bắt đầu đếm chứ không phải giải xong rồi mới hiển thị thời gian cuối cùng lên lable.
     + Đề xuất khi bấm nút giải tự động các picture box phải tự di chuyển đi đến trạng thái cuối cùng hoàn chỉnh mà không cần người chơi phải click nút “đi tới” để xem.
     + Mở rộng trò chơi lên 16 ô, 25 ô.

**CHƯƠNG 5: NGUỒN THAM KHẢO**

1. <https://www.brainkart.com/article/Best-First-Search--Concept,-Algorithm,-Implementation,-Advantages,-Disadvantages_8881/>
2. <https://en.wikibooks.org/wiki/Artificial_Intelligence/Search/Exhaustive_search/Breadth-first_search>
3. <https://www.geeksforgeeks.org/c-sharp-random-next-method/>
4. <https://docs.microsoft.com/en-us/dotnet/api/system.diagnostics.stopwatch.elapsedmilliseconds?view=netframework-4.8>

<https://stackoverflow.com/questions/6068856/c-sharp-swapping-objects-without-a-placeholder>

KẾT LUẬN

.....

**1. Kết quả đạt được**

......

**2. Hướng phát triển của đề tài**

......