

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
ĐHQG THÀNH PHỐ HỒ CHÍ MINH



ĐỒ ÁN MÔN TRÍ TUỆ NHÂN TẠO

CỜ OTHELLO (REVERSI)



Lớp	:	CS106.J21	
Giáo viên hướng dẫn	:	PHẠM NGUYỄN TRƯỜNG AN	
Sinh viên thực hiện	:	TRẦN MINH ĐỨC	17520368
		NGUYỄN ĐỨC AN	17520213
		VŨ ĐỨC VĨ	17521259
		NGUYỄN THỊ PHƯƠNG HẢO	17520449

MUC LUC

1. Giới thiệu [1]	5
1.1. Cờ othello là gì?	5
1.2. Lịch sử	5
1.3. Cách chơi	5
1.4. Luật cờ quốc tế	8
1.4.1. Thời gian	8
1.4.2. Thực hiện nước đi	9
1.4.3. Những nước đi hợp lệ	9
1.4.4. Những giải đấu quốc tế	10
2. Phân tích thiết kế và xây dựng chương trình cờ othello	13
2.1. Giới hạn	13
2.2. Bàn cờ	13
2.3. Quân cờ	13
2.4. Chiến thuật trò chơi [2]	13
2.5. Các hàm trên lớp Board.cs	14
2.6. Dự đoán nước đi tốt nhất	17
2.6.1. Cây trò chơi	17
2.6.2. Hàm đánh giá	19
2.6.3. Kiểm tra nước đi hợp lệ	24
3. Cài đặt chương trình cờ Othello	25
3.1. Giới thiệu ngôn ngữ C# và winform	25
3.1.1. Cài đặt bàn cờ	27
3.1.2. Applicable_actions của bàn cờ	27
3.1.3. Một nước cờ được di chuyển như thế nào ?	28
3.1.4. Cài đặt thuật giải	28
3.2 Cài đặt giao diện và các chức năng khác cho chương trình	29
3.2.1. Ý tưởng	29
3.2.2. Màn hình chính	30
3.2.3. Chọn độ khó bàn cờ	30
3.2.4. Trò chơi mới (new game)	31
3.2.5. Redo/Undo	31

3.2.6. Luồng chạy giữa máy và người	31
3.2.7. Cờ thế.....	32
3.2.8. Kết thúc trò chơi	33
4. Bugs phát hiện và cách xử lí	33
5. Kết luận	40
6. Tài liệu tham khảo	41

Lời mở đầu

Những năm gần đây, cách mạng công nghiệp 4.0 là từ khóa xuất hiện ở khá nhiều nơi hiện nay, từ tin tức, báo đài cho đến con đường truyền miệng người ta hay nói với nhau hằng ngày. Cuộc cách mạng công nghiệp 4.0 không chỉ là sự phát triển công nghệ thuần túy trong lĩnh vực công nghệ thông tin và truyền thông mà đây sẽ là làn sóng của các giải pháp đột phá về công nghệ trong nhiều lĩnh vực như công nghệ sinh học, công nghệ nano, năng lượng tái tạo, công nghệ vật liệu, tính toán lượng tử và đặc biệt là trí tuệ nhân tạo. Ngày nay, việc nghiên cứu Trí tuệ nhân tạo và đưa nó vào các ứng dụng thực tế đang ngày càng nhiều, và ngày càng chứng tỏ được thế mạnh của mình trong các công việc đòi hỏi khả năng suy nghĩ và tính toán giống như con người. Trong đó, vấn đề Tìm kiếm có đối thủ đang được áp dụng rất rộng rãi trong các trò chơi đối kháng, tất nhiên, tuân theo những tiêu chuẩn nhất định. Bản đồ án này được xây dựng, cũng nằm một trong số đó. Áp dụng lý thuyết Trí tuệ nhân tạo, kết hợp với các hàm đánh giá, từ đó chúng em xây dựng một chương trình cờ Othello có thể hỗ trợ con người trong mục đích giải trí. Chi tiết về trò chơi Othello sẽ được chúng em trình bày trong các chương sau.

Với điều kiện thời gian cũng như kinh nghiệm còn hạn chế của một sinh viên, đồ án này không thể tránh được những thiếu sót. Nhóm chúng em rất mong nhận được sự chỉ bảo, đóng góp ý kiến của thầy để các thành viên có điều kiện bổ sung, nâng cao ý thức của mình và có thể hoàn chỉnh hơn trong những đề tài sau này.

Xin chân thành cảm ơn!

TP. Hồ Chí Minh, tháng 7 năm 2019

BẢNG PHÂN CHIA CÔNG VIỆC

STT	Chủ đề	Chi tiết công việc	Người thực hiện
1	Tìm hiểu công cụ	Ngôn ngữ C# Lập trình winform	Cả nhóm
2	Cơ sở lý thuyết	Tìm hiểu các giải thuật và phương pháp liên quan tới AI	Cả nhóm
3	Giao diện	Thiết kế giao diện	Vĩ , An , Đức
4	Engine	Xây dựng bàn cờ, các hàm cơ bản tạo nước đi	Vĩ, Hào
5	AI	Alpha-beta pruning	Vĩ ,Hào
6	Fix Bugs	Tìm lỗi và sửa lỗi	Vĩ ,Hào
7	Tính năng	Chế độ chơi Undo, redo Time Cờ thế	An, Đức Vĩ, Hào Vĩ, Đức Hào, An
8	Viết báo cáo		Cả nhóm

BẢNG PHÂN TRĂM KHỐI LƯỢNG CÔNG VIỆC CỦA CÁC THÀNH VIÊN

STT	Họ Tên	Phần trăm đóng góp (%)
1	Vũ Đức Vĩ	100
2	Nguyễn Thị Phương Hào	90
3	Nguyễn Đức An	80
4	Trần Minh Đức	80

1. Giới thiệu [1]

1.1. Cờ othello là gì?

Cờ Othello hay còn gọi là Reversi là một trò chơi trên bàn cờ và là môn thể thao trí tuệ dành cho hai người chơi. Bàn cờ được chia lưới ô vuông 8x8 còn những quân cờ có hình dạng giống đồng xu có hai mặt màu nhạt và sẫm (có thể là màu trắng hoặc đen).

1.2. Lịch sử

Nguồn gốc về trò chơi này được biết đến với hai giả thuyết khác nhau. Vào thế kỷ 19, có một người đã phát minh ra trò chơi này và một nhà xuất bản chuyên về các loại trò chơi nổi tiếng của Đức Ravenburger bắt đầu tạo ra vào năm 1898 như là một trong những títt đầu tiên.

Những quy tắc chơi của trò chơi hiện tại được thế giới chấp nhận bây giờ có nguồn gốc từ Nhật Bản và trò chơi đã được gọi là *Othello* vào những năm của thập kỷ 70.

Mattle đã tạo ra các thiết bị có tên là *Othello*. Công ty Anjar đã được cấp thương hiệu có đăng ký *Othello* từ Tsukuda Original.

Goro, người đã viết cuốn sách "*Làm thế nào để trở thành người thắng cuộc trong Othello*" đã phổ biến trò chơi ở Nhật vào năm 1975.

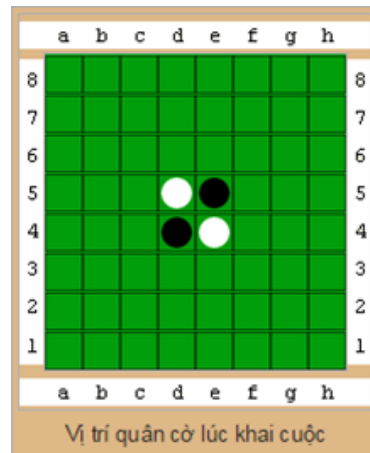
Trò chơi này được lấy tên từ vở kịch *Othello, the Moor of Vinice* của William Shakespeare.

1.3. Cách chơi

Mỗi mặt của quân cờ đại diện cho một bên chơi. Ta có thể gọi cờ hai bên là đen và trắng, nhưng cũng có thể gọi là sấp và ngửa, bởi vì mỗi quân cờ có 2 mặt riêng biệt.

Trước kia, cờ Othello không quy định vị trí đặt quân cờ đầu tiên. Sau đó, nó đã chấp nhận luật chơi mới với điều khoản là phải có 4 điểm đặt đầu tiên vào vị trí 4

hình vuông ở trung tâm bàn cờ, hai quân sẫm và hai quân nhạt. Quân màu sẫm được đi đầu tiên.

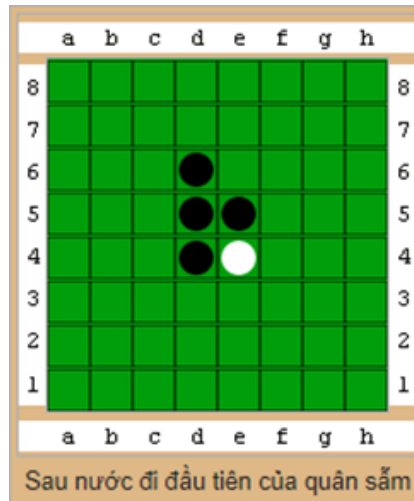


Hình 1.1

Quân màu sẫm cần phải được đặt ở vị trí tồn tại ít nhất một hàng ngang hoặc dọc, hoặc chéo giữa quân mới và quân cũ và ở giữa hai quân này có một hay nhiều quân nhạt.

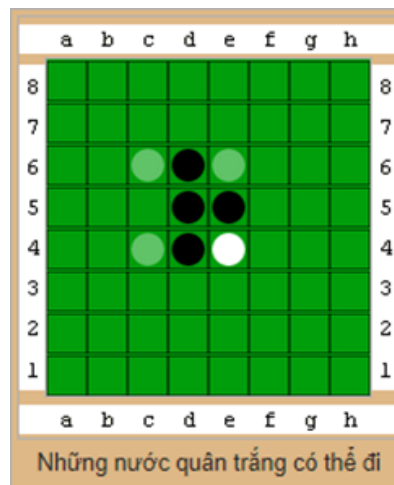
Sau khi đặt một quân, quân sẫm sẽ lật tất cả những quân nhạt nằm trên đường giống giữa quân sẫm mới được đi và quân sẫm cũ. Những quân sáng màu đó bây giờ trở thành màu sẫm và quân sẫm có thể sử dụng chúng trong lượt đi tiếp theo, trừ khi quân nhạt lại lật chúng lại trong một nước đi nào đó.

Nếu quân sẫm quyết định đi ở vị trí d6 (theo hình dưới), một quân nhạt sẽ bị lật mặt và bàn cờ bây giờ có dạng như hình dưới đây.



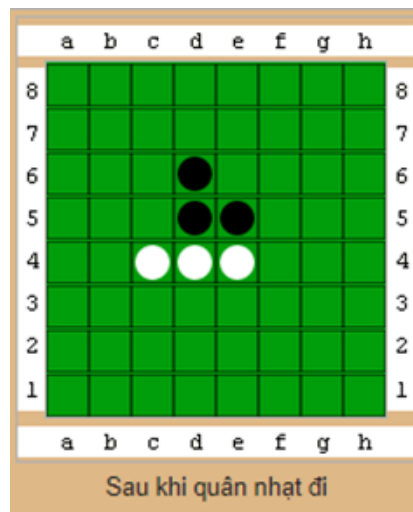
Hình 1.2

Bây giờ đến lượt quân nhạt đi cũng nước tương tự như vậy để tìm cơ hội lật mặt quân sẫm. Các khả năng có thể như sau:



Hình 1.3

Quân nhật đi vào c4 và lật được một quân sâm:



Hình 1.4

Các hình 1.1, 1.2, 1.3, 1.4 được tham khảo từ nguồn:

https://vi.wikipedia.org/wiki/C%E1%BB%9D_Othello#C%C3%A1ch_ch%C6%A1i

Người chơi thay phiên nhau lần lượt đi quân. Nếu một bên không đi được tiếp thì sẽ tiếp tục đến lượt người kia cho đến khi cả hai bên đều không đi được nước nào nữa. Điều này xảy ra khi các ô trên bàn cờ đã kín hết quân hay khi một bên chơi không còn quân nào trên bàn cờ. Người chơi có nhiều quân trên bàn cờ hơn là người thắng cuộc.

1.4. Luật cờ quốc tế

1.4.1. Thời gian

Người chơi có một giới hạn về thời gian để hoàn thành tất cả nước đi của mình trong mỗi ván cờ (Theo luật thi đấu quốc tế năm 2017, các vòng đầu và những vòng playoff, người chơi có 30 phút cho mỗi ván đấu của mình. Vòng bán kết sẽ là 35 phút và chung kết là 40 phút).

Nếu hết giờ, người chơi sẽ bị xử thua cuộc bất chấp mọi trạng thái của bàn cờ ở thời điểm đó.

1.4.2. Thực hiện nước đi

Người chơi không thể thực hiện nước đi trong thời gian của đối thủ (trước khi đối thủ bấm đồng hồ). Người chơi có quyền nhắc đối thủ bấm đồng hồ trong trường hợp đối thủ anh ta quên.

Người chơi phải thực hiện nước đi nếu chạm vào quân cờ hoặc bàn cờ. Trong trường hợp nước đi đó không hợp lệ, anh ta phải chọn một nước hợp lệ gần nhất, cạnh vị trí chạm vào bàn cờ, anh ta bắt buộc phải đi nước đó thay vì một nước đi khác, bất chấp việc đối thủ cho phép đi lại hay không.

Mọi nước đi đều phải được thực hiện bằng một tay, bao gồm cả việc lật những quân cờ chiếm được hoặc bấm đồng hồ.

1.4.3. Những nước đi hợp lệ

Nếu xảy ra tranh chấp về trạng thái bàn cờ, chỉ nước đi gần nhất mới được xem xét. Khi một người chơi xác nhận một nước đi, điều đó đồng nghĩa với việc anh ta đã thừa nhận trạng thái bàn cờ ở thời điểm đó. Không ai, kể cả trọng tài có quyền sửa đổi hay xem xét trạng thái bàn cờ trước đó.

Người chơi cần di chuyển và lật quân cờ sao cho các vị trí quân cờ có thể được nhìn thấy một cách rõ ràng.

Nếu trọng tài xác nhận nước đi của người chơi không rõ ràng, anh ta cần thực hiện mọi hành động để sửa nó (bao gồm cả việc dừng đồng hồ và chơi qua bảng điểm).

1.4.4. Những giải đấu quốc tế

Năm	Địa điểm	Nhà vô địch	Nước	Đội	Runner-Up	Nước
<u>1977</u>	<u>Tokyo</u>	Hiroshi Inoue	Nhật	N/A	Thomas Heiberg	Norway
<u>1977</u>	<u>Monte Carlo*</u>	Sylvain Perez	Pháp	N/A	Michel Rengot Blanchard	Pháp
<u>1978</u>	<u>New York</u>	Hidenori Maruoka	Nhật	N/A	<u>Carol Jacobs</u>	Mỹ
<u>1979</u>	<u>Roma</u>	Hiroshi Inoue	Nhật	N/A	Jonathan Cerf	Mỹ
<u>1980</u>	<u>Luân Đôn</u>	Jonathan Cerf	Mỹ	N/A	Takuya Mimura	Nhật
<u>1981</u>	<u>Brussels</u>	Hidenori Maruoka	Nhật	N/A	Brian Rose	Mỹ
<u>1982</u>	<u>Stockholm</u>	Kunihiko Tanida	Nhật	N/A	David Shaman	Mỹ
<u>1983</u>	<u>Paris</u>	Ken'Ichi Ishii	Nhật	N/A	<u>Imre Leader</u>	Anh
<u>1984</u>	<u>Melbourne</u>	Paul Ralle	Pháp	N/A	Ryoichi Taniguchi	Nhật
<u>1985</u>	<u>Athens</u>	Masaki Takizawa	Nhật	N/A	Paolo Ghirardato	Ý

<u>1986</u>	<u>Tokyo</u>	Hideshi Tamenori	Nhật	N/A	Paul Ralle	Pháp
<u>1987</u>	<u>Milan</u>	Ken'Ichi Ishii	Nhật	Mỹ	Paul Ralle	Pháp
<u>1988</u>	<u>Paris</u>	Hideshi Tamenori	Nhật	Anh	Graham Brightwell	Anh
<u>1989</u>	<u>Warsaw</u>	Hideshi Tamenori	Nhật	Anh	Graham Brightwell	Anh
<u>1990</u>	<u>Stockholm</u>	Hideshi Tamenori	Nhật	Pháp	Didier Piau	Pháp
<u>1991</u>	<u>New York</u>	Shigeru Kaneda	Nhật	Mỹ	Paul Ralle	Pháp
<u>1992</u>	<u>Barcelona</u>	Marc Tastet	Pháp	Anh	David Shaman	Anh
<u>1993</u>	<u>Luân Đôn</u>	David Shaman	Mỹ	Mỹ	Emmanuel Caspard	Pháp
<u>1994</u>	<u>Paris</u>	Masaki Takizawa	Nhật	Pháp	Karsten Feldborg	Đan Mạch
<u>1995</u>	<u>Melbourne</u>	Hideshi Tamenori	Nhật	Mỹ	David Shaman	Mỹ
<u>1996</u>	<u>Tokyo</u>	Takeshi Murakami	Nhật	Anh	Stephane Nicolet	Pháp

<u>1997</u>	<u>Athens</u>	Makoto Suekuni	Nhật	Anh	Graham Brightwell	Anh
<u>1998</u>	<u>Barcelona</u>	Takeshi Murakami	Nhật	Pháp	Emmanuel Caspard	Pháp
<u>1999</u>	<u>Milan</u>	David Shaman	Hà Lan	Nhật	Tetsuya Nakajima	Nhật
<u>2000</u>	<u>Copenhagen</u>	Takeshi Murakami	Nhật	USA	Brian Rose	Mỹ
<u>2001</u>	<u>New York</u>	Brian Rose	Mỹ	Mỹ	Raphael Schreiber	Mỹ
<u>2002</u>	<u>Amsterdam</u>	David Shaman	Hà Lan	Mỹ	Ben Seeley	Mỹ
<u>2003</u>	<u>Stockholm</u>	Ben Seeley	Mỹ	Nhật	Makoto Suekuni	Nhật
<u>2004</u>	London	Ben Seeley	Mỹ	Mỹ	Makoto Suekuni	Nhật
<u>2005</u>	<u>Reykjavik</u>	Hideshi Tamenori	Nhật	Nhật	Kwangwook Lee	Nam Triều Tiên
<u>2006</u>	<u>Mito</u>	Hideshi Tamenori	Nhật	Nhật	Makoto Suekuni	Singapore
<u>2007</u>	<u>Athens</u>	Kenta Tominaga	Nhật	Nhật	Stephane Nicolet	Pháp

2. Phân tích thiết kế và xây dựng chương trình cờ othello

2.1. Giới hạn

Do thời gian và kinh nghiệm còn hạn chế, trò chơi chỉ giới hạn người chơi mặc định đi quân đen, không có chức năng chọn quân.
Sử dụng thuật toán Minimax

2.2. Bàn cờ

Bàn cờ trong trò chơi được lưu dưới dạng mảng 2 chiều kiểu int. Kích thước mảng là 8 x 8.

2.3. Quân cờ

Trò chơi cờ Othello chỉ có 2 màu quân là trắng và đen. Quân cờ được lưu thành các biến kiểu int, với quy ước: quân trắng là -1, đen là 1. Nếu tại một ô trên bàn cờ chưa có quân cờ, giá trị của nó bằng 0.

```
public const int BLACK = 1;  
public const int WHITE = -1;  
public const int EMPTY = 0;
```

2.4. Chiến thuật trò chơi [2]

Các chiến thuật cơ bản của trò chơi bao gồm:

Chiếm góc và lan dần ra biên bàn cờ.

Do đó, ta phải xây dựng ma trận giá trị bàn cờ dựa vào các góc và các biên bàn cờ sao cho:

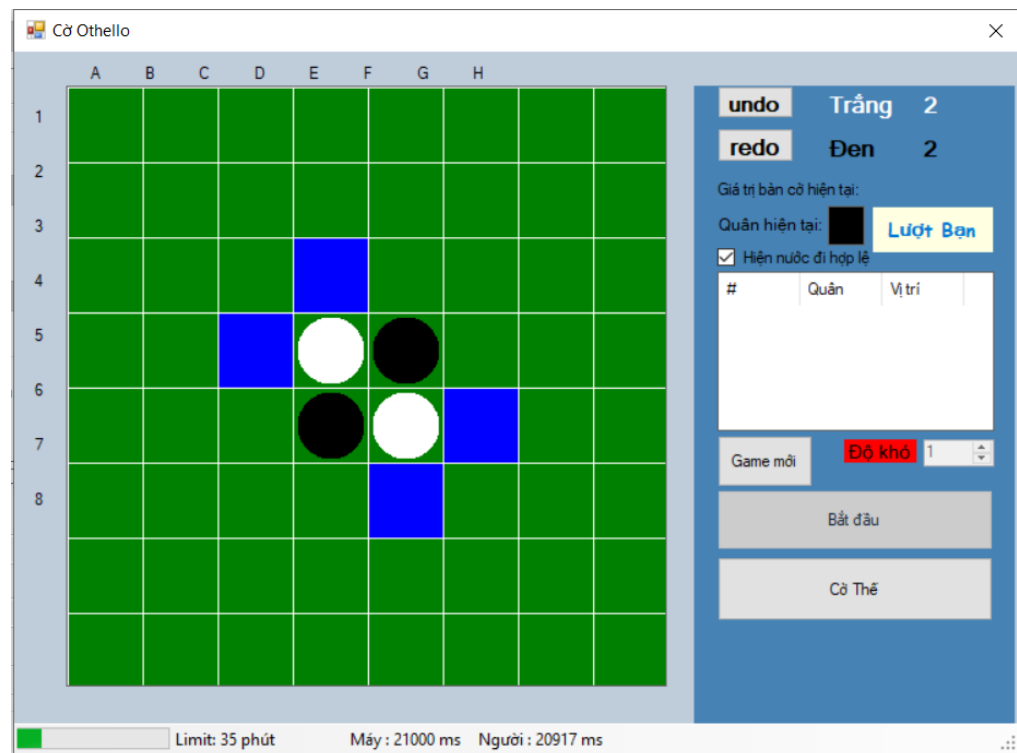
nếu quân trắng đang có lợi thế, giá trị > 0 và ngược lại, quân đen có lợi thế, giá trị < 0 .

2.5. Các hàm trên lớp Board.cs

Ta thấy việc thay đổi trạng thái bàn cờ tương đương với việc thay đổi các thành phần BLACK hoặc WHITE. Điều đó đồng nghĩa ta chỉ cần xem xét các thao tác tương ứng liên quan tới các số BLACK hoặc WHITE.

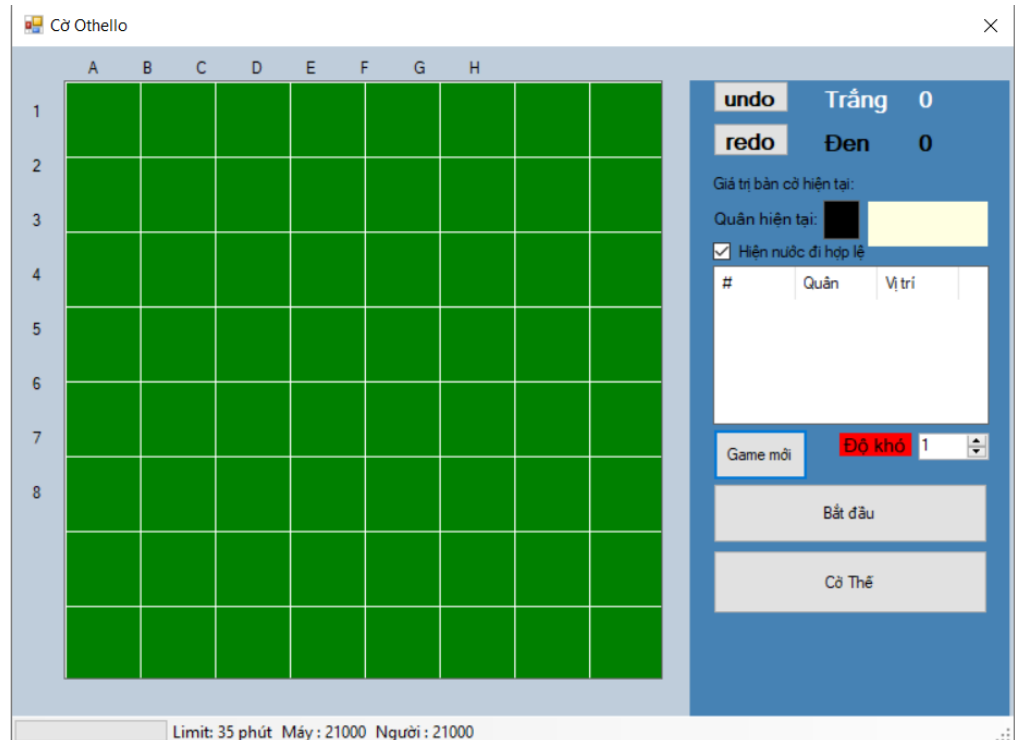
1.Board(true) -> InitMat();

Khởi tạo bàn cờ với 4 quân cờ ban đầu ở giữa bàn cờ .



2. Board(false) -> ResetMat

Reset bàn cờ về trạng thái chưa có quân cờ nào trên bàn cờ (khi chưa bắt đầu ván đấu)



3. Copy()

Trả về 1 bàn cờ đã copy (dùng để tính toán, thao tác mà ko làm thay đổi dữ liệu trên bàn cờ thực hiện tại) bằng cách lưu bàn cờ đã copy vào 1 biến Board .

```
Board copyboard = board.Copy();
```

Copy(Board b) dùng khi ta muốn bàn cờ hiện tại chuyển sang bàn cờ Board b , thể hiện ở chức năng undo , redo .

```
Vd : Board p = Save_Board[0];
```

```
board.Copy(p);
```

4. Draw(Panel p, bool IsDrawHelp, int step)

Vẽ vào bàn cờ và các quân cờ cùng đồ họa cần thiết vào panel .

5. DrawEnableSteps(int step, Graphics graphics)

Vẽ các nước đi gợi ý vào bàn cờ nếu như IsDrawHelp = true trong hàm vẽ bàn cờ Draw .

6. GetValue()

Là hàm heuristic , trả về giá trị của bàn cờ , giá trị bàn cờ càng cao càng lợi thế cho quân trắng và giá trị càng nhỏ càng có lợi thế cho quân đen , vậy nên trong hàm DoBestStep của máy ta để biến maximizingPlayer là true nhằm tìm nước đi tốt nhất cho quân trắng (là khi giá trị minimax lớn nhất trong những giá trị nhỏ nhất của quân đen vì khi này giá trị minimax ở quân đen là lợi thế nhất có thể cho quân đen rồi), giá trị minimax trả về cho quân đen cần chọn càng lớn càng tốt , vì giá trị bàn cờ càng lớn thì quân đen càng bất lợi .

7. Move(int x, int y, int p, bool IsAdd)

Kiểm tra xem 1 quân cờ phe p có di chuyển được vào board[x,y] , nếu giá trị trả về > 0 thì ta có thể di chuyển 1 quân cờ phe p vào vị trí board[x,y] , nếu IsAdd=true thì ta thay đổi bàn cờ theo bước di chuyển đã thực hiện ,

nếu IsAdd=false thì trả về giá trị đã tính .

8. GetEnableSteps(int color)

Tính các nước có thể đi của phe color , hay nói cách khác là hàm applicable_actions .

9. GetBestStep

Là hàm tính minimax .

2.6. Dự đoán nước đi tốt nhất

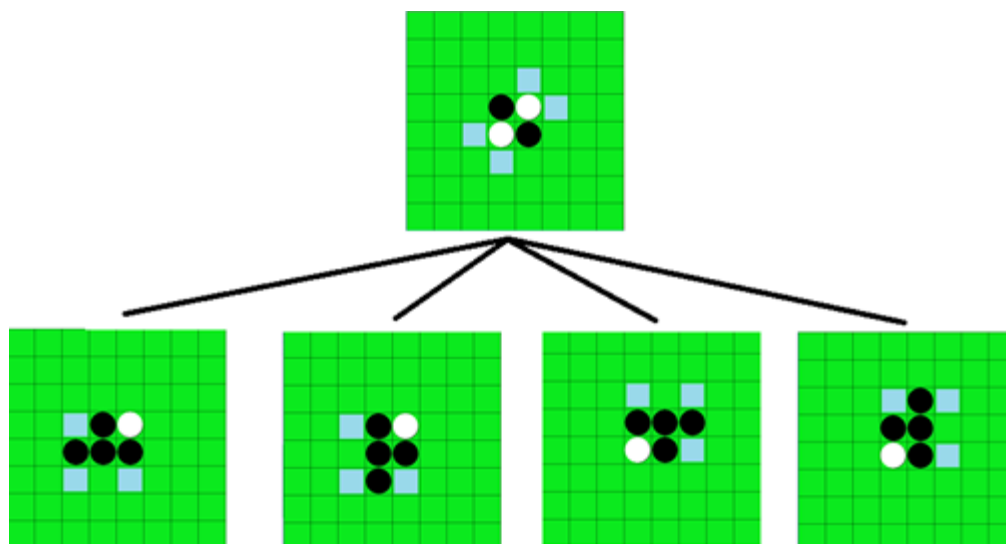
2.6.1. Cây trò chơi

Các trạng thái bàn cờ khác nhau có thể biểu diễn thành một cây tìm kiếm. Để tìm được nước đi tốt nhất ta tiến hành duyệt trên cây này. Các nút tương ứng là các trạng thái của bàn cờ, các nhánh nối giữa các nút tượng trưng cho việc chuyển trạng thái thông qua một thao tác hay một nước đi nào đó.

Xây dựng một cây với trò chơi Othello:

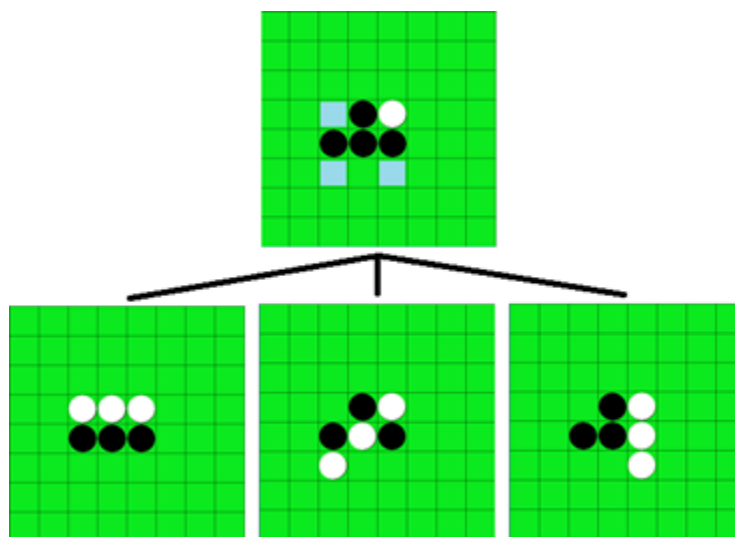
- Gốc của cây trò chơi ứng với trạng thái ban đầu của bàn cờ (trạng thái có 2 quân đen và 2 quân trắng ở giữa).
- Các tầng của cây ứng với lượt đi của quân trắng hoặc quân đen. Nếu tầng nào đó là tầng của quân trắng thì tầng kế tiếp hoặc trước đó (nếu tồn tại) chỉ có thể là tầng ứng với lượt đi của quân đen.
- Tầng MIN tương ứng với nước đã đi của quân đen .
- Tầng MAX tương ứng với nước đã đi của quân trắng.
- Các nút lá tương ứng với trạng thái kết thúc hoặc đến giới hạn của biến depth của trò chơi.

Ta có một ví dụ tường minh về một cây trò chơi Othello 2 tầng. Với gốc là trạng thái ban đầu và quân đen là quân đi trước.



Hình 2.3

Tất nhiên cây trò chơi này vẫn chưa kết thúc, chẳng hạn xét trạng thái ở nút bên trái ngoài cùng ta có thể tạo thêm 3 trạng thái tương ứng (nước đi quân trắng):



Hình 3.4

2.6.2. Hàm đánh giá

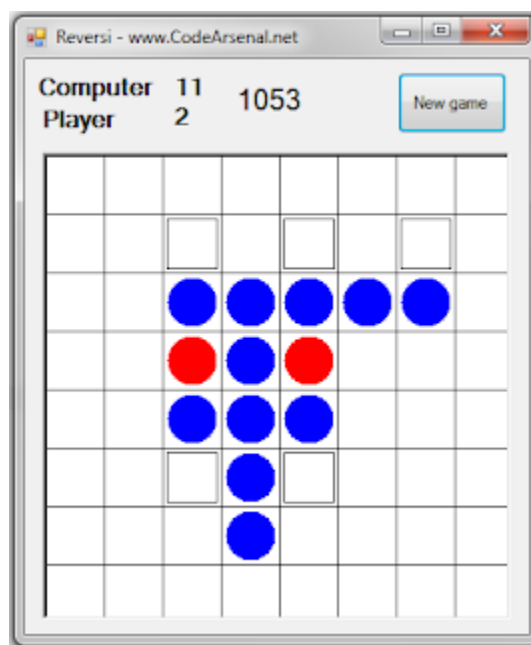
Để tối ưu nước đi cho một quân trong một trạng thái bàn cờ, ta phải duyệt hết qua tất cả đường đi có thể của nó. Cụ thể đối với trường hợp như *Hình 3.3*, ta phải duyệt đủ 4 trường hợp để tìm được nước đi tối ưu cho quân đen .

Từ đây ta cần có một hàm để ứng với mỗi trạng thái ta thu được một giá trị cụ thể có thể dùng so sánh với các trạng thái khác trong cùng một tầng. Hàm này gọi là hàm đánh giá. Để xây dựng được một hàm như thế ta cần xem xét đến các chiến thuật của trò chơi Othello, điểm ta cần quan tâm:

Vị trí các quân cờ: Trên bàn cờ ứng với mỗi vị trí đều có một ưu điểm hoặc nhược điểm riêng biệt, do đó việc phân tích dựa trên các vị trí này là một điểm quan trọng trong việc xây dựng một hàm đánh giá.

2.6.2.a. Giới thiệu về Heuristic sử dụng cho bài toán .

- Hàm heuristic được chúng em tham khảo trên mạng , trong 1 chương trình chơi cờ othello khác .



<http://www.codearsenal.net/2012/06/reversi-game-in-csharp-winform.html#.XSGop6ITpPY>

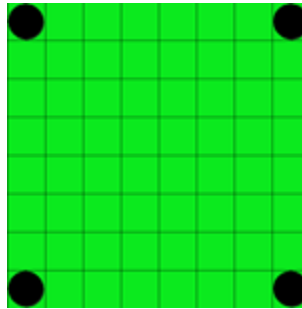
- Hàm heuristic tính toán dựa trên chiến lược chiếm góc và chiếm biên , tức là chiếm được góc và chiếm được càng nhiều phần biên thì có nhiều cơ hội chiến thắng phe địch vì góc và biên là những phần cố định , không thể bị ăn ngược lại . Trong đó chiếm góc có phần quan trọng hơn chiếm biên vì các quân cờ ở biên có thể bị ăn ngược lại còn ở góc thì là cố định , quân cờ ở góc có thể dùng làm bàn đạp để chiếm biên .
- Hàm heuristic trả về theo cơ chế : nếu quân trắng (máy) càng chiếm lợi thế thì giá trị bàn cờ trả về càng có giá trị dương , nếu quân đen (người) càng chiếm lợi thế thì giá trị bàn cờ càng âm , nếu bằng 0 thì đang hòa cờ. Nhưng 4 thông số của bàn cờ lại ngược lại vì về sau các thông số này đều nhân với số âm và được cộng lại , cuối cùng phù hợp với cơ chế trả về của hàm heuristic .
- Cơ chế trả về của hàm heuristic là như vậy để phù hợp với hoạt động của hàm minimax về sau .

2.6.2.b. Vị trí quân cờ

- Góc: Đây là vị trí không thể thay đổi cho đến cuối bàn cờ, do đó việc chiếm góc là một trong những chiến thuật cơ bản nhất của người chơi cờ Othello, đây được xác định là vị trí có điểm cao nhất trong một trạng thái cụ thể.

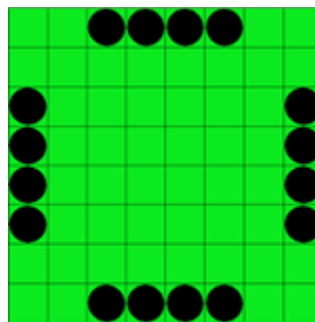
- Chiếm góc: là một lợi thế không thể bỏ qua trên bàn cờ. Vì các quân ở góc không bị lật quân bởi nước đi của đối phương. Bạn có thể sử dụng nó để tạo hàng và chiếm giữ nhóm quân trên bàn cờ.

-



```
corner_value = board[0, 0] + board[0, 7] + board[7, 7] + board[7, 0];
```

- Biên: Vị trí không dễ dàng chiếm được, đây cũng là nơi có điểm số cao (sau góc).
- Chiếm biên: là các hàng ngoài cùng của bàn cờ. Quân cờ đặt ở vị trí biên rất khó bị lật. Các quân này rất hữu dụng cho các nước đi sau này để tăng số quân.



```
for (int i = 0; i < 8; i++)
    edge_value += (board[i, 0] + board[i, 7] + board[0, i] +
board[7, i]);
```

- Hiệu số quân: $\text{difference_pieces} = \text{WhiteCount} - \text{BlackCount}$
quân càng nhiều thì có lợi cho đội mình ,hai chiến lược trên là quan trọng để tính toán nước đi nhưng kết quả lại phụ thuộc vào số lượng

quân trên bàn cờ ở đây nếu quân trắng nhiều hơn thì máy sẽ có lợi hơn .

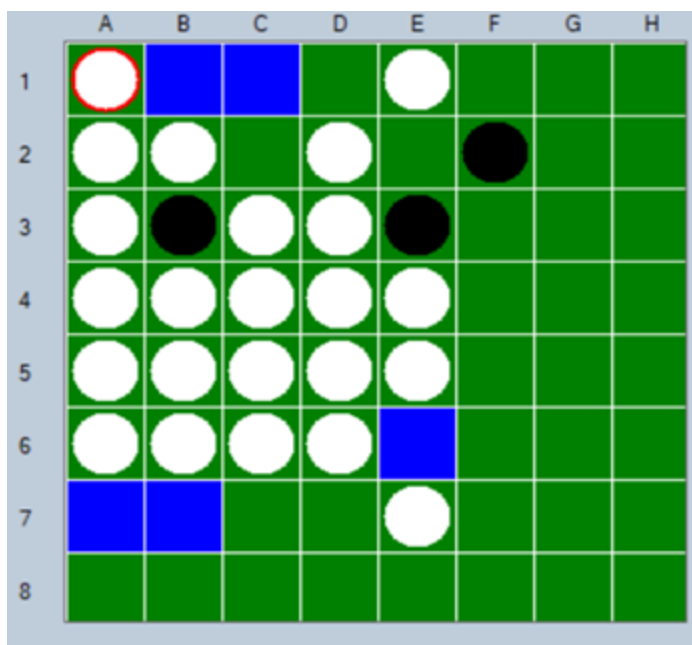
nếu thông số này càng lớn tức là máy càng nhiều quân , khi này để đảm bảo tiêu chí quân trắng càng chiếm lợi thế thì giá trị bàn cờ càng lớn ta sẽ sử dụng lệnh $((100 + \text{WhiteCount} + \text{BlackCount}) * \text{differentce_pieces})$

- same_color :

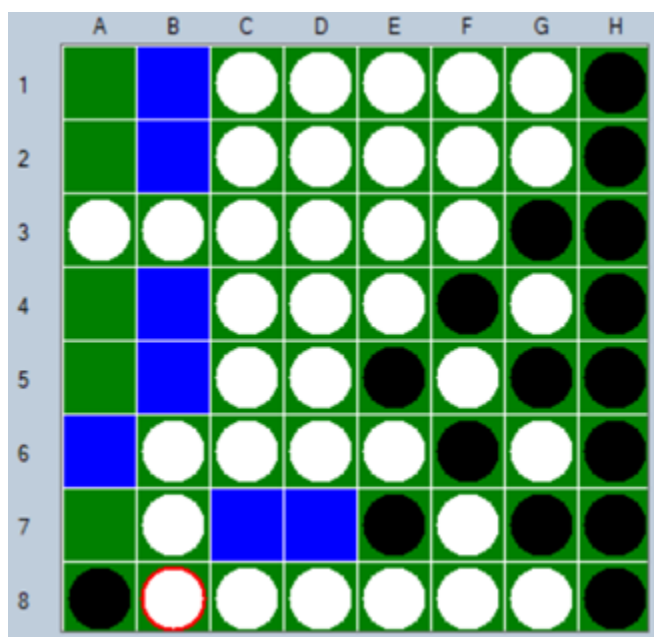
Tính toán tổng các quân cờ ở biên không thể bị ăn ngược lại . Các quân cờ ở biên này có 1 đặc điểm là phải liên tiếp nhau và liên kề với quân cờ cùng màu ở góc . Đây là yếu tố quan trọng nhất trong chiến lược chiếm góc và chiếm biên , vì chiếm được góc không có nghĩa là chiếm được biên và chiếm nhiều phần biên nhưng không chiếm được góc thì các quân cờ biên này hoàn toàn có thể bị ăn ngược lại 1 khi phe địch chiếm được 1 trong 2 góc ở biên đó . Để hoàn toàn chiếm được góc và biên thì cách tốt nhất là chiếm được góc và các ô biên lân cận góc đó để từ từ chiếm được hoàn toàn góc và biên , vì khi này các quân cờ góc và biên này là “bất khả xâm phạm” và được dùng làm vũ khí đặc lực để uy hiếp đối phương và lật ngược thế cờ điều này biểu thị càng nhiều quân cùng màu nằm sát góc thì rất khó để lật ngược thế cờ vì đã chiếm được góc thì không có cách nào ăn lại các quân này ,

càng nhiều quân như thế này càng khiến cho phe địch bị lật kèo vì khi này góc và biên có nguy cơ bị chiếm nhiều hơn và các quân không tiếp giáp biên của phe bạn càng dễ bị ăn ngược lại

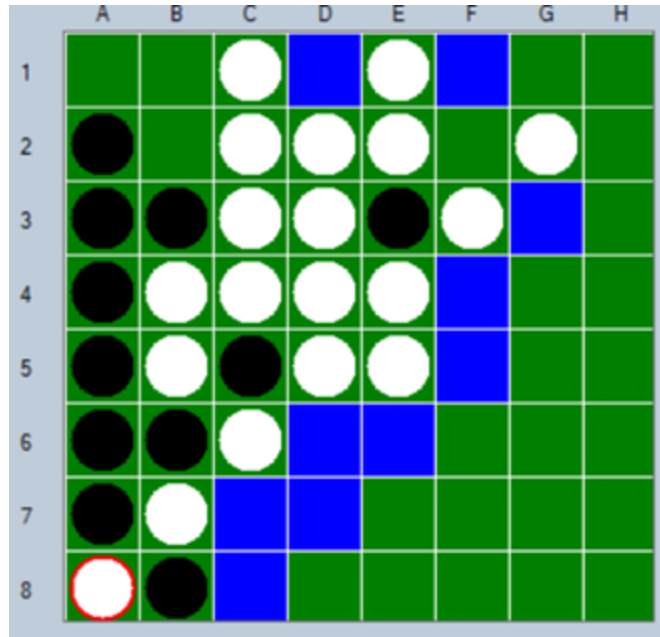
thông số này khác với edge_value vì phụ thuộc vào quân nào đang chiếm ở góc nữa , vì rõ ràng nếu biên đang nhiều quân đen thì chiếm được góc cũng có thể lật ngược thế cờ càng nhiều quân cùng màu với góc ở 2 biên thì khả năng thắng càng cao , và thế cờ có thể bị lật ngược cho dù quân địch có nhiều quân trên bàn cờ đi chăng nữa và những quân này có 1 đặc tính là không thể bị ăn ngược lại , gây sức ép cho đối phương vì đối phương có thể bị ăn ngược bất cứ lúc nào



Ví dụ tiêu biểu thể hiện giá trị thông số `same_color` , khi này cột A đã bị quân trắng chiếm được góc và liên tiếp chiếm được biên và các quân này là “bất khả xâm phạm”



Ví dụ tiêu biểu cho việc chiếm được góc nhưng không chiếm được biên ở hàng 8



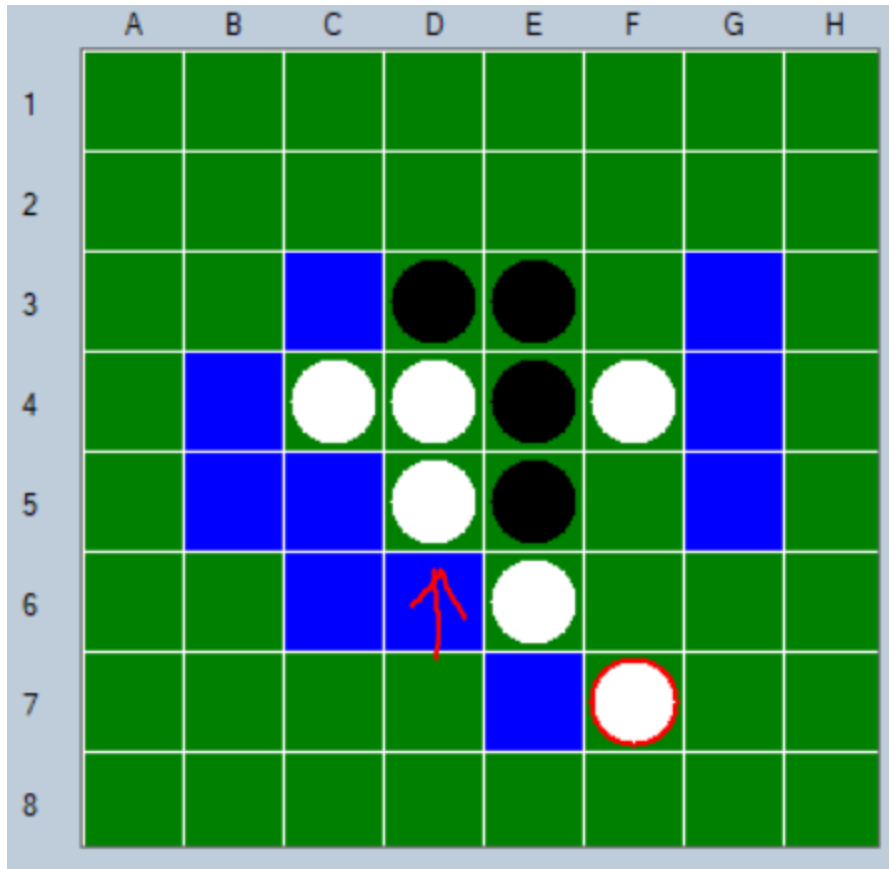
Ví dụ tiêu biểu cho việc chiếm được nhiều biên cũng không chắc là an toàn vì hoàn toàn có thể bị ăn ngược lại chỉ với 1 nước đi của quân trắng vào ô A1

2.6.2.c. Giá trị hàm đánh giá

$$\begin{aligned} \text{boardvalue} = & ((100 + \text{WhiteCount} + \text{BlackCount}) * \text{difference_pieces}) \\ & + (-200 * \text{corner_value}) \\ & + (-150 * \text{edge_value}) \\ & + (-250 * \text{same_color}); \end{aligned}$$

2.6.3. Kiểm tra nước đi hợp lệ

Nước đi được coi là hợp lệ nếu trên ô cờ đó không có quân cờ và có ít nhất một trong tám hướng kiểm tra tại ô đó thỏa mãn luật di chuyển của cờ othello . Các hướng để kiểm tra có chung 1 quy luật : ứng với hướng đó , nếu ô kế tiếp ô thực hiện nước cờ cần kiểm tra là ô trống thì hướng đó không thỏa mãn , ta đi xét hướng khác . Hoặc nếu các ô kế tiếp là quân cờ khác màu quân mình và bị chặn bởi 1 quân cờ phe mình thì đó là hướng hợp lệ .



Ví dụ kiểm tra hướng UpCheck tại ô D6 là thỏa mãn

3. Cài đặt chương trình cờ Othello

3.1. Giới thiệu ngôn ngữ C# và winform

C# (hay C sharp) là một ngôn ngữ lập trình đơn giản, được phát triển bởi đội ngũ kỹ sư của Microsoft vào năm 2000, trong đó người dẫn đầu là Anders Hejlsberg và Scott Wiltamuth.

C# là ngôn ngữ lập trình hiện đại, hướng đối tượng và nó được xây dựng trên nền tảng của hai ngôn ngữ mạnh nhất là C++ và Java.

C# được thiết kế cho Common Language Infrastructure (CLI), mà gồm Executable Code và Runtime Environment, cho phép chúng ta sử dụng các ngôn ngữ high-level đa dạng trên các nền tảng và cấu trúc máy tính khác nhau.

C# với sự hỗ trợ mạnh mẽ của .NET Framework giúp cho việc tạo một ứng dụng Windows Forms hay WPF (Windows Presentation Foundation), . . . trở nên rất dễ dàng.

Kiến trúc ứng dụng Windows Forms

- Windows Forms là cách cơ bản để cung cấp các thành phần giao diện (GUI components) cho môi trường .NET Framework
- Windows Forms được xây dựng trên thư viện Windows API
- Windows Forms cơ bản bao gồm
 - Một Form là khung dùng hiển thị thông tin đến người dùng
 - Các Control được đặt trong form và được lập trình để đáp ứng sự kiện

Ứng dụng Windows Forms

- C# đơn giản
- Loại bỏ các vấn đề của C++, Java (như: macro, template...)
- C# là ngôn ngữ hiện đại
- C# là ngôn ngữ lập trình hướng đối tượng
- Hỗ trợ tính bao đóng, kế thừa và tính đa hình
- C# mạnh mẽ và linh hoạt
- Sử dụng phát triển nhiều loại ứng dụng (xử lý văn bản, hình ảnh, bảng tính phục vụ cho quản lý....)

3.1.1. Cài đặt bàn cờ

```
public const int WIDTH = 40; // chiều rộng 1 ô

public const int BLACK = 1; // quân đen(người)

public const int WHITE = -1; // quân trắng(máy)

public const int EMPTY = 0; // ô trống

private int whitecount = 0; // số quân trắng được khởi tạo ban đầu với giá
trị 0

private int blackcount = 0; // số quân đen được khởi tạo ban đầu với giá trị
0

private int[,] board; // ma trận bàn cờ, được khởi tạo với kích thước
8x8 với tất cả phần tử =0 (trống)

private int x_pre = -1; // vị trí vừa đi (bôi đỏ)

private int y_pre = -1;

// 4 quân cờ đầu tiên khi tạo 1 ván mới

board[3, 3] = WHITE;

board[4, 3] = BLACK;

board[3, 4] = BLACK;

board[4, 4] = WHITE;
```

- Khi di chuyển 1 nước cờ , bàn cờ sẽ duyệt hết tất cả các biên trong board để lấy giá trị của mỗi biên , căn cứ vào giá trị đó để vẽ gì vào bàn cờ .

3.1.2. Applicable_actions của bàn cờ

- Vì ta lưu bàn cờ dưới dạng ma trận 8*8 nên khi lưu 1 nước cờ kế tiếp là ta đi lưu cặp số [i,j] là vị trí trên bàn cờ .
- Nguyên lí hoạt động : ta xét hết tất cả các ô trên bàn cờ để tìm ô là nước đi hợp lệ , nếu sd trợ giúp “Hiện nước đi hợp lệ” thì dấu hiệu nhận biết là các ô sáng màu xanh dương . Ta đi lưu vị trí [i,j] tại các ô này vào 1 List

```
List<int[]> EnableStepsList = new List<int[]>();
```

Và add vị trí hợp lệ bằng lệnh

```
EnableStepsList.Add(new int[] { i,j });
```

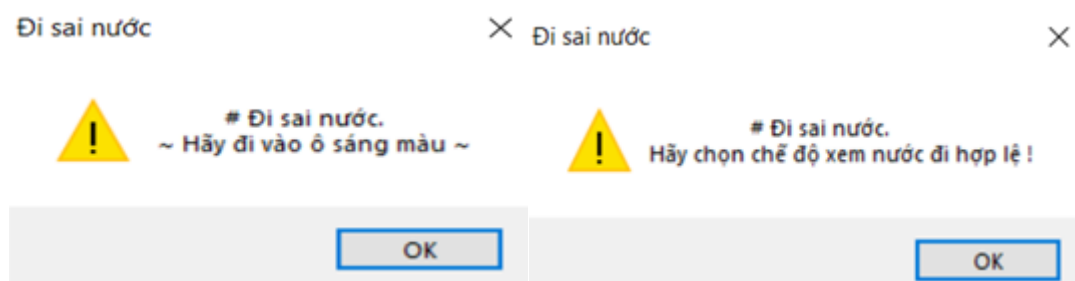
3.1.3. . Một nước cờ được di chuyển như thế nào ?

- Khi máy di chuyển : Sử dụng GetEnableSteps tạo ra các applicable_actions của máy. Ứng với mỗi bước di chuyển đó sẽ là 1 bàn cờ mới được tạo ra tương đương với 1 nước đi của máy, bàn cờ lúc này sẽ là do phe người xử lý. Sử dụng minimax với mỗi bàn cờ mới được tạo ra này (đang ở tầng MAX) để tìm giá trị bàn cờ tốt nhất mà phe người có thể đi được (lúc này phe người đi quân đen, giá trị bàn cờ càng thấp là càng lợi thế cho quân đen). Sau đó ứng dụng nguyên lý hoạt động của cây minimax, tìm nước đi tốt nhất của máy có thể đi được là nước đi khiến cho giá trị bàn cờ là lớn nhất trong các bàn cờ đã tạo ra, tức là các bàn cờ tạo ra dùng để tính minimax ở trên. Hành động của máy lúc này là tìm ra nước đi khiến cho phe người chịu nhiều tổn thất nhất, phe người càng tổn thất thì giá trị minimax trả về sẽ càng cao, mặc dù giá trị bàn cờ trong cây minimax của phe người đã là tốt nhất có thể cho phe người.

Vì bản chất của cây minimax là tìm giá trị bàn cờ minimax của phe địch, sao cho bàn cờ đó là tệ nhất trong những bàn cờ tốt nhất của phe địch.

Sau khi tìm ra nước đi tốt nhất thì máy tiến hành di chuyển làm thay đổi vị trí các quân cờ trên bàn cờ, sau đó sử dụng thao tác thay đổi giao diện trên bàn cờ.

- Khi người di chuyển : ghi nhận vị trí đúp chuột, nếu đó là vị trí không hợp lệ sẽ hiện ra 1 bảng thông báo nhắc nhở “#Đi sai nước.”, nếu hợp lệ sẽ làm thay đổi vị trí các quân cờ, sau đó sử dụng thao tác thay đổi giao diện bàn cờ.



3.1.4. Cài đặt thuật giải

Việc cài đặt thuật giải đơn thuần chỉ là cài đặt hàm minimax bằng phương pháp cắt tỉa alpha beta theo tư tưởng đã được đề cập

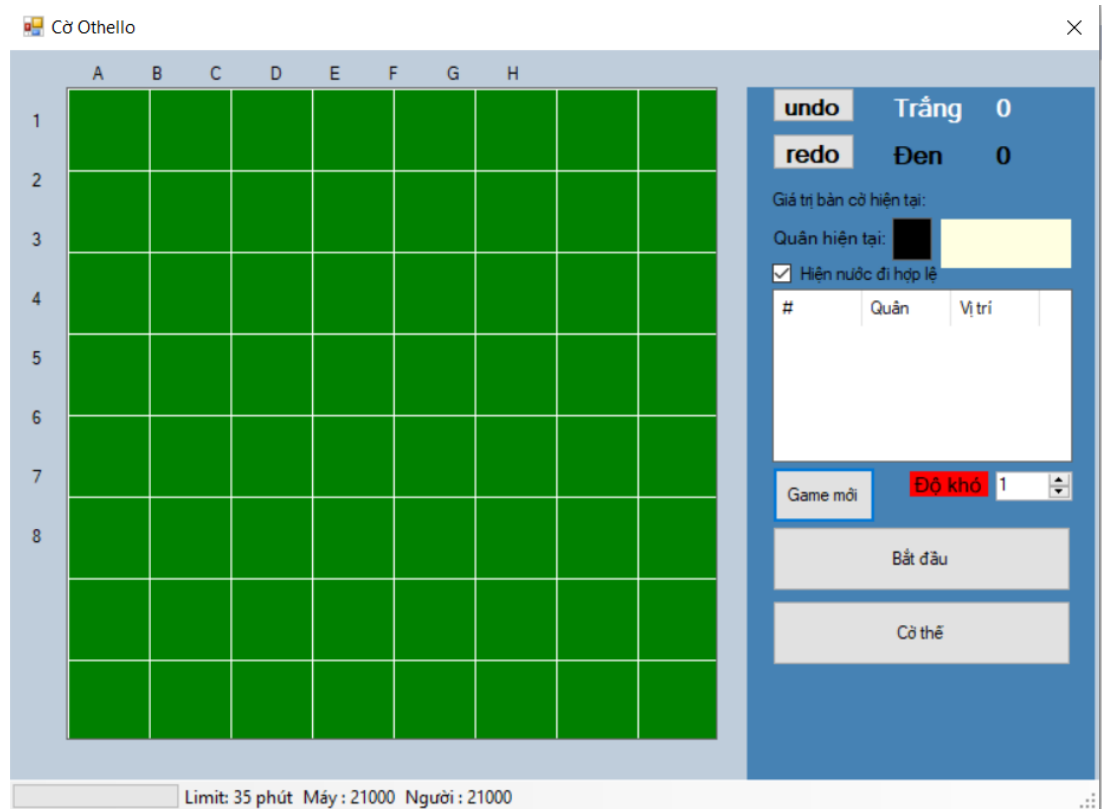
3.2 Cài đặt giao diện và các chức năng khác cho chương trình

3.2.1. Ý tưởng

- Dùng 1 biến started để kiểm tra bàn cờ đã bắt đầu hay chưa , started = true đã bắt đầu trận đấu , started = false là trận đấu chưa bắt đầu hoặc đã kết thúc , phòng khi chưa bắt đầu trận đấu mà người chơi bấm lung tung .
- Dùng 1 biến flag để đánh dấu lượt đi của 2 quân cờ , flag = true là máy đánh , flag = false là người đánh .
- Dùng 1 biến fail để xét tính khả thi của Undo và Redo , nếu fail = 1 thì sử dụng được còn nếu fail = 0 thì hiện bảng thông báo không thể sử dụng chức năng này .
- Dùng 1 biến Count để đếm số lượng bàn cờ đã lưu , dùng cho tính năng Undo và Redo .
- Dùng 1 biến helped để xét liệu đã sử dụng Undo hay chưa ?

helped = 1 nếu đã sử dụng Undo , helped = 0 nếu chưa sử dụng Undo . Khi Undo chưa được sử dụng thì ta sẽ lưu trạng thái bàn cờ máy đi vào 1 List . Khi Undo đã được sử dụng để đi nước cờ mới thì Clear List dùng để lưu trạng thái bàn cờ đi và bắt đầu lưu các trạng thái bàn cờ máy đi sau khi đã sử dụng Undo , điều này đồng nghĩa với việc không thể Undo các trạng thái đã bị Clear .

3.2.2. Màn hình chính



Hình 4.1

Việc cài đặt cho trường hợp này rất đơn giản, ta chỉ việc kéo thả các control trong toolbox

3.2.3. Chọn độ khó bàn cờ

- Được thể hiện bằng chức năng :



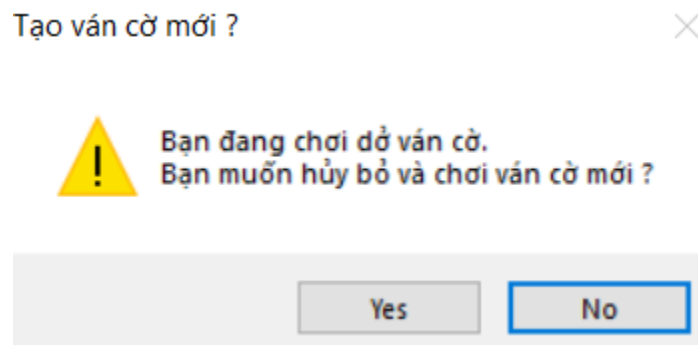
- Độ khó được thể hiện qua biến depth (giới hạn là 10) trong hàm tính minimax và được lấy thông qua lệnh :

```
int difficulty = Convert.ToInt32(numericUpDown1.Value);
```

- Control này chỉ hiển thị giá trị là những con số mà bạn có thể chỉnh lên xuống tùy ý.
- Giá trị mà nó trả về là dạng số nguyên , ta có thể chỉnh định dạng của nó thành chuỗi (string) nếu cần. Control này thì bản thân nó đã không cho phép chứa các ký tự không phải số rồi .

3.2.4. Trò chơi mới (new game)

Sử dụng khi ta thấy không thể thắng được máy và muốn chơi lại ván cờ mới :



Sử dụng ResetMat() để đưa bàn cờ về trạng thái chưa bắt đầu ván đấu và tự do lựa chọn độ khó .

3.2.5. Redo/Undo .

- Undo : Dùng để khôi phục lại quyền đi trước đó của người chơi , được triển khai bằng cách : sau mỗi nước đi của máy , lưu các trạng thái bàn cờ đó vào 1 danh sách . Khi cần thực hiện Undo , hệ thống sẽ lấy bàn cờ ngay trước bàn cờ hiện tại trong danh sách các bàn cờ đã lưu ở trên , vì bàn cờ hiện tại cũng là bàn cờ sau nước đi của máy vậy nên cũng được lưu vào trong danh sách , khi muốn trở lại bàn cờ trước đó ta sử dụng Redo .
- Redo : Dùng để khôi phục lại bàn cờ trước khi Redo .

3.2.6. Luồng chạy giữa máy và người .

- Khởi đầu trận đấu với bàn cờ khởi tạo , khi này biến flag = false và quyền đi nước cờ đầu tiên là thuộc về người chơi .

- Trước khi người đi phải kiểm tra biến trạng thái flag có bằng false hay không , nếu biến flag = true thì thao tác của người không được ghi nhận , nếu biến flag = false và người đi nước cờ của mình và đó là nước đi hợp lệ thì bàn cờ sẽ thay đổi . Biến flag lúc này được chuyển thành true và gọi lệnh yêu cầu máy đi nước cờ của mình .
- Máy thực hiện nước đi của mình khi được gọi , trước khi máy thực hiện nước đi , bộ đếm giờ dành cho máy được bật (60 giây) , trong khoảng thời gian này máy phải xử lý bước đi của mình . Đến lượt máy đi nhưng không có nước đi hợp lệ thì thông báo lại là máy bị mất lượt . Nếu máy đã đi nước cờ của mình và kiểm tra phe người không có nước đi hợp lệ (người mất lượt) thì sẽ hiện thông báo người mất lượt và tiếp tục đi nước cờ tiếp theo của máy cho đến khi người có nước đi hợp lệ . Hoặc nếu máy đã hết nước đi hợp lệ và người cũng hết nước đi hợp lệ thì căn cứ vào tỷ lệ số quân trên bàn cờ và kết luận bên nào thắng . Sau khi máy đi nước cờ của mình , flag sẽ chuyển thành false cho người đi và bật bộ đếm giờ cho phe người đi
- Khi một trong hai phe hết giờ thì dựa vào quân cờ hiện tại để chuyển nước đi kế tiếp cho phe đối diện .Nếu phe bỏ lỡ nước đi là máy thì flag được bật bằng false và chuyển cho người tiếp quản . Nếu phe bỏ lỡ là người thì biến flag được bật bằng true và gọi lệnh cho máy thực thi nước cờ tiếp theo .

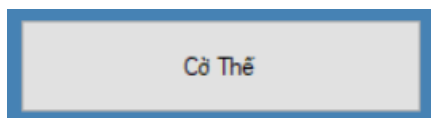
3.2.7. Cờ thề

Sau khi tìm hiểu trên mạng, thì chúng em ko thấy có nguồn tài liệu gì nói về cờ thề trong Othello.

Ở đây chúng em xây dựng cờ thề : tạo 1 thế cờ bất kì bằng cách cho random số lượng quân cờ của 2 phe và random vị trí của các quân cờ.

Sau đó sẽ cho máy đi trước và tiếp đó người sẽ đấu với máy như bình thường .

Khi muốn chơi cờ thề , chúng ta nhấp vào button sau:



3.2.8. Kết thúc trò chơi

TH1 : Khi bàn cờ đã đầy đủ 64 quân cờ , tính toán số quân phe nào áp đảo và hiện ra 1 bảng thông báo người chiến thắng .

TH2 : Khi bàn cờ không đủ 64 quân cờ nhưng có 1 phe đã hết quân cờ , ví dụ : cả bàn cờ đều có quân trắng nhưng chưa đủ 64 ô .

TH3 : Khi cả 2 phe đều hết nước cờ để đi .

TH4 : Khi 1 trong 2 phe hết thời gian 35 phút suy nghĩ .

4. Bugs phát hiện và cách xử lí

- Trong khi xây dựng hàm Undo và Redo có rất nhiều bug nhưng chúng em đã quên không ghi lại và cách xử lí ra sao , nhưng về cơ bản thì những bugs đó đều xuất phát từ việc không hiểu cách hoạt động của chức năng Undo và Redo . Sau khi định hình được tính năng Undo và Redo nên hoạt động như thế nào thì bugs đã được xử lí .
- Một số bugs lẻ tẻ khác xuất phát từ việc chỉnh sửa code mặc dù đã đúng và mất nhiều công sức để tìm lại và sửa (công dã tràng) và đến giờ vẫn còn 1 số code thừa thãi không cần thiết .

- Bug trong hàm tính minimax :

Trước khi tìm ra bug chúng em có hình dùng sai về cây minimax , ngược hẳn lại so với suy nghĩ đúng , đó là :

Tầng MAX là tầng dành cho bàn cờ người đã đi .

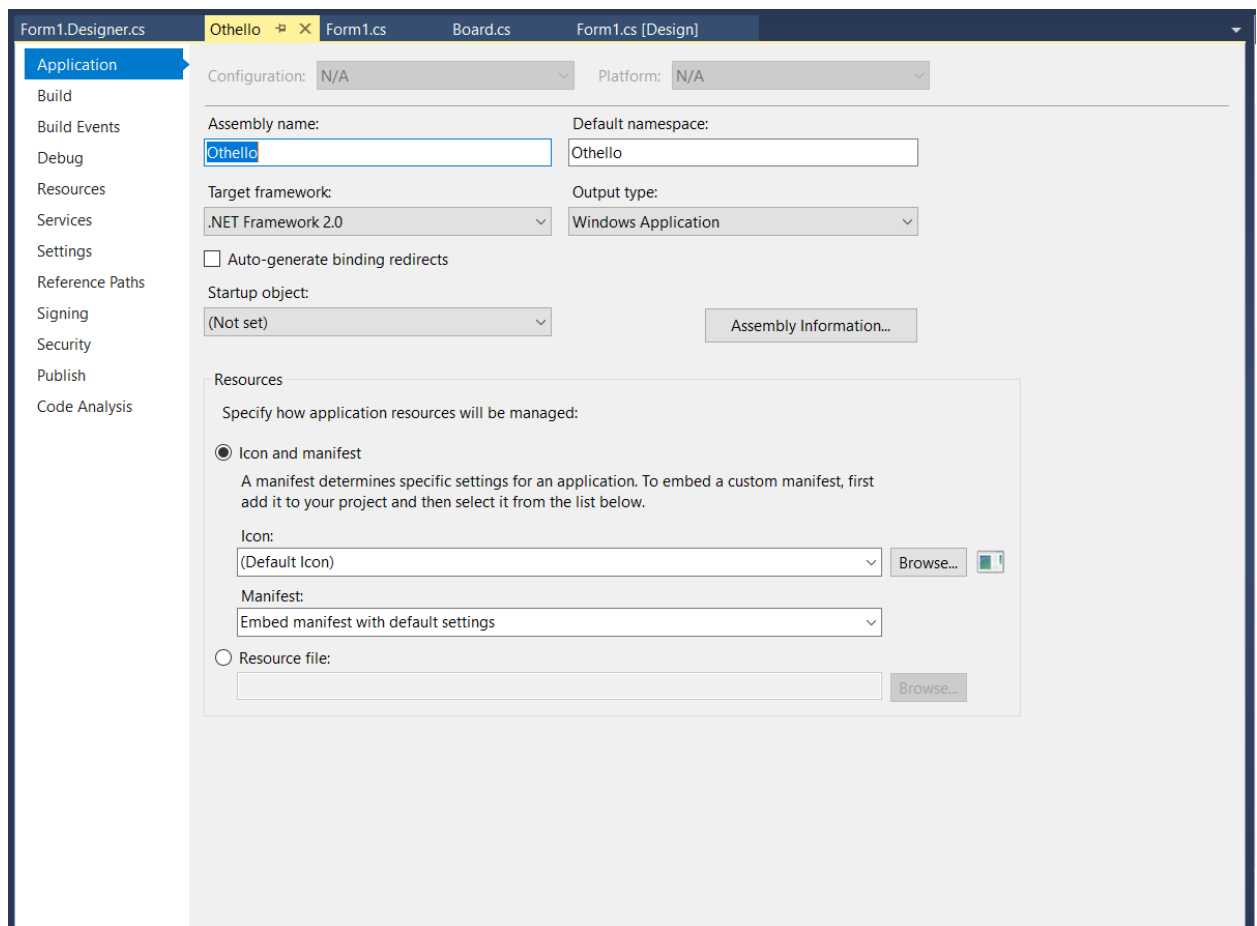
Tầng MIN là tầng dành cho bàn cờ máy đã đi .

Cách xử lí :

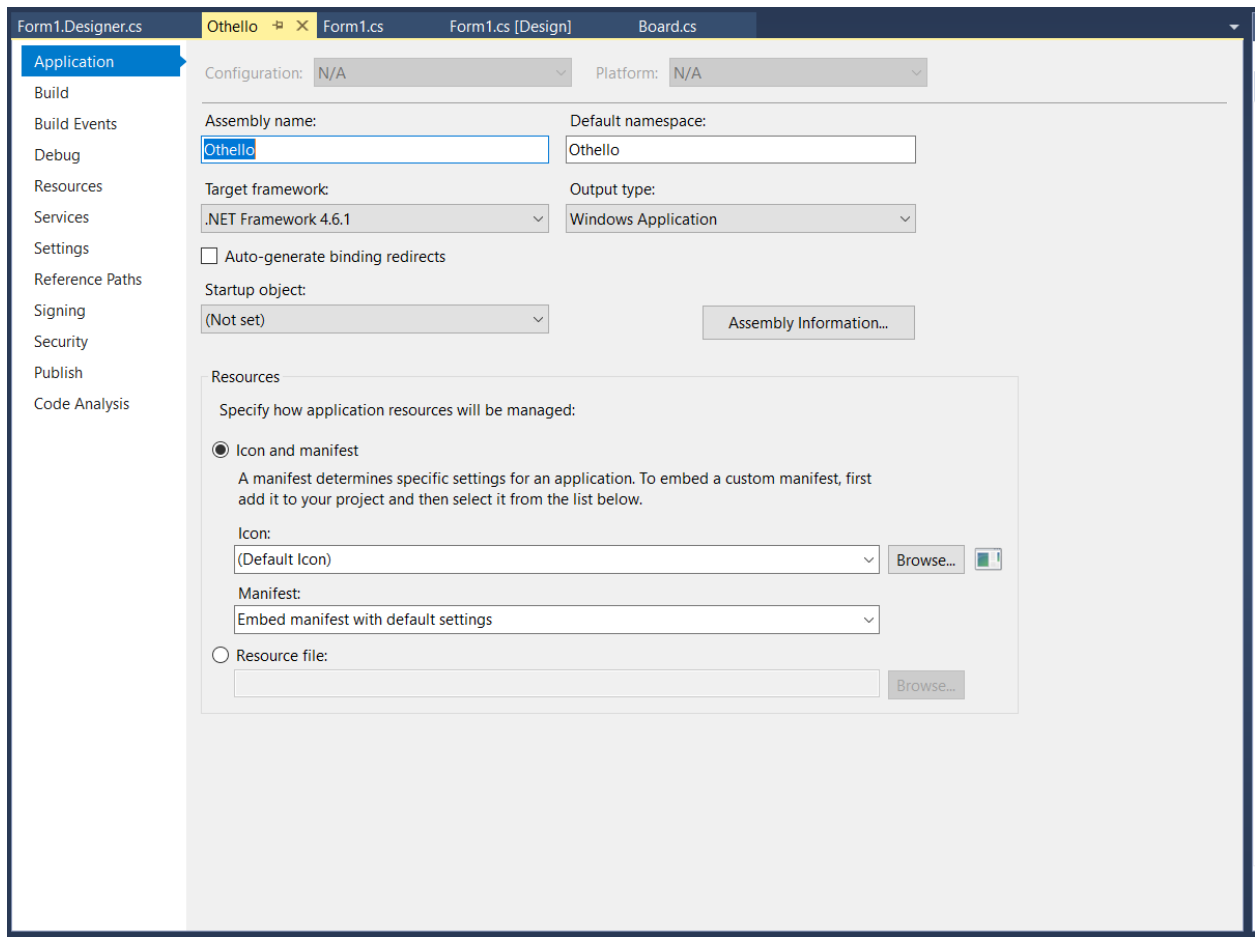
Sau khi phát hiện ra như thế kéo theo EnableStepList trong hàm minimax bị sai và cách tính minimax trong hàm DoBestStep cũng sai luôn (về cơ bản là cả 2 đều bị ngược lại so với cách hoạt động đúng của các tầng), để sửa sai chúng em chỉ cần đảo ngược lại quân cờ trắng và đen ở hàm EnableStepList và DoBestStep.

- **Bug đi được nửa chừng ván đấu bị crash (Cập nhật sau ngày báo cáo 8/7/2019) :**

Vì code ban đầu được xây dựng trên máy tính sử dụng phiên bản .NET Framework 2.0 và đăng folder source code đã xây dựng đồ án lên google drive. Nhưng sau khi được download source code về máy của 1 thành viên khác sử dụng phiên bản .NET Framework 4.6.1 thì code đã bị crash . Thầy có thể chỉnh sửa .Net Framework trong source chúng em nộp từ phiên bản 2.0 thành phiên bản 4.6.1 để thấy lỗi này .

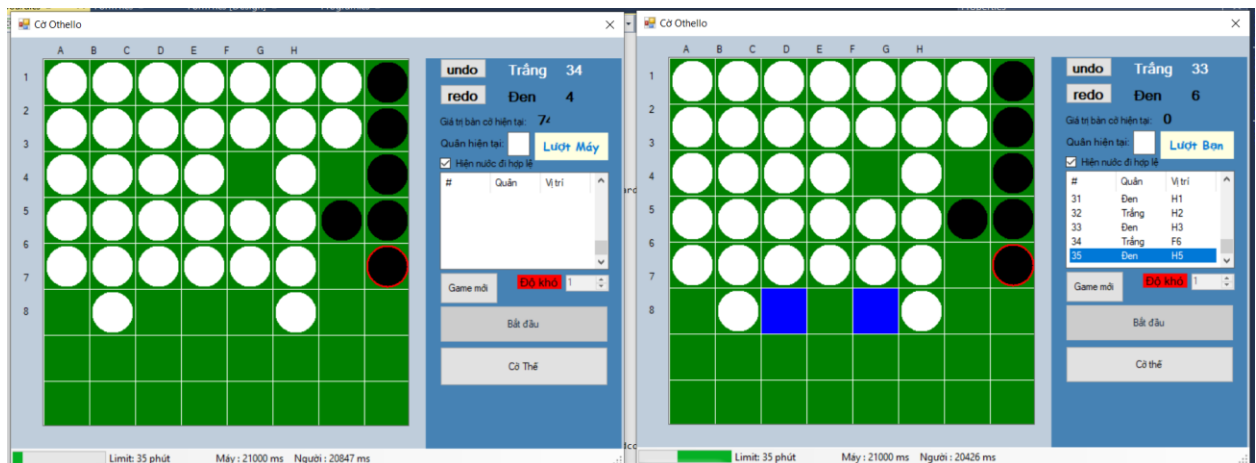


Phiên bản gốc dùng để xây dựng code đồ án (.NET Framework 2.0)



Phiên bản bị lỗi (.NET Framework 4.6.1)

Và đây là lỗi đã được chúng em test lại rõ ràng bằng cách tải source code xây dựng trên phiên bản có lỗi xảy ra và so sánh với phiên bản không bị lỗi :

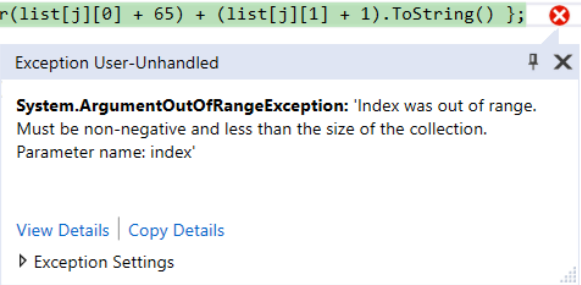


Form bên trái là form bị lỗi với các thông số ở bàn hiển thị các nước đã đi không thể hiển thị khi bị crash(em cũng không biết tại sao) và phiên bản code gốc không bị lỗi.

Lỗi được hiển thị như thế này :

```
private void InsertToTable(List<int[]> list, int j)
{
    int m = listView1.Items.Count + 1;
    string[] stritems = { m.ToString(), "Trắng", Convert.ToChar(list[j][0] + 65) + (list[j][1] + 1).ToString() };
    //thực tập
    //string Undoeed = "##### Đã undo !!!!!!!";
    //ListViewItem Undo_ListViewItem = new ListViewItem(Undoeed);
    //
    ListViewItem newitem = new ListViewItem(stritems);
    listView1.Items.Add(newitem);
    listView1.EnsureVisible(m - 1);
    listView1.Items[m - 1].Selected = true;

    for (int i = 0; i < m - 1; i++)
    {
        listView1.Items[i].Selected = false;
    }
    board.X_Pre = list[j][0];
    board.Y_Pre = list[j][1];
}
```



- Bug Undo , Redo (Cập nhật sau ngày báo cáo 8/7/2019) :

Do cách suy nghĩ sai về tính năng Undo,Redo là khi đã sử dụng Undo thì những bàn cờ đã lưu trong danh sách dùng để lưu bàn cờ sẽ bị xóa đi và không thể Undo những nước cờ không dùng tới ,tức là sau khi sử dụng Undo để đi nước cờ mới thì không thể quay trở lại những bàn cờ trước bàn cờ đang xét , khi đó người chơi chỉ có thể Undo nước cờ mới nhất đã lưu trong danh sách khi mà những nước cờ mới được đi bởi người sau khi đã sử dụng Undo .

Cách xử lí :

Chúng em chỉ cần giữ nguyên các bàn cờ chưa được sử dụng sau nước cờ đi Undo trong danh sách , và sau 1 nước đi mới nhất của máy thì tiếp tục lưu vào danh sách sử dụng để thao tác với Undo,Redo . Redo không có gì thay đổi .

- Bug tính giờ (Cập nhật sau ngày báo cáo 8/7/2019) :

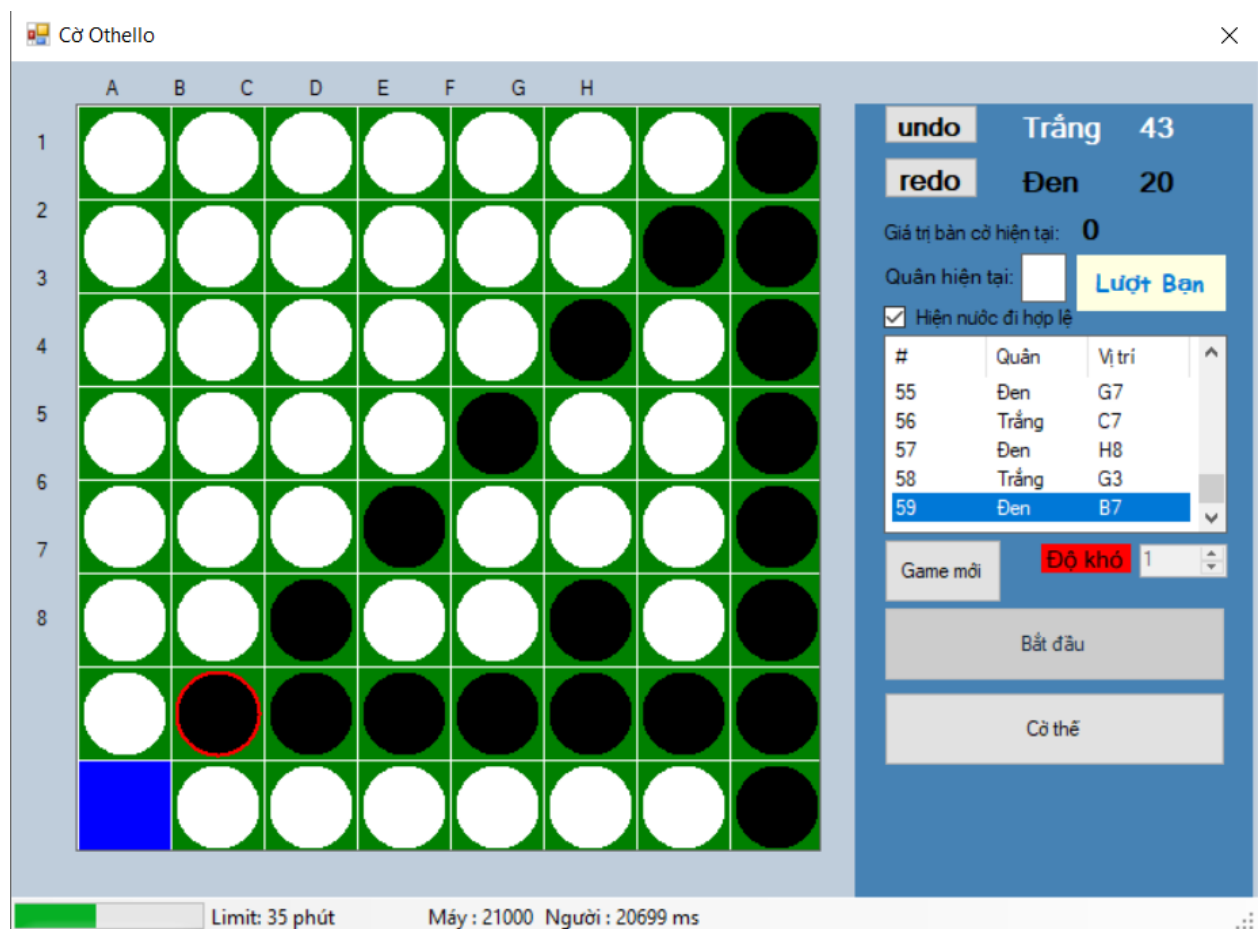
Vẫn do cách suy nghĩ sai về luật tính giờ chơi cờ quốc tế là mỗi người chơi chỉ có thời gian giới hạn cho 1 nước đi (thời gian suy nghĩ 1 nước đi) là 1 phút , sau 1 phút mà người chơi hiện tại không đi nước cờ của mình thì sẽ mất lượt . Sau khi được thầy chỉ lỗi sai , chúng em lại lái tư tưởng sang 1 suy nghĩ sai khác là cả 1 ván đấu chỉ có trong vòng 35 phút và không giới hạn thời gian suy nghĩ của mỗi người chơi . Sau 35 phút , cho dù chưa hết ván cờ thì vẫn sẽ thông báo kết thúc ván đấu và kết luận người thắng cuộc .

Cách xử lí :

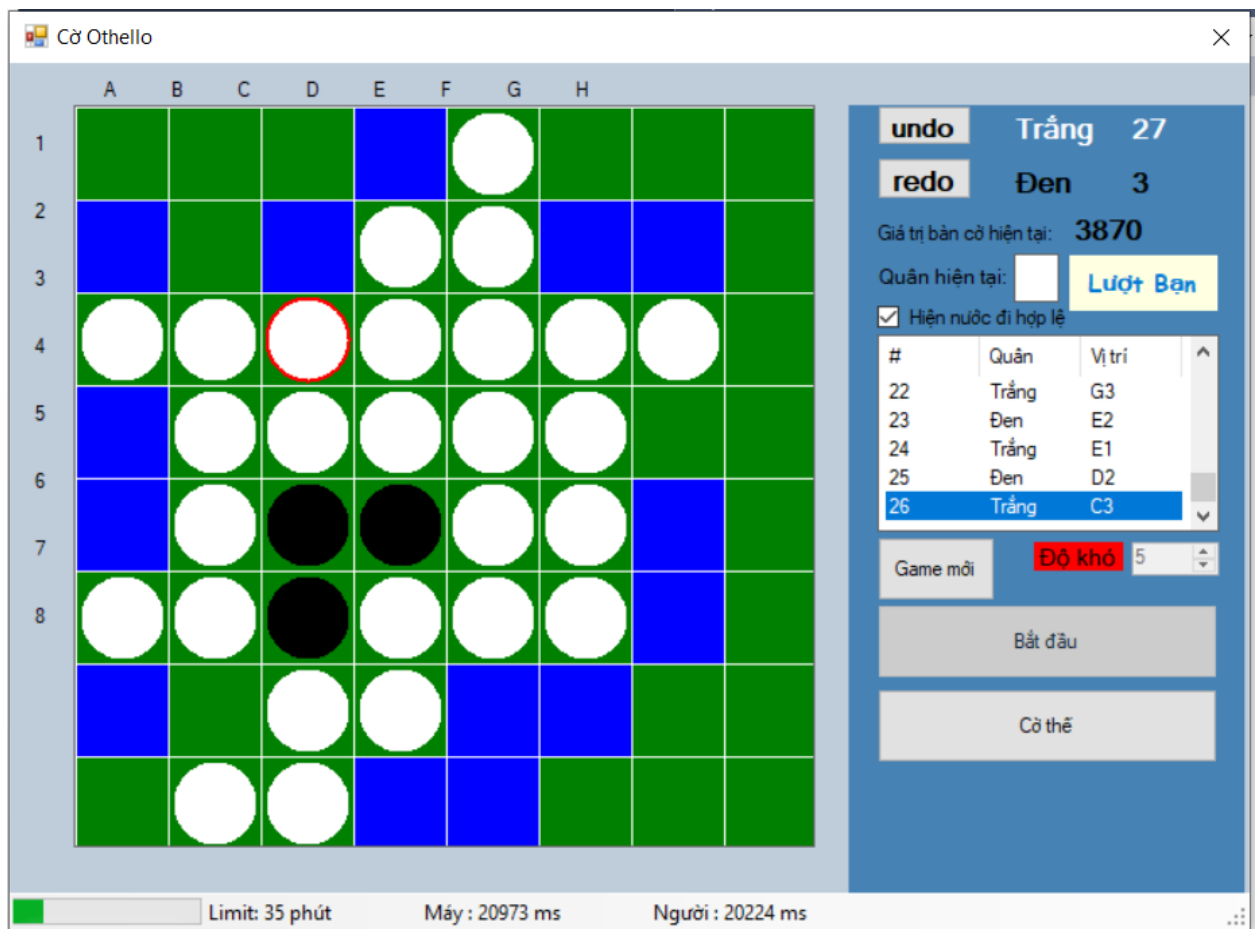
Sau khi tìm hiểu được là không phải cả ván đấu bị quy định trong vòng 35 phút mà là mỗi người chơi có 35 phút để chiến thắng , nếu phe nào hết thời gian 35 phút trước sẽ bị xử thua thì chúng em sửa lại . Vì C# cho phép chạy với thời gian là milliseconds và 35 phút = $35 \cdot 60 \cdot 10 = 21000$ milliseconds nên nếu hết thời gian 21000 milliseconds , đồng hồ đếm giờ của phe người chơi đó sẽ báo kết thúc và thông báo người chơi hiện tại thua . Sau mỗi lượt đi của 1 người chơi thì đồng hồ đếm ngược của người chơi đó sẽ dừng lại và đồng hồ đếm ngược của phe địch sẽ bắt đầu chạy .

Nếu người chơi không có nước cờ để đi (không có nước cờ nào hợp lệ kế tiếp)thì đồng hồ sẽ lập tức tính giờ cho phe đối diện , vì người chơi hiện tại bị mất lượt .

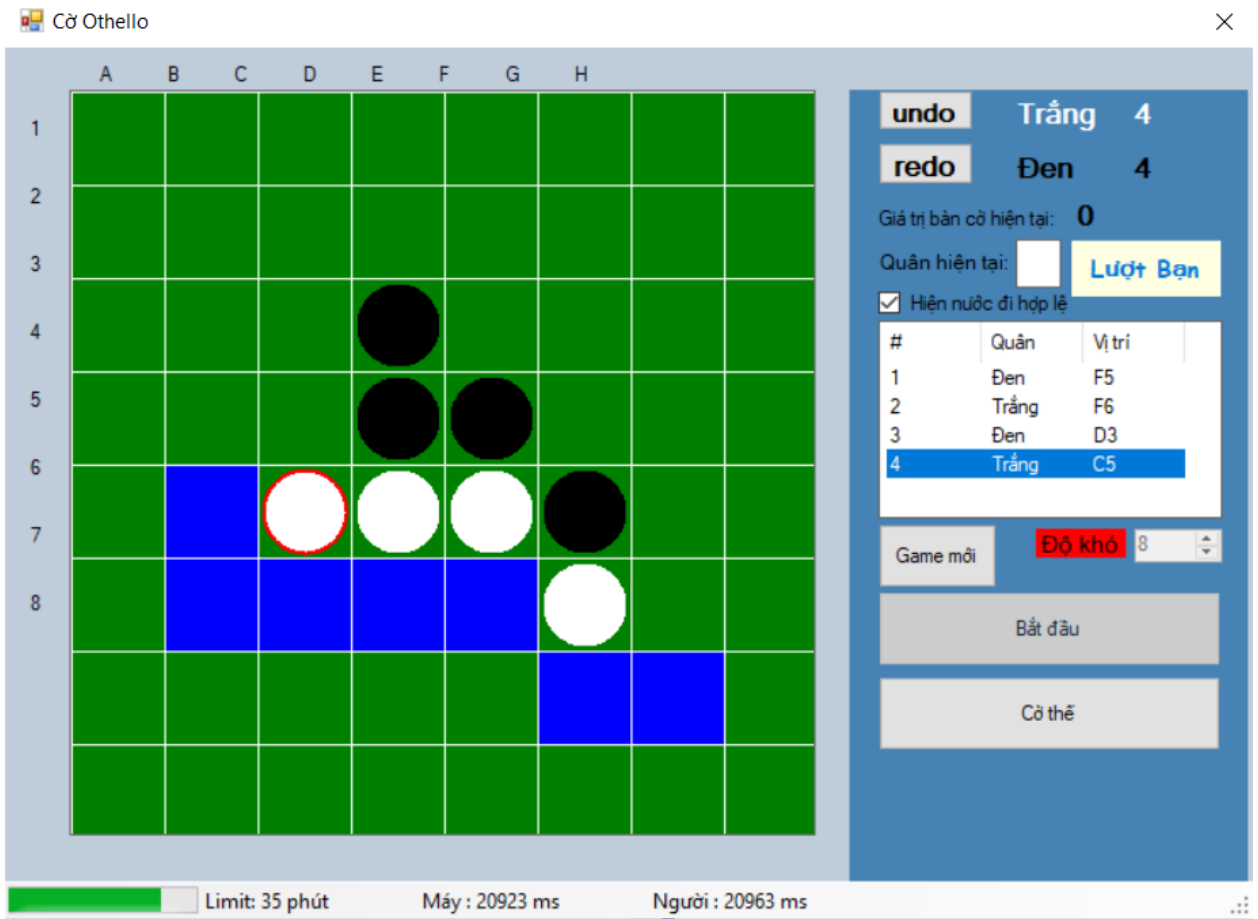
Và sau đây là 1 số thông tin chúng em tìm được sau khi xử lí bug tính giờ :



Ở độ sâu bằng 1 , máy đánh quá nhanh , thậm chí sắp hết ván đấu mà máy vẫn chưa mất 1 millisecond .



Bắt đầu ở độ sâu bằng 5 máy đã bắt đầu có dấu hiệu tính toán lâu hơn ở mỗi nước đi , nhưng tốc độ của máy vẫn còn khá nhanh , các quân cờ đã chiếm nửa bàn cờ nhưng máy vẫn chỉ mất có 27 milliseconds.



Ở độ sâu bằng 8 , tốc độ tính toán của máy lâu hơn rõ rệt ở ngay những nước cờ đầu tiên .

5. Kết luận

Thông qua việc áp dụng trí tuệ nhân tạo và các kiến thức đã học, chúng em đã bước đầu tạo được một trò chơi cờ Othello ở mức độ đơn giản để có thể chơi với con người.

Chất lượng của chương trình phụ thuộc phần lớn vào hàm Heuristic, tuy nhiên hàm Heuristic của chương trình còn chưa đủ mạnh và cần được cải thiện nhiều. Ví dụ, ở hàm đánh giá có thể thêm một lớp các giá trị tương ứng thay vì chỉ sử dụng 4 giá trị: `edge_value`, `corner_value`, `differentce_pieces`, `same_color`. Có nhiều cách để tạo lớp giá trị cho bàn cờ.

Chương trình có thể được hoàn thiện tốt hơn nếu chúng em có thể sử dụng được các công nghệ của máy học (reinforcement learning). Tuy nhiên do khả năng và kiến thức còn hạn hẹp điều này vẫn chưa thể thực hiện được.

Trong thời gian thực hiện báo cáo không thể tránh khỏi những thiếu sót, nhóm chúng em rất mong nhận được sự nhận xét và giúp đỡ của thầy để nhóm có thể hoàn thành chương trình một cách hoàn thiện hơn.

6. Tài liệu tham khảo

[1] Wikipedia Othello :

https://vi.wikipedia.org/wiki/C%E1%BB%9D_Othello

[2] Hướng dẫn chơi Othello :

https://gamevh.net/cms/static/guide_othello.jsp

[3] C# là gì? tổng quan về C# :

<https://freetuts.net/c-sharp-la-gi-tong-quan-ve-c-sharp-1045.html>

[4]Giới thiệu window form :

<https://sites.google.com/site/1074netshare/home/software-technology/programming/c/desktop-user-interfaces/windows-forms/chuong-1---gioi-thieu-windows-forms>

[5]Giải Thuật Cắt Tia Alpha-beta :

<https://www.stdio.vn/articles/giai-thuat-cat-tia-alpha-beta-564>

[6]Giải thuật tìm kiếm Minimax :

<https://www.stdio.vn/articles/giai-thuat-tim-kiem-minimax-283>

[7] Nguyen Gia Bach's Blog, Trí tuệ nhân tạo – Dodgem – AlphaBeta :

<https://nguyengiabach.wordpress.com/trang-web-c%E1%BB%A7a-con/cntt/tri-%20tue-nhan-tao-dodgem-alphabeta/>

[8] Nguồn tham khảo heuristic :

<http://www.codearsenal.net/2012/06/reversi-game-in-csharp-winforms.html#.XSGop6ITpPY>