TRƯỜNG ĐẠI HỌC HÀNG HẢI VIỆT NAM

KHOA CÔNG NGHỆ THÔNG TIN

-----*-----**

BÀI TẬP LỚN

TRÍ TUỆ NHÂN TẠO

Đề tài:

THUẬT TOÁN BFS
ỨNG DỤNG TRONG TRÒ CHƠI Ô SỐ

GVHD: **thầy Nguyễn Duy Trường Giang**

Sinh viên:

1. **Vũ Tiến Chung – 82418**
2. **Đàm Văn Đức - 82779**

LỜI 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 trò chơi 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 bài tập lớn 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 thầy Nguyễn Duy Trường Giang, chúng em đã hiểu được cách thức hoạt động của thuật toán và hoàn thành bài báo cáo bài tập lớn 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!

MỤC LỤC

Trang

CHƯƠNG I:

GIỚI THIỆU VÀ MÔ TẢ BÀI TOÁN THỰC TẾ ĐƯỢC GIẢI QUYẾT

1.1 Tên đề tài

- Game Ghép Tranh 8 ô số.

1.2 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.

1.3 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.

CHƯƠNG II:
CÁC CHI TIẾT CỦA PHƯƠNG PHÁP DÙNG ĐỂ GIẢI QUYẾT BÀI TOÁN

2.1. 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 đỉnh 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**

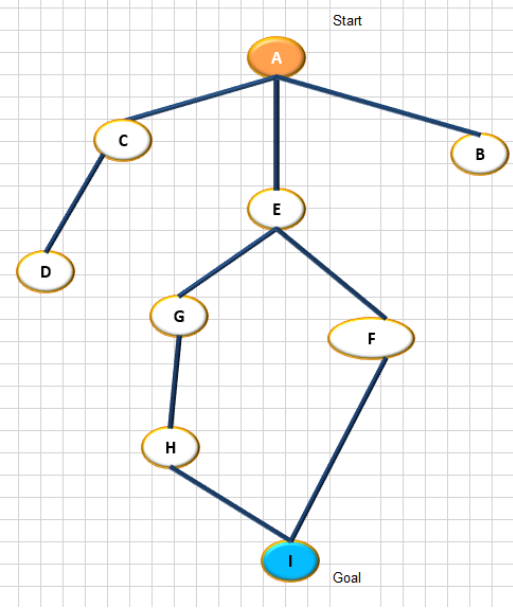
{Thông báo tìm kiếm thành công; Stop};

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:



Đỉnh	Danh sách kề	Danh sách L
A	C , E , B	E , C , B
E	G , F	C , B , G ,F
C	D	B , G ,F , D
B		G , F , D
G	H	F , D , H
F	I	D , H , I
D		H , I
H		I
I	Trạng thái kết thúc	

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.

2.2. 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.

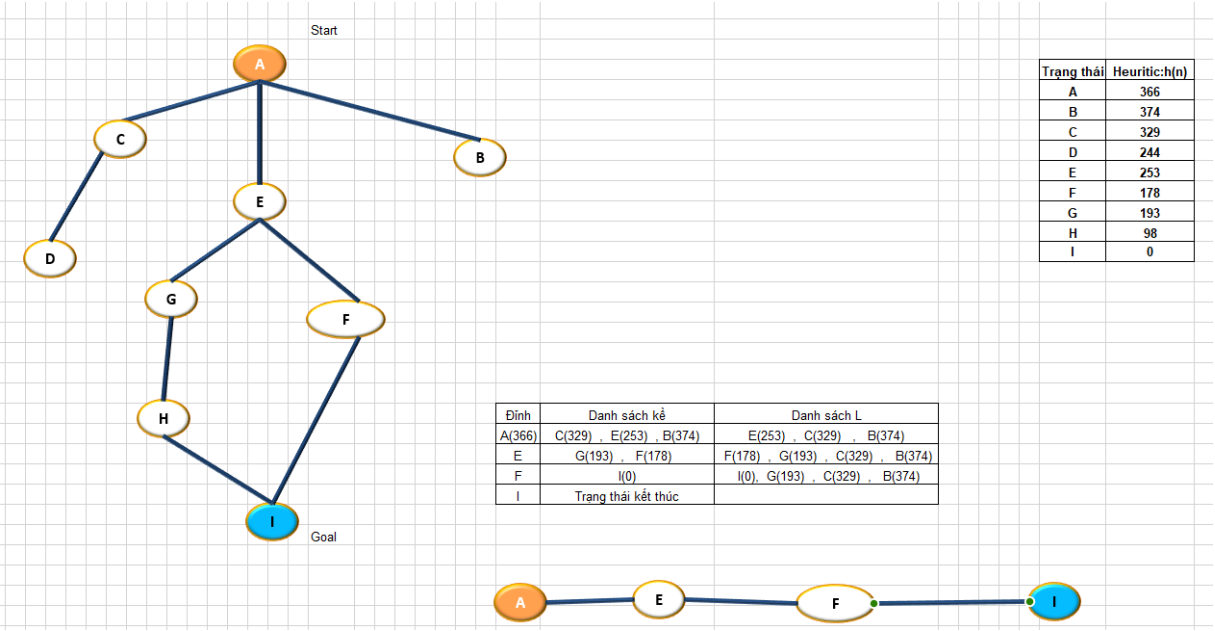
- **Chi tiết chạy từng bước của thuật toán** Best First Search**:**

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:



Trạng thái	Heuritic:h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0

Đỉnh	Danh sách kề	Danh sách L
A(366)	C(329) , E(253) , B(374)	E(253) , C(329) , B(374)
E	G(193) , F(178)	F(178) , G(193) , C(329) , B(374)
F	I(0)	I(0), G(193) , C(329) , B(374)
I	Trạng thái kết thúc	

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.

2.3. Ứng dụng thuật toán vào Game ghép tranh 8 ô số

- Áp dụng 2 thuật toán như đã nói trên vào game 8 ô số. Lúc này mỗi trạng thái hay mỗi đỉnh mà thuật toán duyệt qua chính là một danh sách các số từ 1 đến 9, và trong đó có 1 ô số đánh dấu là ô đen trống để cho người chơi di chuyển.
- Mỗi lần thuật toán duyệt qua một trạng thái, sẽ đưa vào trong hàng đợi, như vậy ta sẽ có danh sách chứa những danh sách.
- Kết quả đường đi tìm được sẽ trả về danh sách của những trạng thái mà nó đã tìm ra(cũng là một danh sách các số từ 1 đến 9).

<u>*Hướng dẫn chơi game Ghép Tranh 8 ô số:</u>

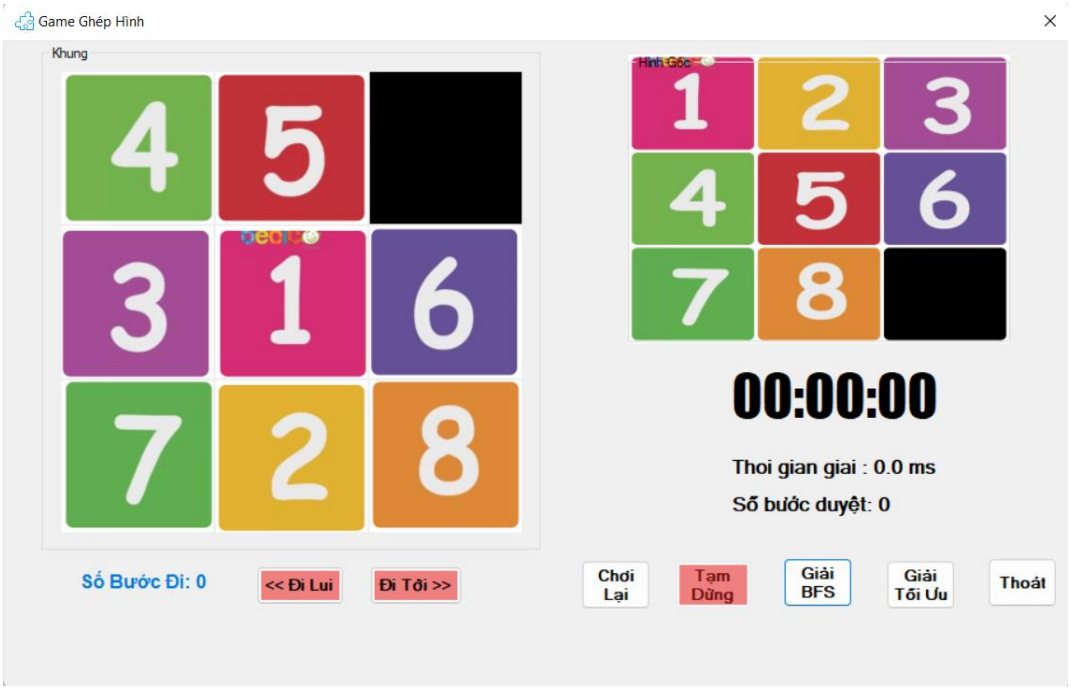
- Khi người dùng khởi động chương trình lên. Màn hình sẽ hiển thị bên táy phải là hình gốc (hình hoàn chỉnh sau ghi ghép đúng bức tranh), bên tay trái là 9 ô số chứa từng phần nhỏ được cắt ra từ bức tranh hoàn chỉnh, 9 ô số này sẽ có 1 ô là màu đen trống để người chơi di chuyển các ô lân cận, 9 ô này sẽ được trộn lẫn ngẫu nhiên không hoàn chỉnh thành 1 bức tranh.
- Nhiệm vụ người chơi là di chuyển từng ô 1 lên, xuống, trái, phải và không được đi chéo dựa theo ô trống đen, sao cho ghép thành 1 bức tranh hoàn chỉnh như hình bên tay phải.
- Khi người chơi bắt đầu chơi, thời gian sẽ được tính, và số bước đi sẽ được đếm nhằm thống kê thời gian và số bước người chơi đã đi đến kết quả cuối cùng.
- Người chơi có thể tạm ngưng trong lúc chơi, nhưng khi chọn nút tạm ngưng màn hình

9 ô số ngẫu nhiên này sẽ bị tắt đi, phòng trường hợp người chơi xem trước đường đi.

- Người chơi có thể chọn nút chơi lại để bắt đầu chơi lại từ đầu, lúc này 9 ô số sẽ được trộn ngẫu nhiên lại một lần nữa và thời gian cũng như bước đi sẽ được cài đặt về 0 như trạng thái ban đầu.
- Nếu người chơi không thể đi đến ghép thành một bức tranh hoàn chỉnh thì người chơi có thể chọn nút giải theo BFS hay giải Tối Ưu trên màn hình, sau khi chọn thuật toán giải số bước đi sẽ được hiển thị lên, thời gian và số trạng thái đã duyệt qua cũng được hiển thị lên, từ đây người chơi click vào nút đi tới để bắt đầu xem kết quả đi từng bước của thuật toán cho đến khi hoàn chỉnh bức tranh. Người dùng cũng có thể đi lui để xem lại bước đi trước đó.
- Người dùng có thể thoát khỏi chương trình.

CHƯƠNG III:
CÁC CHỨC NĂNG CHÍNH CỦA TRÒ CHƠI

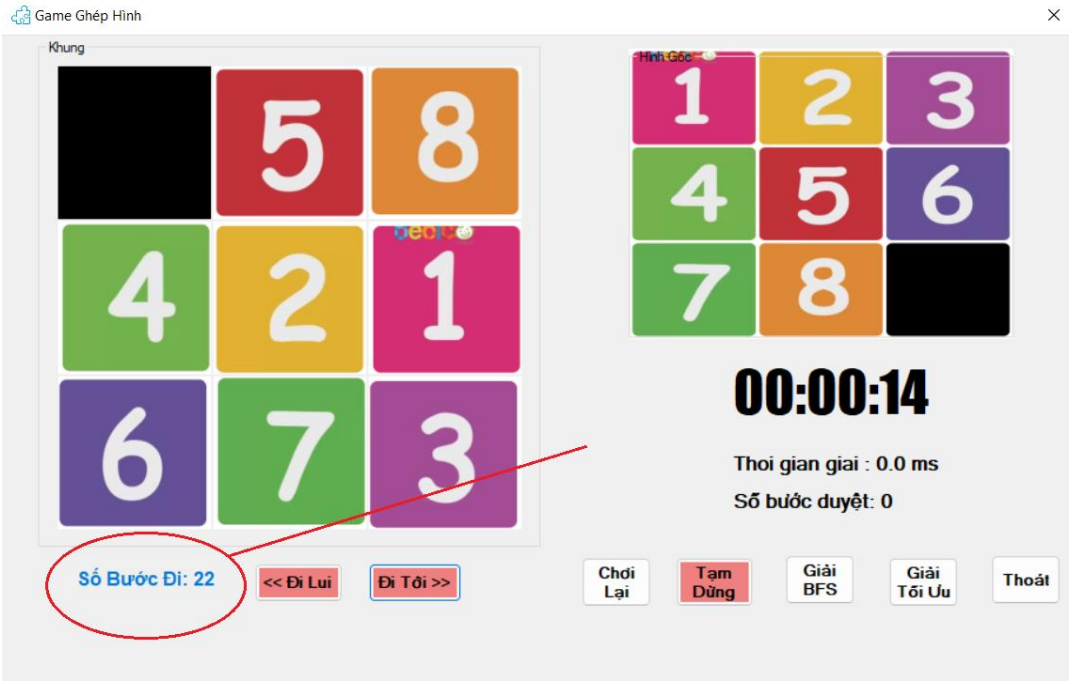
Tổng quan về chức năng, giao diện



Chức năng random hình ảnh lên màn hình

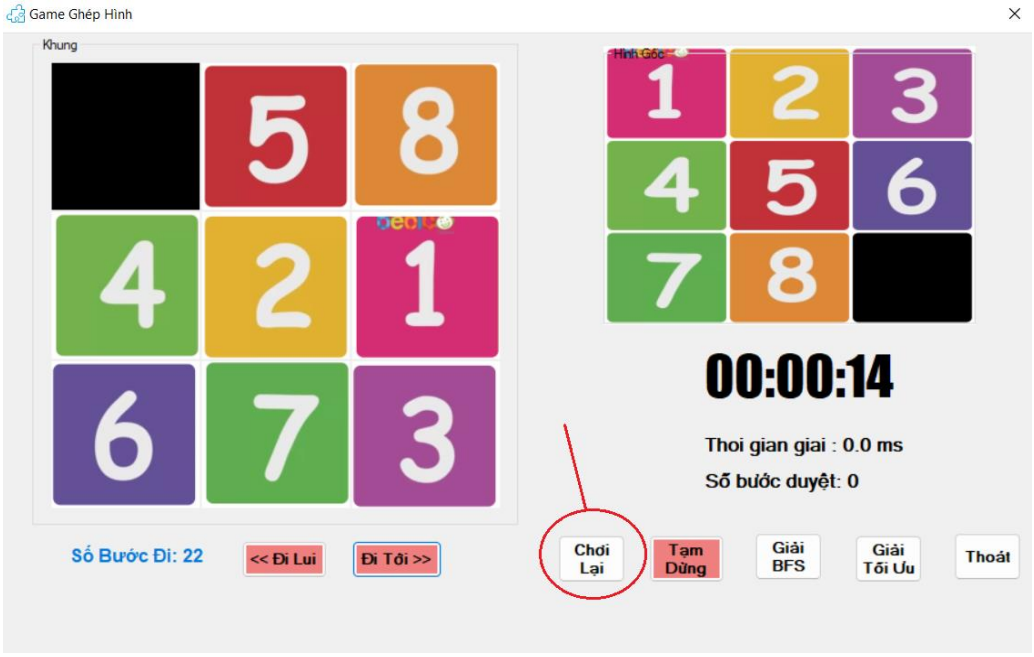
- Khi bước vào trò chơi thì hình ảnh các ô số sẽ được đảo trộn xen lẫn một cách ngẫu nhiên.
- Hoặc khi chúng ta bấm vào nút chơi lại, hình ảnh các ô số cũng được đảo tương tự như vậy.

Chức năng đếm bước đi



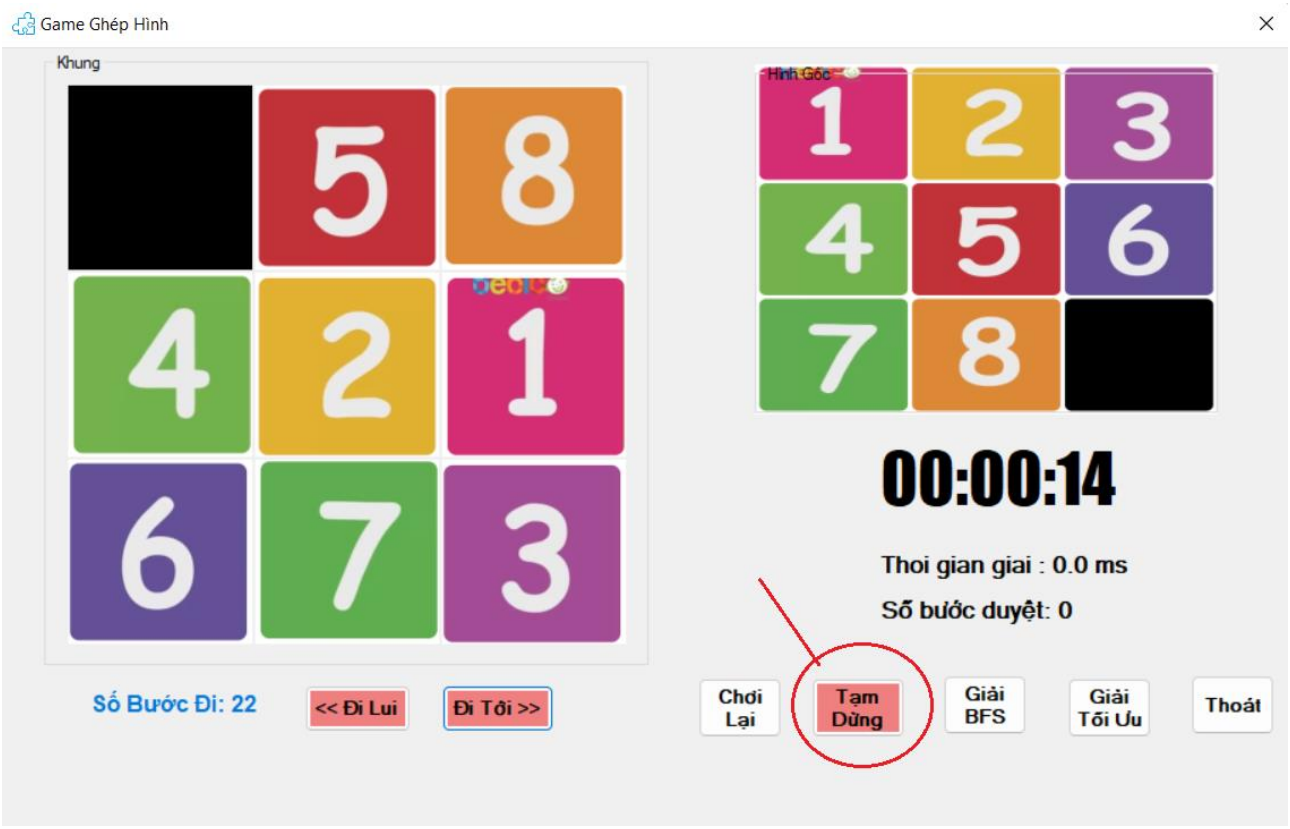
- Khi chúng ta di chuyển các trang thái để sắp xếp hình ảnh thì mỗi lần di chuyển sẽ được tính là một bước.
- Chức năng này giúp đánh giá được bạn đã sử dụng bao nhiêu trạng thái để ghép thành công tấp hình này. Hay gọi cách khác là bạn đã sử dụng bao nhiêu trang thái để hoàn thành trò chơi này.
- Ngoài đánh giá chức năng này còn có thể giúp bạn ghi nhận lại số bước đi để lần tới bạn có thể cải thiện hơn số trạng thái mình đã dùng để chiến thắng trò chơi. Một khi bạn dùng càng ít trạng thái để chiến thắng thì bạn sẽ đánh giá được khả năng chơi trò chơi của mình như thế nào.

Chức năng chơi lại



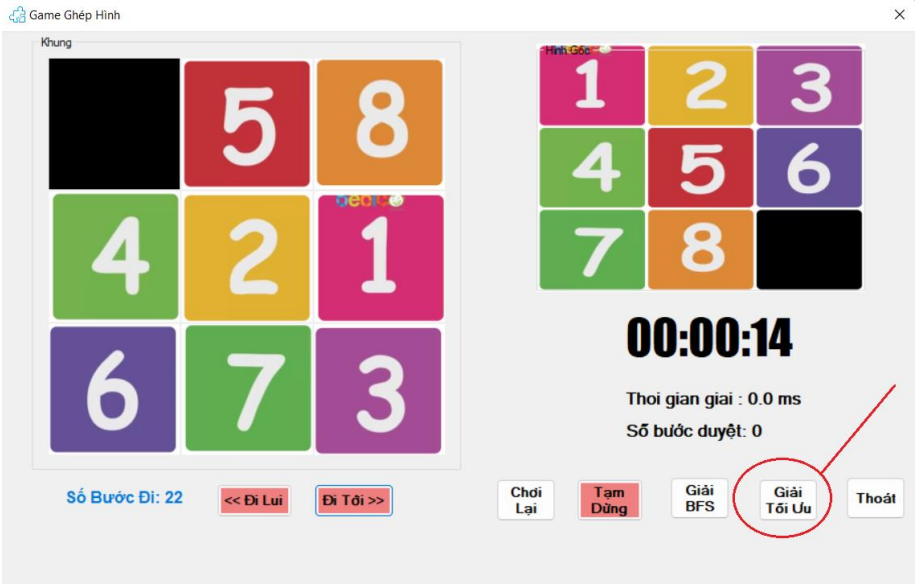
- Khi người chơi muốn chơi gặp khó khăn trong việc di chuyển các ô trên màn hình trò chơi hoặc là người chơi đã tìm ra cách để chiến thắng trò chơi mà tốt ít thời gian nhất cũng như là tốn ít số bước đi để đến chiến thắng nhất thì người chơi có thể chơi lại từ đầu.
- Khi bạn cảm thấy chơi hoài mà không chiến thắng thì bạn cũng có thể bấm chơi lại để bắt đầu lại từ đầu. Khi đó sẽ có một thông báo hỏi bạn có muốn chơi lại từ đầu hay không.
- Khi bạn đã chắc chắn là mình chơi lại từ đầu thì màn hình chơi sẽ random lại các trạng thái một cách ngẫu nhiên.

Chức năng tạm dừng



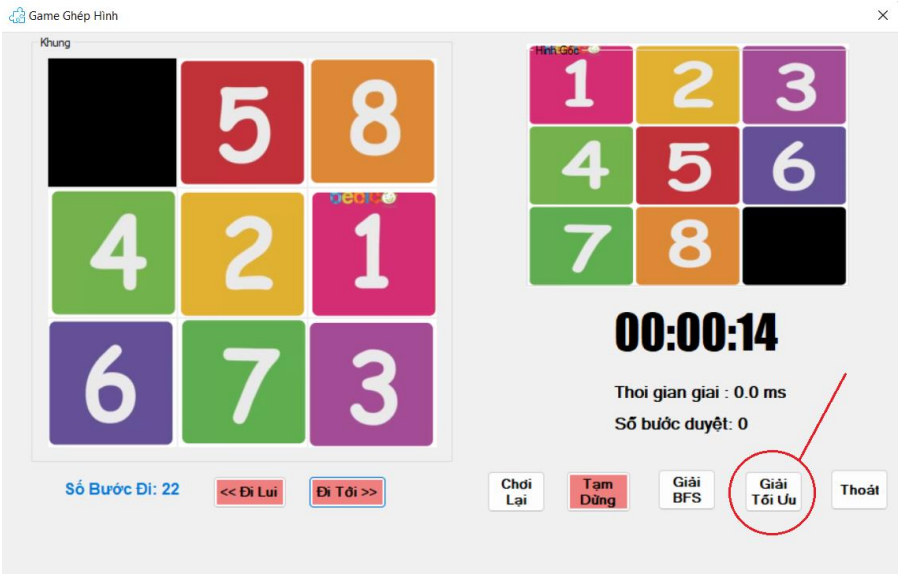
- Khi mới vào màn hình chơi game thì nút tạm dừng sẽ bị ẩn. cho đến khi bạn chơi và bước đi cũng như thời gian bắt đầu được tính thì bạn mới có thể tạm dừng được.
- Chức năng này giúp bạn có thể tạm dừng trò chơi. Khi này, thời gian ở phần đông hồ sẽ dừng đếm.
- Để đảm báo tính hấp dẫn và thú vị cho trò chơi thì khi bạn bấm tạm dừng thì hình ảnh trên màn hình của bạn sẽ mất và chỉ còn để lại tấm hình mẫu bên phải mà thôi. Còn các trạng thái của bạn đang sắp xếp dở thì sẽ biến mất. Khiến cho bạn không thể tạm dừng để suy nghĩ các bước đi của mình được.
- Khi đó button tạm dừng sẽ chuyển thành tiếp tục để người chơi chọn chơi tiếp.

Chức năng giải BFS



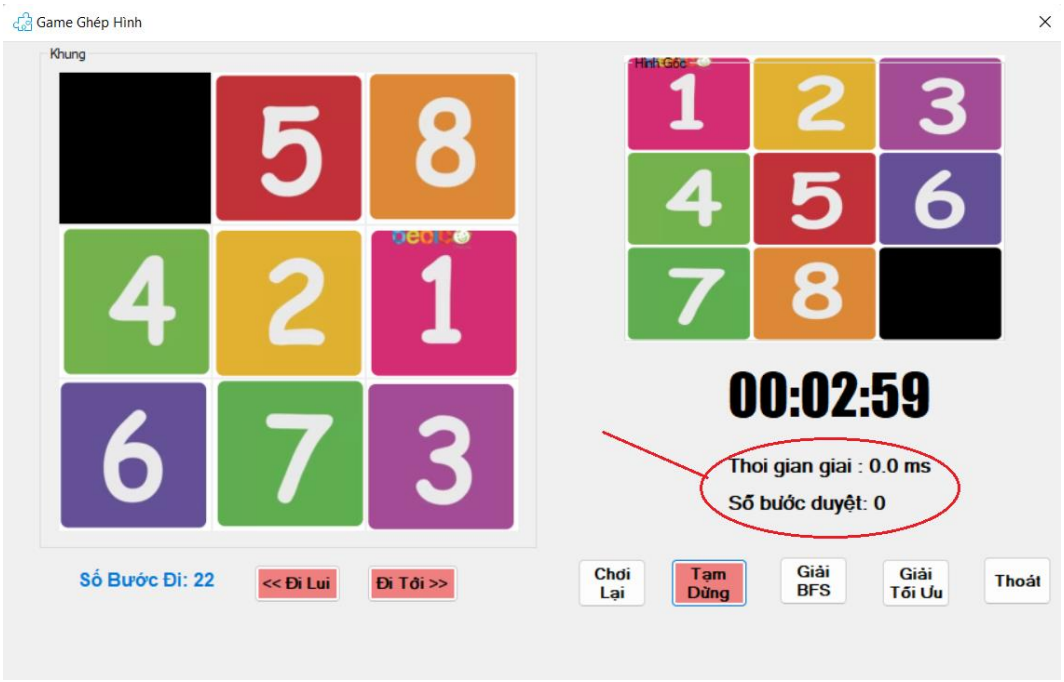
- Click vào button giải BFS thì người dừng phải đợi một chút để máy có thể tự giải. Tới khi nào phần đếm bước đi hiển thị số bước.
- Thì khi đó thuật toán đã được giải và sẽ hiển thị ra màn hình cả thời gian và số bước duyệt để chọn ra được những trạng thái cuối cùng để người chơi có thể đi đến chiến thắng.

Chức năng giải tối ưu



- Khi bạn click vào button giải tối ưu thì người dùng sẽ chờ một chút. Tới khi mà phần đến bước đi hiển thị ra số bước đi thì khi đó trò chơi mới tự được máy giải xong.
- Chức năng này cũng sẽ hiển thi ra các thời gian và số trạng thái đã được duyệt qua để đi đến trạng thái kết thúc.
- Việc hiển thị số bước để duyệt hay còn gọi là số trạng thái để đi đến trạng thái cuối cùng, sẽ giúp người dùng so sánh được hai phương pháp giải với nhau.

Chức năng đếm thời gian giải và số bước duyệt của thuật toán

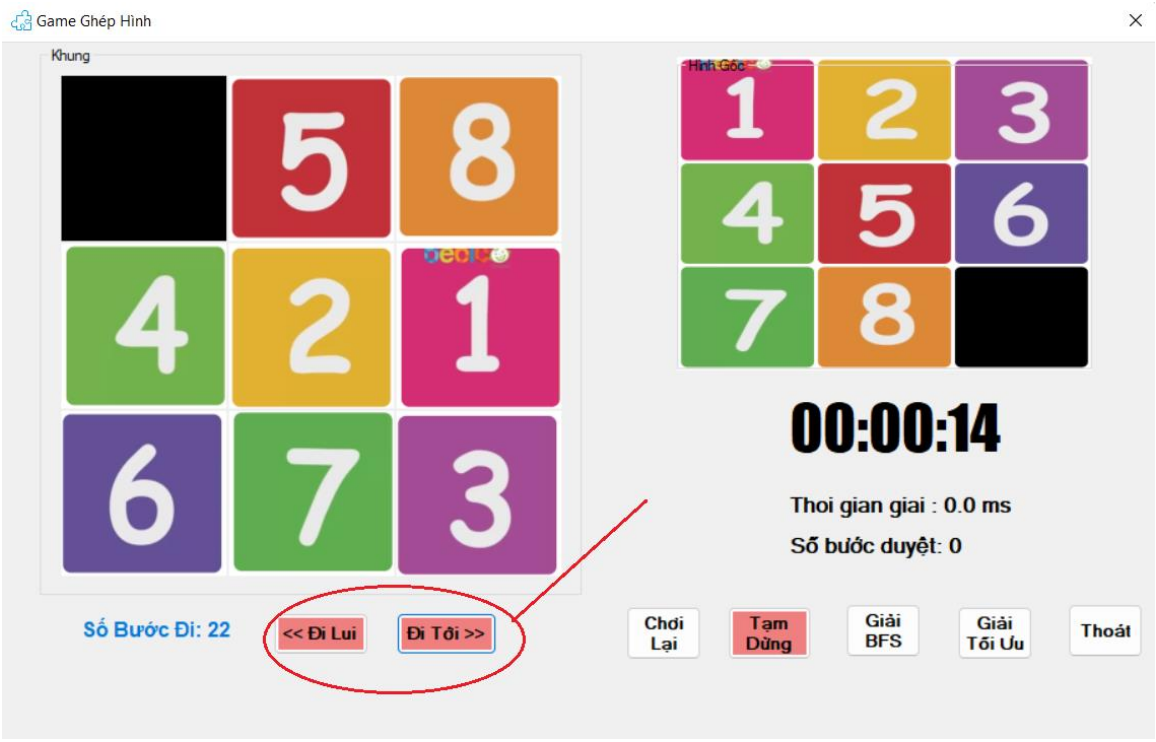


- Khi bạn bấm vào nút giải BFS hoặc giải tối ưu thì máy tính sẽ mất một khoảng thời gian để duyệt số bước đi. Cho tới khi đã giải xong phần thông tin về hai thứ vừa nêu sẽ được hiển thị tại đây.

Chức năng thoát

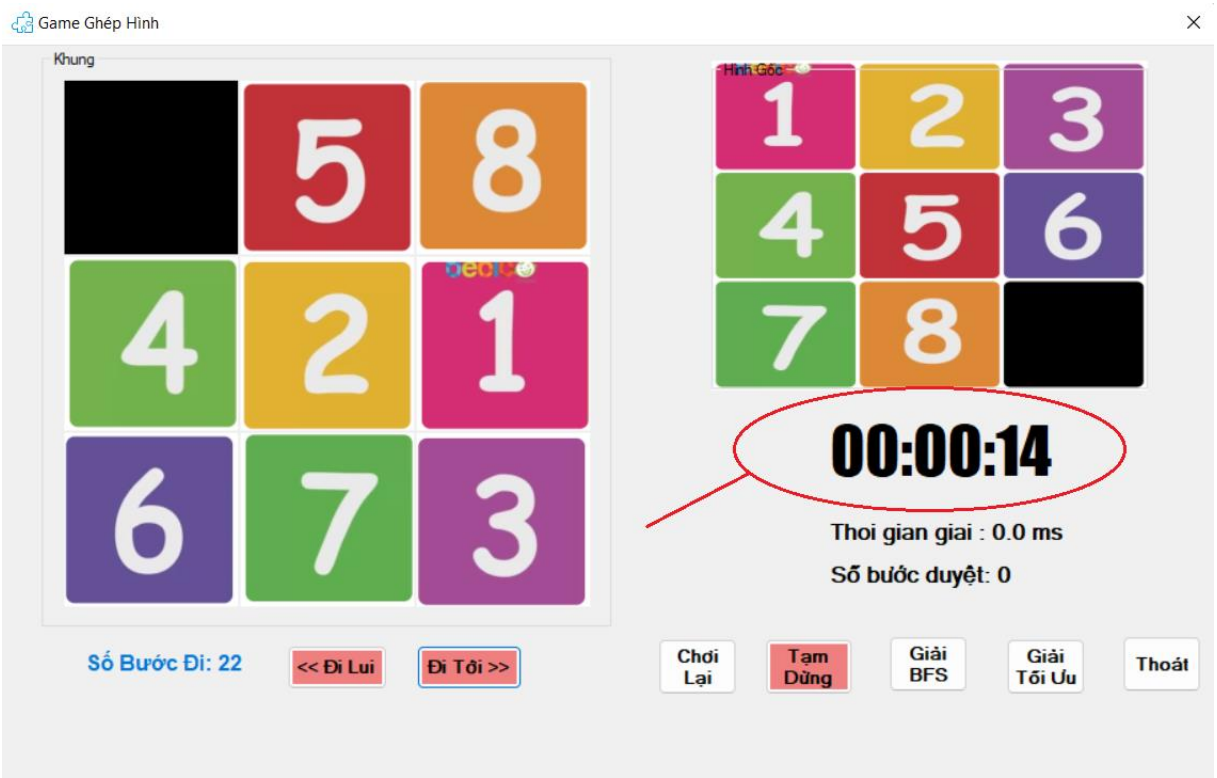
- Khi muốn tắt chương trình thì người dùng sẽ có chức năng thoát. Khí bạn click vào thì sẽ hiển thị một thông báo xác nhận là bạn có muốn thoát hay không.
- Người dùng chọn có thì chương trình sẽ thoát còn chọn không thì chương trình vẫn sẽ tiếp tục.

Chức năng đi lui và đi tới



- Khi bạn đã chọn chức năng giải BFS hoặc giải tối ưu chương trình đã thực thi xong thì bạn sẽ chọn chức năng đi tới và đi lui để đi từng bước tới trạng thái kết thúc.
- Chức năng này giúp người chơi có thể hiểu được các bước giải của máy thông qua việc đi từng bước từng bước.
- Khi đã đi hết tất cả các bước thì bạn sẽ tới trạng thái kêt thúc.

Chức năng tính thời gian



- Khi bắt đầu vào trò chơi thì thời gian vẫn sẽ chưa được tính, cho đến khi bạn bắt đầu di chuyển trạng thái đầu tiên của một ô trên khung thì thời gian sẽ bắt đầu tính. Hiểu nôm na sẽ là bạn di chuyển một ô thì thời gian sẽ bắt đầu tính kể cả khi sau đó bạn không di chuyển thêm một ô nào khác thì thời gian vẫn sẽ tính.
- Chức năng tính thời gian cũng sẽ là một chức năng đánh giá khả năng của bạn tương tự như đếm bước đi của bạn vậy. Bạn hoàn thành trong thời gian càng ngắn thì chứng tỏ bạn rất thông minh và chơi game này rất tốt.
- Thời gian sẽ dừng khi trạng thái bạn hoàn thành sẽ là trạng thái kết thúc. Hay nói cách khác là bạn đã xếp đúng hình thì thời gian sẽ dừng lại.

CHƯƠNG IV: KẾT LUẬN

4.1. 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…

4.2. 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… 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…

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