**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**

**KHOA CÔNG NGHỆ THÔNG TIN**

**\*\*\*\*\*\*\*\*\*\*\*\***

****

**BÁO CÁO BÀI TẬP HỌC PHẦN**

**MÔN: TRÍ TUỆ NHÂN TẠO**

**ĐỀ TÀI: Mini Game Ghép Tranh**

**Giảng viên hướng dẫn: Nguyễn Quốc Tuấn**

**Sinh viên thực hiện: Phạm Đức Khải**

**Đỗ Đức Anh**

**Trần Thị Thoa**

**Nguyễn Học Phú**

**Lớp: CNTT3\_K56**

**Hà Nội, 2018**

**Lời nói đầu**

Trong thời kì công nghiệp hóa hiện đại hóa của thế giới nói chung và đất nước nói riêng. Đặc biệt đang trong thời kì của cuộc cạch mạng công nghệ 4.0 khi các công nghệ mới liên tục được ra mắt đặc biệt máy móc dần thay thế con người trong các hoạt động lao động sản xuất và những con robot ngày càng thông minh hơn để làm được không thể thiếu sự đóng góp to lớn và không thể thiếu của trí tuệ nhân tạo. Game là một ứng dụng nhỏ của trí tuệ nhân tạo giúp giải quyết bài toán ghép tranh.

Tài liệu để thể hiệu thiết kế và cách giải quết bài toán của “Mini game ghép tranh” sử dụng thuật toán A\* mà nhóm đã tìm hiểm và thiết kế. Tài liệu này giúp người đọc có cái nhìn toàn vẹn và hiểu rõ về chức năng của Game cũng như cách dùng thuật toán A\* để giải quyết bài toán này.

Nhóm thực hiện đề tài này nhằm mục đích xây dựng 1 mini game đơn giản mà dễ chơi phục vụ giải trí từ đó tìm hiểu về thuật toán A\* và rộng hơn là trí tuệ nhân tạo. Trong quá trình thực hiện đề tài không tránh khỏi những sai sót, nhóm rất mong được sự đóng góp ý kiến và đánh giá của thầy.

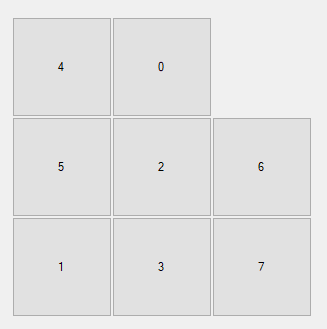
***Xin trân thành cảm ơn!***

1. **Bài toán ghép tranh**

Game ghép tranh là một game khá hay và trí tuệ nó được biết đến với nhiều phiên bản và tên gọi khác nhau như: Ghép hình, xếp hình, xếp tranh…. Bài toán ghép tranh là vấn đề cổ điển cho mô hình bài toán trí tuệ nhân tạo. Bài toán đặt ra là phải tìm đường đi từ trạng thái hiện tại đến trạng thái đích

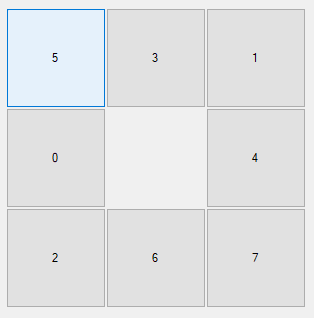
Phần mềm game ghép tranh là một chương trình trò chơi và giải quyết bài toán này. Phần mềm được viết trên ngôn ngữ C#, sử dụng giao diện đồ họa để mô phỏng trò chơi và thuật toán A\* để tìm đường đi. Người dùng sử dụng chuột để di chuyển, đổi chỗ ô hiện tại với ô trống nhằm đưa các ô tranh về đúng vị trí để tạo hình theo đúng mẫu. Hoặc có thể dùng chức năng tìm lời giải nhờ thuật toán A\*.

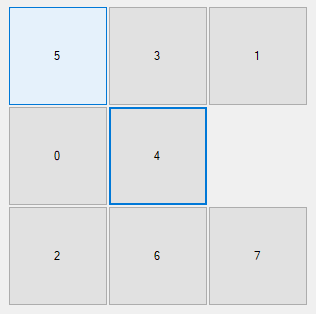
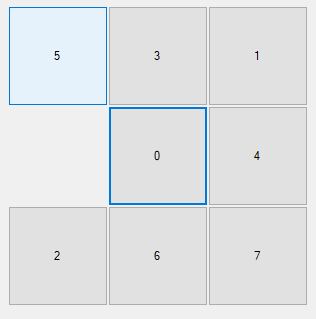
Yêu cầu xây dựng bảng ô vuông n hàng, n cột. Bảng gồm 1 ô trống và n 2 -1 ô chứa các số trong phạm vi [1, n 2 -1]. Xuất phát từ một cách xếp bất kì, di chuyển ô trống lên trên, xuống dưới, sang phải, sang trái để đưa các ô về trạng thái đích. Sửdụng chuột để di chuyển ô trống. Chương trình có chức năng tự động chơi ở bất kì trạng thái nào đó. Mỗi trạng thái của bảng số là một hoán vị của n 2 phần tử. Ở đây ta có thể thêm hình ảnh vào game hoặc gắn số vào hình ảnh để gợi ý cho người chơi. Ở trạng thái ban đầu, các ô được sắp xếp ngẫu nhiên, và nhiệm vụcủa người chơi là tìm được cách đưa chúng về trạng thái đích(ô cuối cùng bên phải trống, các ô khác theo thứ tự tăng dần từ trái qua phải, từ trên xuống dưới). Để đơn giản trong cách tiếp cận bài toán, ta giả định chỉ các ô trống di chuyển trong bảng là di chuyển đến các vị trí khác nhau. Như vậy tại một trạng thái bất kì có tối đa 4 cách di chuyển đến trạng thái khác(trái, phải, lên, xuống).

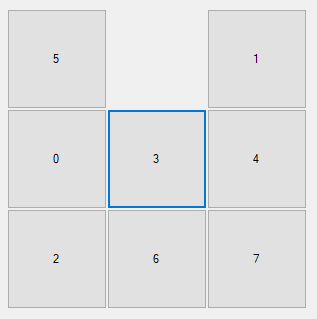
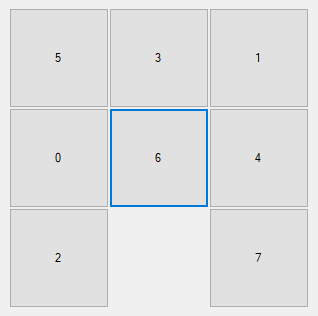
Trạng thái vào Trạng thái kết thúc

Bước di chuyển của ô trống



Phải Trái

Xuống Lên

1. **Thuật toán A\***
2. **Giới thiệu thuật toán**

Trong khoa học máy tính, A\* (đọc là A sao) là một thuật toán tìm kiếm trong đồ thị. Thuật toán này tìm một đường đi từ một nút khởi đầu tới một nút đích cho trước (hoặc tới một nút thỏa mãn một điều kiện đích). Thuật toán này sử dụng một "đánh giá heuristic" để xếp loại từng nút theo ước lượng về tuyến đường tốt nhất đi qua nút đó. Thuật toán này duyệt các nút theo thứ tự của đánh giá heuristic này. Do đó, thuật toán A\* là một ví dụ của tìm kiếm theo lựa chọn tốt nhất (best-first search).

Thuật toán A\* được mô tả lần đầu vào năm 1968 bởi Peter Hart, Nils Nilsson, và Bertram Raphael. Trong bài báo của họ, thuật toán được gọi là thuật toán A; khi sử dụng thuật toán này với một đánh giá heuristic thích hợp sẽ thu được hoạt động tối ưu, do đó mà có tên A\*.

Năm 1964, Nils Nilsson phát minh ra một phương pháp tiếp cận dựa trên khám phá để tăng tốc độ của thuật toán Dijkstra. Thuật toán này được gọi là A1. Năm 1967 Bertram Raphael đã cải thiện đáng kể thuật toán này, nhưng không thể hiển thị tối ưu. Ông gọi thuật toán này là A2. Sau đó, trong năm 1968 Peter E. Hart đã giới thiệu một đối số chứng minh A2 là tối ưu khi sử dụng thuật toán này với một đánh giá heuristic thích hợp sẽ thu được hoạt động tối ưu. Chứng minh của ông về thuật toán cũng bao gồm một phần cho thấy rằng các thuật toán A2 mới là thuật toán tốt nhất có thể được đưa ra các điều kiện. Do đó ông đặt tên cho thuật toán mới là A \*(A sao, A-star).

Năm 1964, Nils Nilsson phát minh ra một phương pháp tiếp cận dựa trên khám phá để tăng tốc độ của thuật toán Dijkstra. Thuật toán này được gọi là A1. Năm 1967 Bertram Raphael đã cải thiện đáng kể thuật toán này, nhưng không thể hiển thị tối ưu. Ông gọi thuật toán này là A2. Sau đó, trong năm 1968 Peter E. Hart đã giới thiệu một đối số chứng minh A2 là tối ưu khi sử dụng thuật toán này với một đánh giá heuristic thích hợp sẽ thu được hoạt động tối ưu. Chứng minh của ông về thuật toán cũng bao gồm một phần cho thấy rằng các thuật toán A2 mới là thuật toán tốt nhất có thể được đưa ra các điều kiện. Do đó ông đặt tên cho thuật toán mới là A \*(A sao, A-star).

1. **Mô tả thuật toán**

Giả sử n là một trạng thái tới(có đường đi từ trạng thái ban đầu n0 tới n). Ta xác định hàm đánh giá: f(n) = g(n) + h(n)

* f(n) là chi phí ước lượng tổng thể đường đi qua nút n đến nút đích
* g(n) là chi phí từ nút gốc n0 đến nút hiện tại n
* h(n) là chi phí ước lượng từ nút n đến nút đích (\*)

1. **Cài đặt thuật toán**

Danh sách OPEN chứa những trạng thái kết tiếp có thể xét

Danh sách CLOSE chứa những trạng thái đã được xét

1. Đặt OPEN chỉ chứa n0, CLOSE rỗng, f(n) = g(n) = h(n) = 0
2. Lặp lại đến khi gặp điều kiện dừng

* Chọn ni trong OPEN sao cho f(ni) là min

Lấy ni ra khỏi OPEN đưa vào trong CLOSE

* Nếu ni là đích thì lấy danh sách các nước đi đưa vào trong STACK
* Nếu ni không phải là đích thì

Xét các trạng thái tiếp theo của ni gọi là nk với mỗi nk thực hiện các bước sau:

+ Tính g(nk) = g(ni) + 1

h(nk) = (\*)

f(nk) = g(nk) + h(nk)

+ Đặt parent (nk) = ni

Nếu nk chưa xuất hiện trong OPEN và CLOSE thì thêm nk vào OPEN

(\*) Hàm h(nk) được tính như sau:

d = |rows - rowd|2 + |cols - cold|2

h(nk) = Tổng(d) + a

Trong đó: rows là vị trí dòng hiện tại của miếng ghép

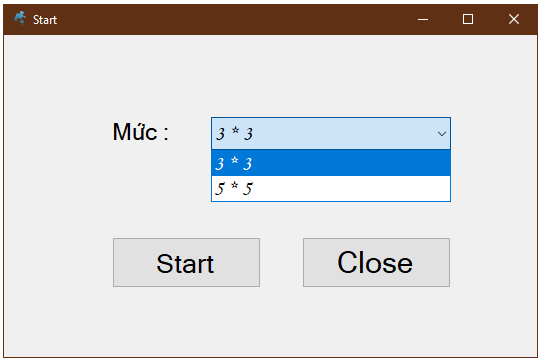
rowd là vị trí dòng đúng của miếng ghép

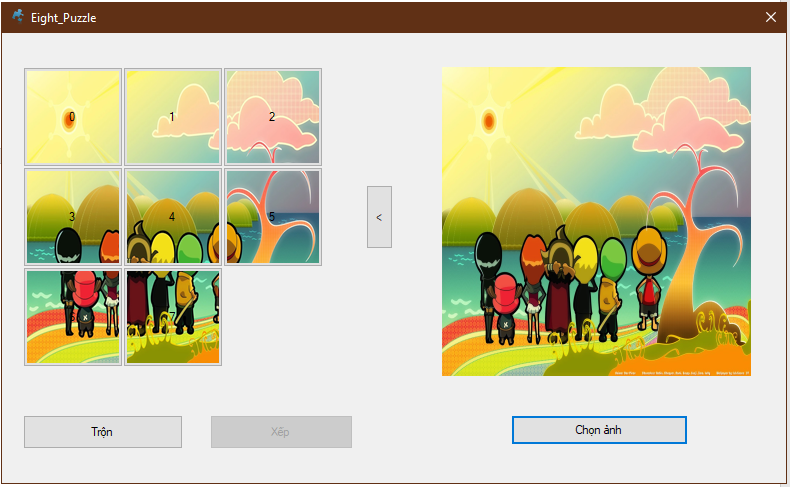
cols là vị trí cột hiện tại của miếng ghép

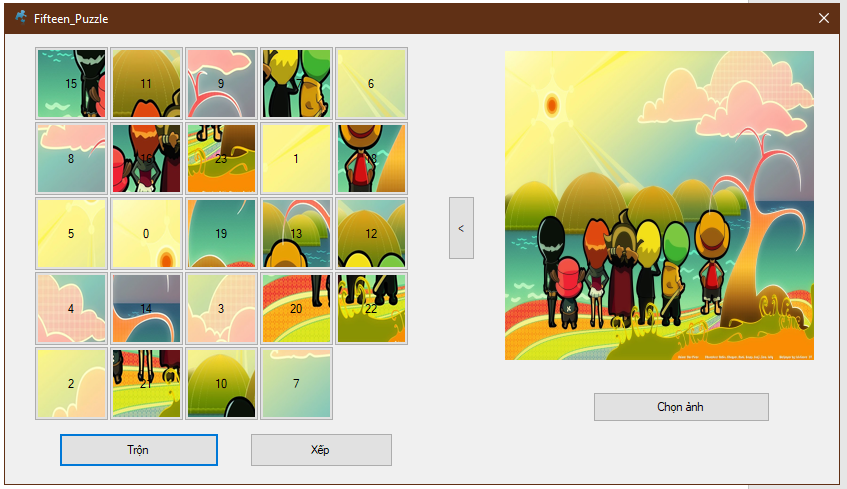
cold là vị trí cột đúng của miếng ghép

a là số cặp hàng xóm đang nằm ngược nhau

1. **Kết quả**
2. **Giao diện**







**Kết luận**

Trạng thái của một bảng số là một hoán vị của n2 phần tử(với n là kích thước của cạnh). Như vậy không gian trạng thái là n2! Với 3x3 là 9! Với 5x5 là 25! Khi kích thước cạnh tăng lên thì không gian trạng thái sẽ lên rất rất nhanh. Điều này khiến cho việc giải quyết bài toán n > 3 mất rất nhiều thời gian và ít được áp dụng.

Vậy nên trong bài việc tự xắp xếp chỉ mang tính chất gợi ý vì thời gian giải quyết bài toán 5x5 là quá lâu.

**Tài liệu tham khảo**

* Bài giảng môn trí tuệ nhân tạo – Thầy Nguyễn Quốc Tuấn
* [https://vi.wikipedia.org/wiki/Gi%E1%BA%A3i\_thu%E1%BA%ADt\_t%C3%ACm\_ki%E1%BA%BFm\_A\*](https://vi.wikipedia.org/wiki/Gi%E1%BA%A3i_thu%E1%BA%ADt_t%C3%ACm_ki%E1%BA%BFm_A*)
* <https://github.com/A2andil/Puzzle?fbclid=IwAR0g4BjqmnI5z5HaDrWGulaSO2icp6lw7O40rSqJahOYd_JkuHM07s3gTa4>
* <https://www.geeksforgeeks.org/check-instance-15-puzzle-solvable/>