**TRÒ CHƠI 8-PUZZLE MÁY VỚI NGƯỜI**

**Nguyễn Khoa Đảnh1, Bùi Trung Hiếu1**

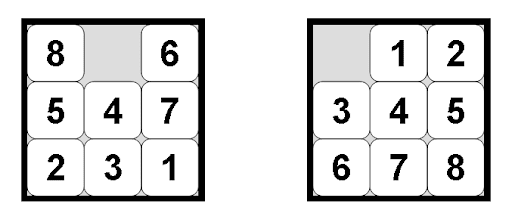
*1Trường đại học Sư phạm Kỹ thuật TP.HCM*

**TÓM TẮT**

Ở thời điểm hiện tại, AI (trí tuệ nhân tạo) đang được áp dụng vào rất nhiều lĩnh vực như các ngành khoa học, công nghiệp và đặc biệt được ứng dụng rất nhiều trong “nghành công nghiệp game” để tạo ra những sản phẩm thật chất lượng lập trình viên sẽ phải áp dụng AI vào trong sản phẩm của mình để có thể tạo ra những NPC hay gọi là “máy” để có thể tương tác với người chơi và cụ thể sau đây là game 8 puzzle nhóm em sẽ áp dụng AI để đưa các hướng đi để cho “máy” có thể chơi được với người. Đồ án áp dụng các kiến thức đã học, một số ràng buộc được tìm ra trong quá trình phát triển cùng với sử dụng tư duy của môn trí tuệ nhân tạo để giải thuật toán.

1. **GIỚI THIỆU**

Tìm hiểu bài toán 8 puzzle, tìm hiểu cách chơi để viết chương trình AI cho máy chơi với người. Phải tìm ra thuật toán giúp AI tìm được đường đi nhanh nhất và ít tốn chi phí nhất.



***Hình 1.*** *Bài toán 8 puzzle*

1. **PHƯƠNG PHÁP**

**2.1. Đơn giản hóa bài toán**

Số hóa 1 table về dạng key là 1 số int. VD: Hình 1 vế trái key = 806547231 , vế phải key = 12345678. Việc làm này giúp việc so sánh 2 trạng thái bằng nhau hay không nhanh hơn rất nhiều. Ít tốn bộ nhớ hơn và dễ dàng cài đặt trò chơi hơn.

**2.2. Viết hàm khởi tạo trạng thái bắt đầu Start và trạng thái đích Goal**

Trò chơi có trạng thái đầu (Start state) và trạng thái đích (Goal state).

Có 3 phương án để khởi tạo 2 trạng thái:

Người chơi đưa vào 2 mã key của trạng thái bắt đầu và trạng thái kết thúc, nếu mã key hợp lệ thì máy sẽ nhận, không thì nhập lại. Người chơi đưa vào mã key của trạng thái đích , máy sẽ hiển thị table để người chơi di chuyển theo ý muốn để xác định trạng thái bắt đầu. Máy random ra trạng thái ban đầu, sau đó sẽ tự di chuyển X lần với Heuristic càng lớn càng tốt (X có thể do người chơi thay đổi)

**2.3. Áp dụng các Thuật toán tìm kiếm để xác định luôn lối đi từ Start đến Goal**

Áp dụng 2 thuật toán Astar và BFS để bổ trợ lẫn nhau vì theo ý tưởng Astar sẽ tìm đường đi nhanh nhất còn BFS sẽ tìm đường đi tối ưu nhất (Lí do là ở bài toán 8puzzle này, phí di chuyển là 1). Để tối ưu cả 2 thuật toán:Sử dụng cấu trúc cây để lưu trữ các key đã qua sử dụng => khi duyệt key đã dùng nhanh hơn.Thêm hàm kiếm tra xem hướng đi có lặp lại trạng thái cũ không để nhanh chóng bác bỏ hướng đi đấy. Để tối ưu Astar, thuật toán Heuristic dùng ở đây là tổng khoảng cách lệch của mỗi ô nhân với giá trị của ô đó, ý tưởng việc này chính là giúp thuật toán có chiều hướng giải các ô lớn trước sau đó mới giải các ô bé, giảm việc thuật toán chạy theo nhiều hướng , tối ưu thời gian tới đích. Sau khi test nhiều trường hợp thì thấy kết quả rất khả quan.

**2.4. Mã giả của các thuật toán và phương pháp được sử dụng**

**Mã giả Astar:**

If key của start = key của goal

Kết thúc;

Tạo node list rank (tăng dần theo cost);

Tạo node tree used ;

Thêm start vào rank ;

Chạy vòng lập trong 3s

Lấy run = node đầu tiên trong rank;

Xét từng hướng có thể di chuyển của run và không trái ngược với hướng đi tới run

Tạo child với hướng đi đó ;

Nếu child không tồn tại trong used:

Nếu key của child = key của goal:

Đã tìm thấy ;

Nhảy tới Point ;

Nếu không :

Tính heuristic và cost của child tiến tới goal;

Thêm child vào rank ;

Nếu không :

Hủy child ;

Thêm run vào used ;

Point :

Nếu đã tìm thấy :

Chạy BFS để tối ưu;

Nếu BFS chạy thất bại thì kết nối từ start tới goal = Astar ;

Xóa node list rank ;

Xóa node tree used ;

Trả về true nếu tìm được hoặc ngược lại;

**Mã giả BFS:**

If key của start = key của goal

Kết thúc;

Tạo node list rank

Tạo node tree used ;

Thêm start vào rank ;

Chạy vòng lập tới độ sâu 30

Lấy run = node đầu tiên trong rank;

Xét từng hướng có thể di chuyển của run và không trái ngược với hướng đi tới run

Tạo child với hướng đi đó ;

Nếu child không tồn tại trong used:

Nếu key của child = key của goal:

Đã tìm thấy ;

Nhảy tới Point ;

Nếu không :

Thêm child vào cuối rank ;

Nếu không :

Hủy child ;

Thêm run vào used ;

Point :

Nếu đã tìm thấy thì kết nối từ start tới goal = Astar ;

Xóa node list rank ;

Xóa node tree used ;

Trả về true nếu tìm được hoặc ngược lại;

**2.4. Cài đặt Game:**

Thiết kế giao diện game thân thiện với người dùng với đầy đủ các tính năng (New game, Continue Game, Option , Exit). Option với 3 mức độ easy(7), medium(15) , hard(30) để máy khởi tạo start và goal với X là số trong ngoặc. Cũng có thể chọn custom với 2 phương án khởi tạo còn lại , hoặc là X là con số bạn muốn. Để người dùng có khả năng chơi game này, hỗ trợ việc xem hướng dẫn lối đi. Đồng xu với tỉ lệ úp ngửa 50/50. Bắt đầu chơi, máy và bạn sẽ thay phiên nhau lắc xu, người nào lắc được ngửa sẽ được đi và lắc tiếp. Nếu đi khác hướng tối ưu mà máy đã tìm ra, hướng đi đó sẽ được cập nhật vào bộ nhớ. khoảng cách giữa start và goal sẽ dài ra. Nếu đi đúng hướng tối ưu mà máy đã tìm ra, khoảng cách giữa start và goal sẽ ngắn lại. Khi đi tới nhưng bước cuối cùng máy sẽ lưỡng lự giữa việc đi tiếp tới đích hoặc đi rời xa đích để làm khó người chơi (dùng xác suất).Người đi tới goal trước sẽ thắng. Có thể thoát ra menu giữa trận, khi thoát ra menu như vậy thì vẫn có thể vào lại game khi nhấn continue game.

1. **KẾT QUẢ**

**So sánh giữa BFS, Astar với số ô sai(1), Astar với khoảng cách(1) và BFS gắn cây, Astar với heuristic mới.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Trường hợp | BFS | Astar (1) | Astar (2) | BFS gắn cây | Astar mới |
| Start:  563701842  Goal:  516873042 | Thời gian:  0.033000s  Số bước:6 | Thời gian:  0.032000s  Số bước:6 | Thời gian:  0.037000s  Số bước:6 | Thời gian:  0.000029s  Số bước:6 | Thời gian:  0.000021s  Số bước:6 |
| Start:  761324508  Goal:  276345801 | Thời gian:  1.961000s  Số bước:16 | Thời gian:  2.143000s  Số bước:16 | Thời gian:  2.146000s  Số bước:16 | Thời gian:  0.005023s  Số bước:6 | Thời gian:  0.000093s  Số bước:16 |
| Start:  837615240  Goal:  175684023 | Thời gian:  1.259000s  Số bước:16 | Thời gian:  1.382000s  Số bước:16 | Thời gian:  1.418000s  Số bước:16 | Thời gian:  0.005497s  Số bước:16 | Thời gian:  0.000079s  Số bước:16 |
| Start:  012345678  Goal:  123456780 | Thời gian:  >5p  Số bước:?? | Thời gian:  >5p  Số bước:?? | Thời gian:  >5p  Số bước:?? | Thời gian:  0.100696s  Số bước:22 | Thời gian:  0.000904s  Số bước:22 |
| Start:  841603257  Goal:  182076543 | Thời gian:  >5p  Số bước:?? | Thời gian:  >5p  Số bước:?? | Thời gian:  >5p  Số bước:?? | Thời gian:  0.557650s  Số bước:27 | Thời gian:  0.098050s  Số bước:29 |

1. **KẾT LUẬN**

Thuật toán tìm kiếm cũ thì thời gian sinh trường mới càng chạy càng dài , dẫn đến việc ảnh hưởng hiệu suất tìm kiếm rất lớn. Sau khi được áp dụng cấu trúc cây và hàm nhận biết để không đi ngược tới trạng thái cũ ở hàm BFS và áp dụng

heuristic mới ở hàm Astar thì ta thấy sự cải thiện rõ rệt về thời gian tìm kiếm. Gần như mọi trường hợp của 8puzzle nếu không phải trường hợp lỗi đều có thể giải được trong tích tắc. Game hầu như không gặp trường hợp không tìm được đường về đích.

**TÀI LIỆU THAM KHẢO**

[1]:<https://vi.wikipedia.org/wiki/T%C3%ACm_ki%E1%BA%BFm_theo_chi%E1%BB%81u_r%E1%BB%99ng>

[2]:[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*)

[3] <https://stackoverflow.com/questions/32983617/improve-8-puzzle-using-bfs>