# GIẢI BÀI TOÁN 8 PUZZLE TRONG TRƯỜNG HỢP CÁC Ô BỊ CHE CHỈ CÓ MỘT Ô ĐỂ HIỂN THỊ GIÁ TRỊ

**Phùng Vĩnh Đức1 18110100**

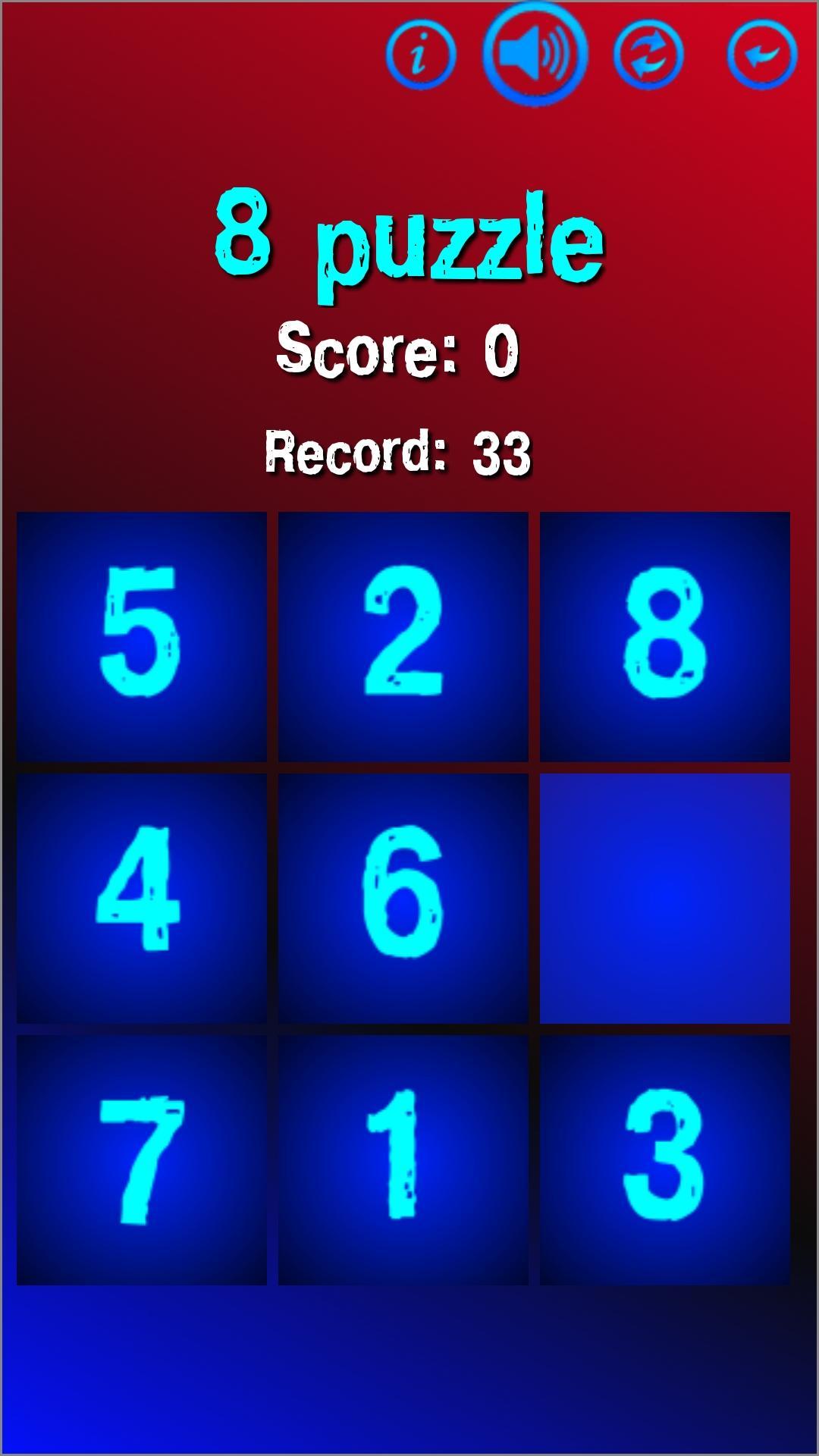
**Đào Hồng Vinh1 18110209**

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

**TÓM TẮT**

Game 8-Puzzle là game khá cơ bản mục đích của trò này là từ một trạng thái đầu vào gồm 9 ô được đánh số thứ tự ngẫu nhiên chúng ta sẽ sắp xếp lại thứ tự của các ô sao cho trùng với trạng thái đích cho trước sẽ chiến thắng. Puzzle có thể tồn tại nhiều kích cỡ khác nhau, phổ biến nhất là 8-puzzle với số ô là 3x3 hay 15-puzzle với kích cỡ 4x4 và có thể là n-puzzle. Trong đồ án này nhóm chúng em sử dụng 8-puzzle để thực hiện bài toán. Vậy vấn đề đặt ra là làm thế nào để có hướng di chuyển các ô sao cho hiệu quả và từ trạng thái đầu vào cho trước có thể về trạng thái đích nhanh nhất, ít bước di chuyển nhất. Từ đó chúng em vận dụng các thuật toán tìm kiếm trong môn trí tuệ nhân tạo để tìm lời giải cho bài toán. Trạng thái đầu vào của bài toán sẽ được sinh tự động và bị che lại chỉ hiện thị giá trị ô chính giữa. Sau đó sẽ tiến hành mở các ô trống để xác định được trạng thái đầu vào từ đó nhóm em dùng các thuật toán tìm kiếm để tìm lời giải cho bài toán. Nhóm chúng em sử dụng 2 thuật toán là Breadth-First Search và Astar để giải bài toán 8-puzzle từ đó so sánh thời gian chạy và số bước di chuyển đến trạng thái đích của mỗi thuật toán.

**Từ khóa:** 8-Puzzle, Breadth-First Search, Astar



**Hình 1**: Giao diện game 8 puzzle

1. **ĐẶT VẤN ĐỀ**

8-puzzle là một vấn đề khá cổ điển để mô hình hóa, ứng dụng các thuật toán liên quan đến phương pháp phỏng đoán. Vấn đề đặt ra là từ trạng thái đầu vào với các ô bị đặt sai vị trí thì làm sao để di chuyển các ô sai vị trí về đúng với trạng thái đích một cách nhanh nhất và ít bước di chuyển nhất. Trong những năm gần đây, ngành học trí tuệ nhân tạo (AI) đang phát triển mạnh và có tác động lớn trong xã hội con người. AI đang dần thay thế con người trong nhiều lĩnh vực trong cuộc sống. Và việc tìm hiểu bài toán 8-puzzle là một trong những ứng dụng của AI trong việc tìm ra trạng thái đích.

1. **PHƯƠNG PHÁP THỰC HIỆN**

2.1) Tìm giá trị đầu vào

Chúng ta tiến hành xác định giá trị các ô của trạng thái đầu vào bằng cách di chuyển các ô chưa biết giá trị về ô trống chính giữa khi đó giá trị cần biết của ô trống sẽ xuất hiện ở ô chính giữa. Sau khi xác định được trạng thái bắt đầu thì chúng em sẽ dùng các thuật toán tìm kiếm để giải bài toán 8 puzzle này với trạng thái đích được cho trước.

2.2) Sử dụng các thuật toán tìm kiếm

Chúng ta dùng các giải thuật tìm kiếm như BFS, DFS, A Star, Hill Climbing… để tìm trạng thái đích cho bài toán 8 puzzle. Tùy thuộc vào thuật toán tìm kiếm thì số bước di chuyển và thời gian để tìm được trạng thái đích sẽ rút gọn hơn. Ở đây nhóm em dùng thuật toán BFS(Breadth First Search) để giải bài toán 8 puzzle này.

Thuật toán BFS(u)

Bước 1(Khởi tạo):

Queue = ; Push(Queue,u); chuaxet[u] = False;

Bước 2 (Lặp):

While(Queue ≠ ) do

S = Pop(Queue); <Thăm đỉnh S>

For each t ϵ Ke(S) do

If (chuaxet[t]) then

Push(Queue,t) ; chuaxet[t] = False

End If;

End For

End While

Bước 3 (Trả lại kết quả)

Return (Tập đỉnh được duyệt)

End.

**Hình 2**: Mã giả thuật toán BFS

Thuật toán BFS của nhóm em hoạt động như sau:

Ban đầu em sẽ lưu trạng thái bắt đầu vào hàng đợi Queue, một vòng lặp kiểm tra nếu Queue khác rỗng thì tiếp tục lặp trong vòng lặp em kiểm tra nếu trạng thái bắt đầu bằng trạng thái goal thì dừng và đưa ra kết quả còn không thì:

Tại trạng thái ban đầu lấy vị trí hiện tại của ô trống từ đó di chuyển đi lên hoặc xuống hoặc qua trái hoặc qua phải để tạo trạng thái mới. Rồi lưu trạng thái mới sinh vào hàng đợi Queue rồi từ đó sinh trạng thái tiếp theo. Và vòng lặp như vậy cứ tiếp tục đến khi tìm ra trạng thái đích. Tại mỗi lần sinh trạng thái mới em sẽ kiểm tra trạng thái đó có được duyệt lần nào chưa nếu chưa mới đưa vào hàng đợi Queue.

2.3) Thuật toán Astar

1. Gọi Open: Tập các trạng thái đã được sinh ra nhưng chưa được xét

2. Close: Tập các trạng thái đã được xét đến

3. Cost(p, q): là khoảng cách giữa p và q

4. g(p): khoảng cách từ trạng thái đầu tiên đến trạng thái hiên tại p

5. h(p): giá trị được lượng giá từ trạng thái hiện tại đến trạng thái đích.

6. f(p) = g(p) + h(p)

Bước 1:

Open: = {S}

Close: = {}

Bước 2: While (Open != {})

+ Chọn trạng thái (đỉnh) tốt nhất p trong Open (xóa p khỏi Open)

+ Nếu p là trạng thái tốt nhất thì thoát

+ Chuyển p qua Close và tạo ra các trạng thái kế tiếp q sau p

* Nếu q đã có trong Open
  + Nếu g(q) > g(p) + Cost(p, q)
    - g(q) = g(p) + Cost(p, q)
    - f(q) = g(q) + h(q)
    - prev(q) = p // đỉnh cha của q là p
* Nếu q chưa có trong Open
  + g(q) = g(p) + Cost(p, q)
  + f(q) = g(q) + h(q)
  + prev(q) = p
  + Thêm q vào Open
* Nếu q có trong Close
  + Nếu g(q) > g(p) + Cost(q, p)
    - Bỏ q khỏi Close
    - Thêm q vào Open

Bước 3: Không tìm được

**Hình 3**: Mã giả thuật toán Astar

Để dùng được thuật toán Astar giải bài toán 8 puzzle thì nhóm em xây dựng một hàm Heuristic như sau:

Thứ tự ưu tiên được gán cho một đường đi p quyết định bởi hàm:

f(p) = g(p) + h(p) trong đó:

g(p) : khoảng cách từ trạng thái đầu tiên đến trạng thái hiện tại

h(p) : tổng khoảng cách các ô sai lệch của trạng thái hiện tại so với trạng thái đích

Thuật toán Astar sẽ lưu các đường dẫn có giá trị tốt đến đích trong hàng đợi. Khi tìm được trạng thái đích sẽ dựa vào các trạng thái cha để in ra tất cả các đường dẫn đến đích. Dựa vào path\_Cost được tính bằng hàm heuristic, thuật toán lưu các đỉnh sẽ xét trong 1 tập ưu tiên theo thứ tự giá trị path\_Cost từ bé đến lớn, tức là ở mỗi bước sinh đỉnh con, thuật toán sẽ tính toán chi phí dựa trên hàm heuristic để gán cho đỉnh con, sau đó cho đỉnh con vào hàng đợi ưu tiên giá trị path\_Cost bé hơn. Thuật toán dừng lại khi hàng đợi không còn phần tử nào để xét hoặc đã tìm thấy đích.

1. **SO SÁNH THUẬT TOÁN ASTAR VÀ THUẬT TOÁN BFS**

BFS là thuật toán tìm kiếm mù nó sẽ duyệt hết tất cả các trạng thái kề với trạng thái gốc và lưu vào hàng đợi. Nếu tìm thấy lời giải sẽ in ra các trạng thái đường đi dẫn tới đích. Tuy nhiên vì phải lưu tất cả các đường dẫn nên thuật toán chạy châm và mất thời gian, tốn bộ nhớ.

Astar là thuật toán tìm kiếm có hàm đánh giá Heuristic nên chỉ duyệt những trạng thái thỏa hàm đánh giá mà thôi. Vì vậy thời gian chạy sẽ nhanh và cũng tiêu tốn ít bộ nhớ hơn là hàm BFS. Và nếu như có lời giải thì thời gian tìm kiếm và đưa ra lời giải của thuật toán Astar dường như là tức thì.

1. **KẾT QUẢ**

Kết quả so sánh thời gian tìm ra lời giải cho bài toán 8-puzzle

* **Trường hợp 1:**

**Bảng 1**: So sánh thời gian tìm ra lời giải 1

|  |  |  |  |
| --- | --- | --- | --- |
| Thuật toán | Trạng thái đầu vào | Trạng thái đích | Thời gian thực thi(s) |
| BFS | 024318765 | 123456780 | 10.01 |
| Astar | 024318765 | 123456780 | 2.033 |

* **Trường hợp 2:**

**Bảng 2**: So sánh thời gian tìm ra lời giải 2

|  |  |  |  |
| --- | --- | --- | --- |
| Thuật toán | Trạng thái đầu vào | Trạng thái đích | Thời gian thực thi(s) |
| BFS | 725081346 | 123456780 | 5.11 |
| Astar | 725081346 | 123456780 | 0.477 |

Qua hai trường hợp so sánh ta thấy thời gian tìm được lời giải của thuật toán Astar nhanh

gấp nhiều lần thời gian tìm lời giải của thuật toán BFS.

1. **KẾT LUẬN**

Sau khi xác định được trạng thái đầu vào thì giải thuật BFS và Astar sẽ thay ta tính toán đường đi của ô trống đến trạng thái đích cho biết số bước đi và cách di chuyển ở từng bước của ô trống. Tuy nhiên nhược điểm của thuật toán BFS là thời gian chạy và tìm kiếm lời giải rất lâu. Còn đối với Astar thì thời gian chạy và tìm kiếm lời giải sẽ nhanh hơn. Điểm chung của 2 thuật toán là có một số trường hợp từ một trạng thái bắt đầu sẽ không bao giờ tìm đến đích.

1. **BẢNG PHÂN CÔNG**

|  |  |  |
| --- | --- | --- |
| Tên sinh viên | Mô tả công việc | Đóng góp |
| Đào Hồng Vinh | + Viết code cho phần hiển thị các ô bị che  + Viết code thuật toán Astar  + Kiểm tra bài báo cáo | 50% |
| Phùng Vĩnh Đức | + Viết code thuật toán Breadth First Search  + Viết code thuật toán Astar  + Viết báo cáo | 50% |

1. **TÀI LIỆU THAM KHẢO**

[1]: YinYang,”[Algorithm – Phân tích và giải bài toán n-puzzle](https://yinyangit.wordpress.com/2010/12/16/algorithm-phan-tich-va-gi%e1%ba%a3i-bai-toan-n-puzzle/)”, 2010

Available: <https://yinyangit.wordpress.com/2010/12/16/algorithm-phan-tich-va-gi%E1%BA%A3i-bai-toan-n-puzzle/>

[2]: [Ajinkya Sonawane](https://blog.goodaudience.com/@sonawaneajinks?source=post_page-----7b509c331288----------------------), “Solving 8-Puzzle using A\* algorithm”, 2018.

Available: <https://blog.goodaudience.com/solving-8-puzzle-using-a-algorithm-7b509c331288>.

[3]: Wikipedia,”Tìm kiếm theo chiều rộng”, 2020

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