ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

BÁO CÁO BÀI TẬP LỚN

MÔN TRÍ TUỆ NHÂN TẠO

ĐỀ TÀI: TRÒ CHƠI 2048

Giảng viên hướng dẫn: Ts. Nguyễn Nhật Quang

Nhóm sinh viên thực hiện:

* Nguyễn Tiến Thạo 20144161 CNTT 2.02
* Lê Bảo Chi 20140413 CNTT 2.04
* Đinh Hữu Hải Quân 20143632 CNTT 2.02
* Dương Trung Nghĩa 20143158 CNTT 2.02

Hà nội, 30 tháng 11 năm 2016

MỤC LỤC

**Lời nói đầu**

**PHẦN 1 : Giới thiệu**

* 1. Giới thiệu về đề tài
  2. Giới thiệu về Game 2048

**PHẦN 2: Thuật toán và hàm heuristic**

3.1 Hàm heuristic

3.2 Thuật toán

3.2.1 Thuật toán minimax

3.2.2 Thuật toán minimax với alpha-beta cắt tỉa

3.2.3 Thuật toán Expectimax

**PHẦN 3: Thống kê và nhật xét**

4.1. Thống kê giữa các thuật toán khác nhau

4.2 Thống kê các độ sâu khác nhau của cùng thuật toán

**PHẦN 4: Hướng dẫn sử dụng**

**PHẦN 5: Hạn chế và hướng phát triển**

**PHẦN 6: Tài liệu tham khảo**

**Lời nói đầu**

Được sự phân công và cho phép của giảng viên hướng dẫn –TS Nguyễn Nhật Quang nhóm chúng em đã chọn đề tại Game 2048 AI để làm bài tập lớn môn học Trí tuệ nhân tạo.

Để hoàn thành bài tập lớn này, nhóm em xin chân thành cảm ơn thầy giáo TS.Trần Nhật Quang đã cung cấp những kiến thức cơ bản hết sức bổ ích và đưa ra những hướng dẫn, lời khuyên và giải đáp cụ thể cho chúng em.

Mặc dù đã cố gắng thực hiện đề tài một cách hoàn chỉnh nhất, song do những kiến thức môn học là hoàn toàn mới mẻ và khả năng làm việc nhóm còn nhiều hạn chế nên sản phẩm của chúng em không thể tránh khỏi những thiếu sót nhất định mà bản thân nhóm chưa thấy được. Nhóm chúng em rất mong nhận được ý kiến phê bình, đóng góp của thầy để đồ án được hoàn chỉnh hơn.

Xin chân thành cảm ơn !

**Phần 1: Giới thiệu**

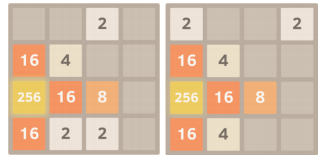
1.1 Giới thiệu về đề tài:

2048 là game được viết bởi Gabriele Cirulli vào tháng 3 năm 2014. Nó nhanh chóng được biết đến và chơi bởi hàng triệu người trên internet. Nhiều người đã tìm kiếm các thuật toán tối ưu để giải quyết vấn đề này. Dưới đây là một vài hướng tiếp cận thường được mọi người sử dụng là thuật toán minimax và expectimax. Ý tưởng chính của các thuật toán này là tính toán tất cả các hướng di chuyển có thể và chọn hướng di chuyển tốt nhất. Ngoài ra kỹ thuật cắt tỉa alpha-beta cũng thường được sử dụng để tăng hiệu suất cho thuật toán.

* 1. Giới thiệu về Game 2048:

Game 2048 là một game có cách chơi rất đơn giản. Bạn có một bảng 4x4, mỗi ô của bảng có thể có bao gồm một số ở bên trong. Những số đó sẽ luôn là bội số của 2. Ban đầu trò chơi xuất hiện với 2 ô có một trong hai giá trị 2 hoặc 4. Bạn chơi trò chơi bằng cách di chuyển theo một trong bốn hướng và các ô trên bảng 4x4 sẽ di chuyển theo hướng đó. Ví dụ nếu bạn nhấn lên trên tất cả các ô của bảng 4x4 sẽ di chuyển lên trên. Trong khi di chuyển nếu một ô gặp ô hàng xóm có cùng số thì hai ô sẽ trộn thành một ô. Sau mỗi lượt di chuyển sẽ có thêm một ô mới được sinh ra vào một trong các vị trí các ô trống trên bảng. Giá trị ô mới được sinh ra quy tắc sau. Ô số 2 được sinh ra với xác suất 0.9 và ô 4 được sinh ra với xác suất 0.1.

Để hiểu rõ hơn hãy xem ví dụ dưới đây:



Ví dụ một trò chơi 2048 khi di chuyển sang trái

Trò chơi sẽ chuyển từ hình bên trái sang hình bên phải

Mục đính của Game là đạt tới 2048 . Game cũng có thể mở rộng để đạt đến giá trị lớn nhất có thể.

**Phần 2: Thuật toán và hàm heuristic:**

2.1 Hàm heuristicScore:

Để có thể sử dụng các thuật toán minimax và expectimax chúng ta cần một hàm heuristic. Về cơ bản hàm heuristic dùng để xếp hạng các hướng di chuyển trong thuật toán. Giá trị này sẽ lớn hơn nếu di chuyển theo hướng đó là tốt hơn ( đạt điểm cao hơn).

Chúng em xây dựng hàm heuristic cho Game 2048 dựa trên quan sát cách chơi Game 2048 trong thực tế.

Chúng em thấy một số yếu tố ảnh hưởng quyết đinh một trạng thái game là tốt so với trạng thái khác như sau:

+ Số ô trống: Rõ rãng với một bảng mà càng nhiều ô trống thì sẽ dễ để chiến thắng hơn.

+ Sự chênh lệch: Mục đích của trò chơi là đạt đến 2048 mà để đạt được 2048 ta cần gộp hai ô 1024 cạnh nhau, để đạt 1024 ta cần hai ô 512 cạnh nhau cứ như vậy. Do vậy để hai ô nhỏ có thể gộp lại thành một ô lớn hơn nó sẽ phải ở cạnh nhau. Sự chênh lệch chúng em nói đến ở đây là sự khác nhau giữa điểm của một ô với tất cả các ô xung quanh nó. Giá trị này càng nhỏ thì trạng thái đó càng tốt.

+ Giá trị lớn nhất ở góc: Giá trị lớn nhất sẽ là giá trị có khả năng kết hợp với các giá trị khác ít nhất trên bảng nên nó nên được đặt ở góc.

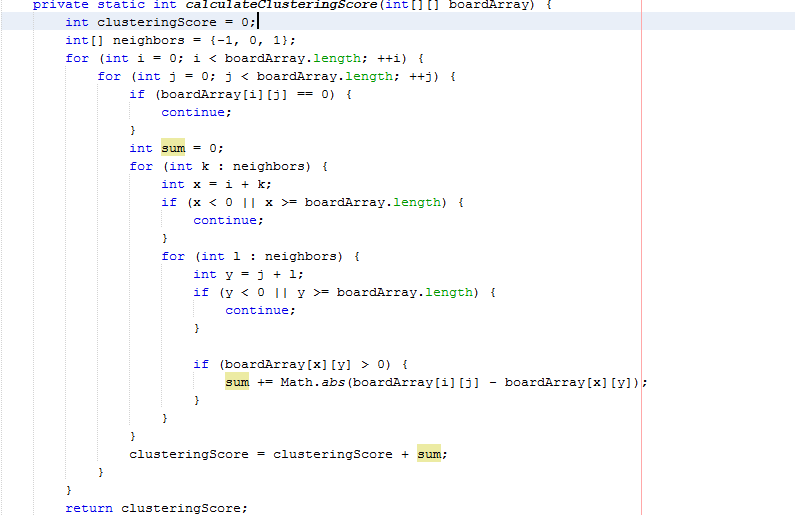
Trên cơ sở những quan sát trên chúng em xây dựng hàm heuristic như sau:

**calculateClusteringScore – calculatePositonScore + Math.log(gameScore)+numberOfEmpty\* numberOfEmpty**

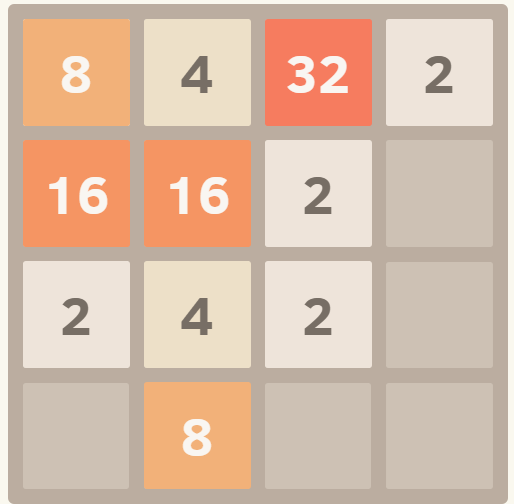
2.1.1 Hàm calculateClusteringScore:

Giá trị của hàm này được tính bằng cách duyệt tất cả các phần tử của mảng, mỗi phần tử tính điểm chênh lệch với tất cả các ô xung quanh nó và cộng tổng lại.

Hàm thực hiện :



Ví dụ minh họa:



Score=504

2.1.2 Hàm calculatePositonScore:

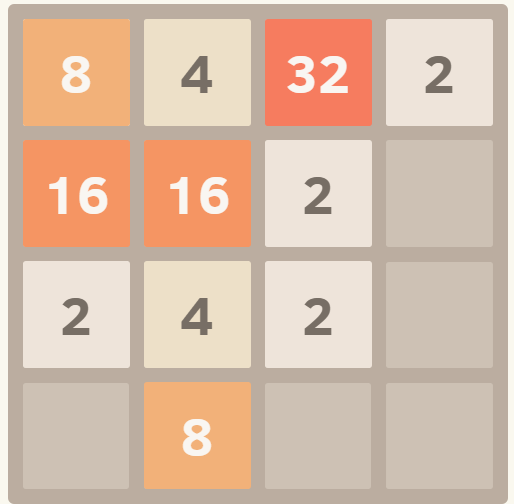
Giá trị của hàm này được tính bằng cách bình phương các phần tử của mảng lên rồi đem nhân với ma trận trọng số W.

|  |  |  |  |
| --- | --- | --- | --- |
| 6 | 5 | 4 | 1 |
| 5 | 4 | 1 | 0 |
| 4 | 1 | 0 | -1 |
| 1 | 0 | -1 | -2 |

Hàm này có tác dụng ưu tiên các xếp các phần tử có điểm cao vào các góc bên trái. Có nhiều cách để chọn bảng W này nhưng nhóm em thấy thực tế chơi thường đưa các số to vào góc nên chúng em chọn bảng W này.Một ma trận W tương tư nhưng qua thí nghiệm ở độ sâu 5 với thuật toán expectimax thì kết quả thấp hơn .

|  |  |  |  |
| --- | --- | --- | --- |
| 6 | 5 | 4 | 3 |
| 5 | 4 | 3 | 2 |
| 4 | 3 | 2 | 1 |
| 3 | 2 | 1 | 0 |

Ví dụ minh họa:

 Score=6904

2.1.3 Hàm heuristicScore:

Hàm này là hàm tổng hợp của hai hàm trên thêm vào đó là số ô trống và điểm cao nhất của bảng.

Giá trị hàm này tính bằng :

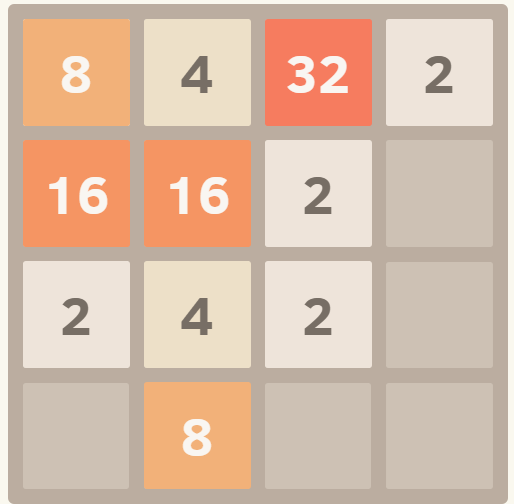
**calculatePositonScore –calculateClusteringScore + Math.log(gameScore)+numberOfEmpty\* numberOfEmpty**

Với :

+ gameScore là điểm số của trạng thái game

+numberofEmpty : là số ô trống ở trạng thái đó

Ví dụ minh họa:

 => Score=6467

2.2 Thuật toán:

2048 có thể được xem như một trò chơi hai người giữa người và máy tính. Trong đó người chơi sẽ di chuyển theo một trong bốn hướng còn máy tính sẽ đặt thêm một ô ngẫu nhiên 2 hoặc 4 vào các ô trống trên bảng trò chơi. Vì vậy mà chúng em sẽ sử dụng thuật toán minimax và expectimax cho trò chơi này .

Minimax và expectimax là những thuật toán để xác định hướng di chuyển tốt nhất trong một số trò chơi hai người.

2.2.1 Thuật toán minimax:

Minimax là một luật quyết định để tối thiểu hóa khả năng thua trong trường hớp xấu nhất. Đó là lý do tại sao nó được gọi mà minimax.

Lý thuyết của minimax:

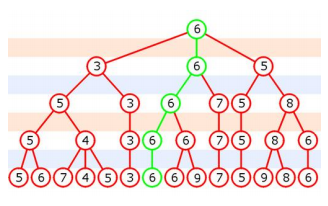
Với hai người , trò chơi tổng bằng không có hữu hạn chiến lược, tồn tại một giá trị V và một tập các chiến lược cho mỗi người chơi mà

+ Với chiến lược của người chơi thứ hai, Giá trị tốt nhất của người thứ nhất là V và

+ Với chiến lược của người thứ nhất, Giá trị tốt nhất của người thứ hai là -V.

Điều đó có nghĩa là chiến lược của người chơi thứ nhất đảm bảo rằng người đó phải đạt được giá trị V bất kể chiến lược của người còn lại có như thế nào.

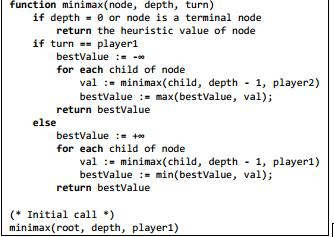
Đồ thị dưới đây giải thích tìm kiếm trên cây sử dụng thuật toán minimax:



Nút với độ sâu chẵn ( nút gốc có độ sâu là 0) là lượt chơi của người thứ nhất người mà muốn cực đại hóa giá trị. Nút với độ sâu lẻ là lượt chơi của người thứ hai. Với mỗi lượt đi của người thứ nhất chúng ta muốn cực đại hóa giá trị thu được, ngược lại với mỗi lượt đi của người thứ hai chúng ta muốn cực tiểu hóa giá trị thu được.

Giá trị thu được sẽ được khi ta đạt đến nút lá. Trong hầu hết trường hợp ta sẽ không thể đạt đến nút lá thực sự (nơi mà game kết thúc) do độ sâu quá lớn. Do đó thông thường thuật toán minimax sẽ được thực hiện với một đô sâu giới hạn thay vì vô hạn như depth-first search. Bởi vì chúng ta không biết chính xác giá trị của nút lá nên chúng ta sẽ tính toán ước lượng giá trị của nút đó sử dụng hàm heuristic. ( Đã được nêu ở phần trước).

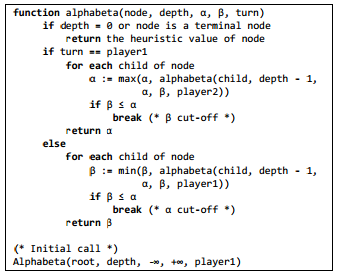
Mã giải thuật toán minimax :



Thuật toán minimax gốc yêu cầu mở rộng cây tìm kiếm điều này là cực kỳ tốn kém trong những vấn đề phức tạp. Chúng ta có thể cải thiện nó với alpha-beta cắt tỉa.

2.2.2 Thuật toán minimax với alph-beta:

Alpha-beta cắt tỉa sử dụng để cắt bớt số lượng nút trong cây đánh giá của thuật toán minimax. Thuật toán sẽ dừng hoàn toàn việc đánh giá một hướng di chuyển khi ít nhất có thể tìm thấy một hướng có thể chứng minh hướng đó là trường hợp tồi hơn hướng đã khảo sát trước đó. Mã giả của thuật toán cắt tỉa alph-beta như sau:

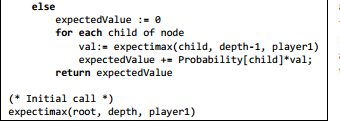
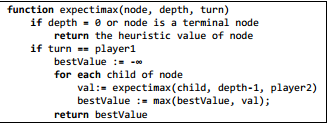


Alpha là cận dưới giá trị còn beta là cận trên giá trị. Tại lượt đi của người thứ nhất chúng ta muốn cực đại hóa giá trị thu được với mọi hướng di chuyển có thể. Giá trị của alpha được cập nhật sau mỗi di chuyển. Trong một số điểm ta thấy beta không lớn hơn alpha do vậy chúng ta sẽ dừng tìm kiếm ở đây bởi vì ta đã chứng minh được nó là hướng di chuyển tốt hơn. Đó gọi là beta cutoff. Hoàn toàn tương tự với alpha cutoff.

2.2.3 Thuật toán expectimax:

Trong thuật toán minimax chúng ta chọn hướng di chuyển dựa trên giá trị cực đại hoặc giá trị cực tiểu. Có nghĩa là chỉ cần giá trị lớn nhất hoặc nhỏ nhất của nút con. Trong thuật toán expextimax chúng ta sẽ đánh giá nút đối thủ, chúng ta sẽ đánh giá tất cả các nút có thể , tính theo khả năng xảy ra. Nói một cách khác chúng ta sẽ tính toán giá trị mong đợi trong tất cả các giá trị có thể thu được.

Để hiểu rõ hơn dưới đây là mã giải :



Trong hầu hết trường hợp xác xuất của mỗi nút là tương tự nhau. Với game 2048 xác xuất này đã biết trước. Với mỗi ô trống xác xuất để xuất hiện giá trị 2 là 0.9 và giá trị 4 là 0.1

**Phần 3: Thống kê và nhận xét:**

3.1 Thống kê giữa các thuật toán khác nhau:

|  |  |  |
| --- | --- | --- |
|  | Tỷ lệ thắng | Thời gian trung bình / game |
| Minimax | 508/1000 | 10115 s ~ 10s /game |
| Alpha-beta | 622/1000 | 1058 s ~ 1s/game |
| Expectimax | 675/1000 | 13488 s ~ 13.4 giây/ game |

Bảng 3.1.1 Thống kê về thời gian và tỷ lệ thắng của các thuật toán

3.2 Thống kê các độ sâu khác nhau của cùng thuật toán:

|  |  |  |
| --- | --- | --- |
| MINIMAX | Tỷ lệ thắng | Thời gian trung bình/game |
| 3 | 193/1000 | 0.186 giấy/game |
| 5 | 508/1000 | 10 giây / game |
| 7 |  |  |

Bảng 3.2.1 Thống kê thời gian và tỷ lệ thắng của thuật toán minimax theo độ sâu

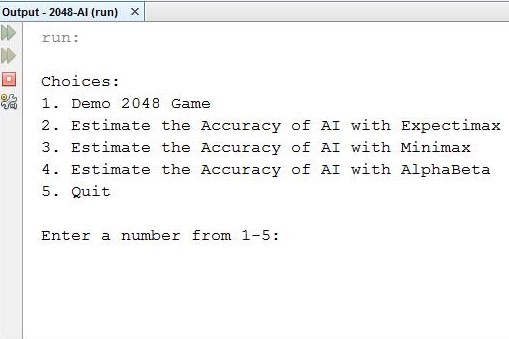
|  |  |  |
| --- | --- | --- |
| ALPHA-BETA | Tỷ lệ thắng | Thời gian trung bình/game |
| 3 | 319/1000 | 0.069 giây/game |
| 5 | 622/1000 | 1 giây / game |
| 7 | 700/1000 | 37.1 giây/game |

Bảng 3.2.2 Thống kê thời gian và tỷ lệ thắng của thuật toán alpha-beta theo độ sâu

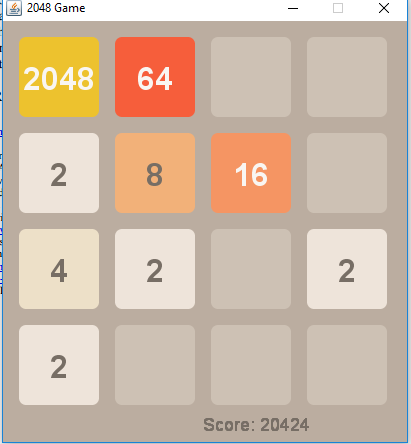
|  |  |  |
| --- | --- | --- |
| EXPECTIMAX | Tỷ lệ thắng | Thời gian trung bình/game |
| 3 | 350/1000 | 2.11 giây |
| 5 | 675/1000 | 13.4 giây |
| 7 | 880/1000 | 20 phút |

Bảng 3.2.3 Thống kê thời gian và tỷ lệ thắng của thuật toán Expectimax theo độ sâu

PHẦN 4: Hướng dẫn sử dụng:



Chức năng 1: Demo 2048 Game:



Chức năng 2,3 và 4: Thống kê 100 game về tỷ lệ thắng và thời gian với từng thuật toán.

Chức năng 5: Thoát.

PHẦN 5: Hạn chế và hướng phát triển:

5.1: Hạn chế:

Tỉ lệ thắng chưa cao.

5.2: Hướng phát triển:

PHẦN 6 : Tài liệu tham khảo:

1. Bải giảng Trí tuệ nhân tạo TS Nguyễn Nhật Quang
2. Russell, Stuart J.; Norvig, Peter (2010). Artificial Intelligence: A Modern Approach (3rd ed.). Upper Saddle River, New Jersey: Pearson Education, Inc.
3. “What is the optimal algorithm for the game, 2048?”. [http://stackoverflow.com/questions/22342854/what-is-the-optimalalgorithm-for-the-game-2048/22389702#22389702](http://stackoverflow.com/questions/22342854/what-is-the-optimalalgorithm-for-the-game-2048/22389702%2322389702).