TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

# 

BÀI TẬP LỚN TRÍ TUỆ NHÂN TẠO

ĐỀ TÀI: XÂY DỰNG GAME CỜ CARO SỬ DỤNG THUẬT TOÁN CẮT TỈA ALPHA - BETA

Sinh viên thực hiện: Bùi Hoàng Lưu - 20142776

Nguyễn Võ Linh -

Nguyễn Văn Đức -

Nguyễn Đình Đoàn -

Giáo viên hướng dẫn: Nguyễn Nhật Quang

**Hà Nội, ngày 5 tháng 12 năm 2016**

Mục lục

[**Sơ lược về trò chơi cờ caro**](#_hf3p6w4te38s)[**2**](#_hf3p6w4te38s)

[**Giải thuật MINIMAX**](#_yoxawaw0a1jl)[**2**](#_yoxawaw0a1jl)

[**Giải thuật cắt tỉa alpha - beta**](#_27yg6iymlast)[**3**](#_27yg6iymlast)

[**Phân tích, giải quyết bài toán**](#_kyhjb7gf2iox)[**4**](#_kyhjb7gf2iox)

[**Thử nghiệm**](#_ugkwuzrbpvqq)[**7**](#_ugkwuzrbpvqq)

[**Các đề xuất cải tiến hệ thống**](#_ffi49h2hgqnt)[**7**](#_ffi49h2hgqnt)

[**Kết luận**](#_h6ffb0ug4otj)[**7**](#_h6ffb0ug4otj)

[**Tài liệu tham khảo**](#_uesantzbgi17)[**8**](#_uesantzbgi17)

# Sơ lược về trò chơi cờ caro

Cờ caro là một trò chơi rất phổ biến trên thế giới. Trò chơi gồm 1 bàn cờ 15x15 ô và 2 quân cờ X, O.

Luật chơi:

* 2 người chơi, mỗi người chơi sẽ chọn một quân cờ.
* 2 người chơi sẽ lần lượt đánh quân cờ của mình.
* Người chiến thắng là người có 5 quân cờ liền nhau trên một hàng dọc, hàng ngang hoặc đường chéo và không bị chặn 2 đầu.
* Nếu có trên 5 quân cờ liên tiếp thì sẽ không được tính là thắng.
* Ván cờ hòa khi đã hết ô để đi mà vẫn chưa có người nào giành chiến thắng.

# Giải thuật MINIMAX

Giải thuật MINIMAX là giải thuật vét cạn để giải quyết bài toán tìm nước đi trong cờ caro:

Hai đối thủ trong trò chơi được gọi là MIN và MAX luân phiên thay thế nhau đi. MAX đại diện cho người quyết dành thắng lợi và cố gắng tối đa hóa ưu thế của mình, ngược lại người chơi đại diện cho MIN lại cố gắng giảm điểm số của MAX và cố gắng làm cho điểm số của mình càng âm càng tốt. Giả thiết đưa ra MIN và MAX có kiến thức như nhau về không gian trạng thái trò chơi và cả hai đối thủ đều cố gắng như nhau.

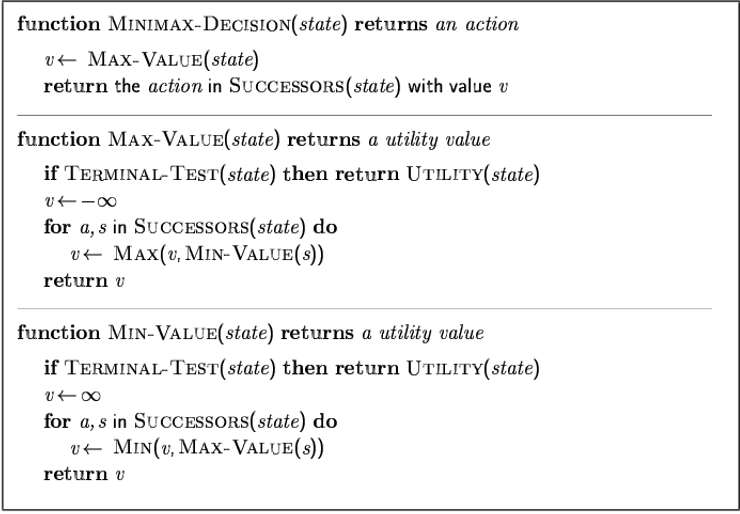
Mỗi node biểu diễn cho một trạng thái trên cây trò chơi. node lá là node chứa trạng thái kết thúc của trò chơi.

Giải thuật Minimax thể hiện bằng cách xét giá trị MINIMAX của các node trên cây trò chơi:

* Node thuộc lớp MAX thì gán cho nó giá trị lớn nhất của con node đó.
* Node thuộc lớp MIN thì gán cho nó giá trị nhỏ nhất của con node đó.
* Nếu là node lá (trạng thái kết thúc) thì giá trị của nó sẽ được tính bởi một hàm tiện ích (utility function)

Từ các giá trị này người chơi sẽ lựa chọn cho mình nước đi tiếp theo hợp lý nhất.

Giải thuật:



→ Độ phức tạp của giải thuật với độ sâu m và hệ số phân nhánh b

* Thời gian: O(bm)
* Bộ nhớ: O(bm)

# Giải thuật cắt tỉa alpha - beta

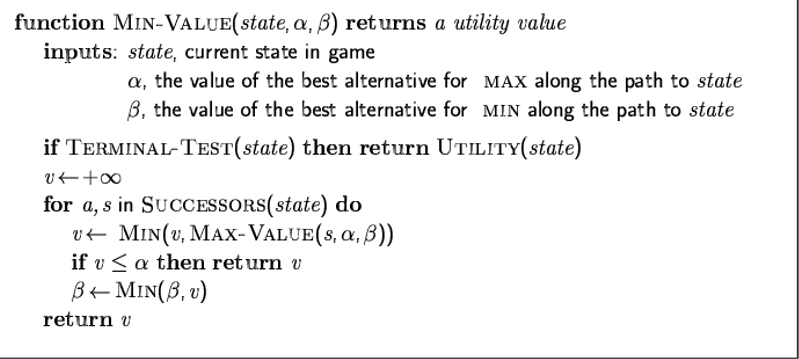
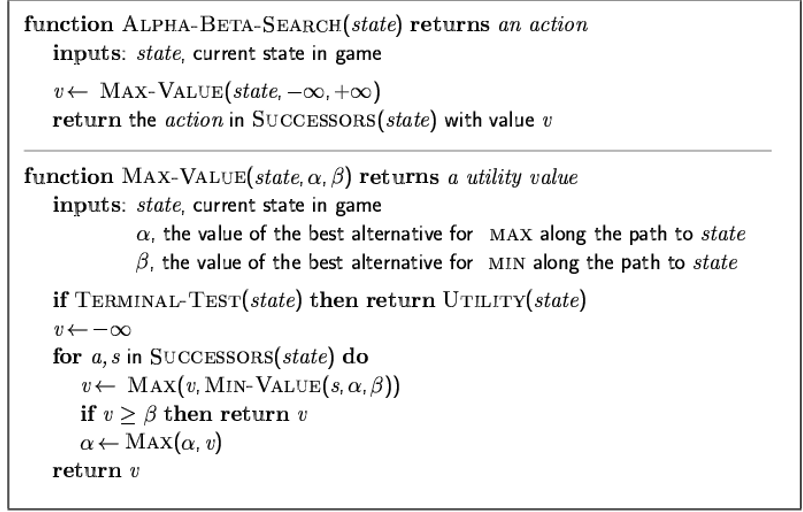
Giải thuật tìm kiếm MINIMAX sẽ dẫn đến bùng nổ (mức hàm mũ) các khả năng phải xét, không phù hợp với bài toán thực tế. Giải thuật cắt tỉa alpha - beta ra đời nhằm cải thiện giải thuật MINIMAX.

Ý tưởng: Nếu một nhánh tìm kiếm nào đó không thể cải thiện giá trị đối với hàm tiện ích mà chúng ta đã có thì không cần xét đến nhánh tìm kiếm đó nữa.

Cắt tỉa alpha - beta:

* Alpha là giá trị của nước đi tốt nhất đối với MAX (giá trị tối đa) tính đến thời điểm hiện tại đối với nhánh tìm kiếm.
* Nếu v là giá trị tồi hơn alpha, MAX sẽ bỏ qua nước đi tương ứng với v → cắt tỉa nhánh tương ứng với v.
* Tương tự với beta và MIN

Giải thuật:



# Phân tích, giải quyết bài toán

Giải thuật cắt tỉa alpha - beta đã giảm đi một số lượng lớn nhánh phải xét so với giải thuật MINIMAX nhưng trong các bài toán thực tế có không gian trạng thái lớn, cụ thể ở đây là trò chơi cờ caro 15x15 thì phương pháp cắt tỉa alpha - beta vẫn không phù hợp.

Để hạn chế hơn nữa không gian tìm kiếm, chúng em đã thêm vào trị thức thực tế về các nước đi và sử dụng một hàm lượng giá để đánh giá các trạng thái:

* Xây dựng hàm lượng giá:
  + Lưu trữ các thế cờ có khả năng dẫn tới chiến thắng và điểm tương ứng cho mỗi thế cờ đó (0- chưa đánh, 1- người đánh, 2- máy đánh, ‘;’ là rìa bàn cờ)

|  |
| --- |
| String theCoCuaNguoi[] = {"0110", "01010", "01112", "21110", "01110", "011010", "010110", "011001", "100110", "211010", "010112", "201110", "011102", "210110", "011012", "0101010", ";11100", "00111;", ";10110", "01101;", ";01110", "01110;", "011110", "2111100", "0011112", "2011110", "0111102", "0110110", "2110110", "0110112", "2101110", "0111012", "2111010", "0101112","0101110", "0111010", "01101010", "01010110", "2101101", "1011012", ";11110", "01111;", ";11011", "11011;", ";01111", "11110;", ";10111", "11101;", ";11101", "10111;", ";101101", "101101;", ";101011", "110101;", "1111", "111110", "11111", "011111", "111111", ";11111", "11111;"};  String theCoCuaMay[] = {"0220", "02020","02221", "12220", "02220",  "022020", "020220", "022002", "200220", "122020", "020221", "102220",  "022201", "120220", "022021", "0202020", ";22200", "00222;", ";20220",  "02202;", ";02220", "02220;", "022220", "1222200", "0022221", "1022220",  "0222201", "0220220", "1220220", "0220221", "1202220", "0222021",  "1222020", "0202221", "0202220", "0222020", "02202020", "02020220",  "1202202", "2022021", ";22220", "02222;", ";22022", "22022;", ";02222",  "22220;", ";20222", "22202;", ";22202", "20222;", ";202202", "202202;",  ";202022", "220202;", "2222", "222220", "22222", "022222", "222222",  ";22222", "22222;"};  int diemTheCo[] = {6, 6, 4, 4, 12, 12, 12, 8, 8, 4, 4, 7, 7, 7,  7, 9, 4, 4, 7, 7, 7, 7,3000, 800, 800, 800, 800, 900, 800, 800, 800, 800,  800, 800, 900, 900, 800, 800, 800, 800, 800, 800, 800, 800, 800, 800,800,  800, 800, 800, 800, 800, 800, 800, 12, 100000, 3000, 100000, 100000,  100000, 100000}; |

* + Tính điểm của mỗi trạng thái: dùng một xâu s lưu trữ tất cả các hàng, cột, đường chéo (ví dụ một hàng ngang có dạng ‘;010112022011010;’ )
    - Duyệt tất cả các hàng ngang, cộng vào xâu s
    - Duyệt tất cả hàng dọc, cộng vào xâu s
    - Duyệt các đường chéo độ dài từ 5 trở lên, cộng vào xâu s
    - Với mỗi thế cờ của máy, tìm số lần xuất hiện của nó trong xâu s, mỗi lần xuất hiện sẽ tăng lên một số điểm tương ứng
    - Với mỗi thế cờ của người, tìm số lần xuất hiện của nó trong xâu s, mỗi lần xuất hiện sẽ trừ đi một số điểm tương ứng

→ kết quả cuối cùng sẽ là điểm của trạng thái bàn cờ

→ hàm lượng giá này cũng được dùng để đánh giá trạng thái đích

* Chiến lược phân nhánh cho cây tìm kiếm: Với mỗi trạng thái bàn cờ, em sẽ tính điểm cho tất cả các ô trên bàn cờ, chọn ra một số lượng nhất định những ô điểm cao nhất (tức là nếu đánh vào sẽ dẫn đến khả năng thắng cao nhất) để đưa vào cây tìm kiếm. Số lượng ô được chọn sẽ phụ thuộc vào hệ số phân nhánh.

Cách tính điểm cho ô cờ: em đưa vào 2 mảng tính điểm là

|  |
| --- |
| int diemCong[] = {0, 2, 7, 33, 127};  int diemThu[] = {0, 3, 8, 34, 138}; |

Xét 5 ô cờ liên tiếp theo các chiều ngang, dọc, chéo. Giả sử trong 5 ô đó có i (0 ≤ i < 5) quân cờ X (quân cờ của người) và không có quân cờ O (quân của máy) nào, thì có 2 trường hợp:

* Nếu đang là lượt của X: thì 5 ô cờ đó mỗi ô sẽ được công thêm số điểm là diemCong[i]
* Nếu là lượt củaO: mỗi ô trong 5 ô cờ đó sẽ được cộng thêm số điểm bằng diemThu[i]
* Nếu 5 ô đó nằm giữa 2 con O thì mỗi ô trong đó sẽ có điểm bằng 0

Tính tương tự với trường hợp trong 5 ô có i con O (quân của máy) và không có con X nào, thì nếu đang là lượt máy đánh thì mỗi ô sẽ được cộng điểm công tương ứng, … . Ở đây em cho điểm công cao hơn điểm thủ tương ứng, thì điều này sẽ làm cho máy đánh có xu hướng tấn công.

Với mỗi chuỗi 5 ô cờ liên tiếp ta đều tính điểm như thế: giả sử đang tính theo hàng ngang, thì các chuỗi được xét sẽ là (0 → 4), (1 → 5), ... , (10 → 14). Tương tự đối với hàng dọc và đường chéo.

Việc tính điểm ô này sẽ được thực hiện mỗi lần gọi hàm MIN-VALUE hoặc MAX-VALUE. Sau đó, hàm MIN-VALUE (MAX-VALUE) sẽ chọn ra số lượng nhất định b các ô có điểm cao nhất (b chính là độ phân nhánh) để xét. Phần còn lại là thực hiện cắt tỉa alpha - beta.

* Độ sâu: giới hạn độ sâu tìm kiếm, nếu đạt đến ngưỡng độ sâu thì trả về giá trị hàm lượng giá.
* Hàm kiểm tra kết thúc: mỗi lần một ô được đánh, hàm này sẽ kiểm tra xem trong các hàng ngang, dọc, đường chéo đi qua ô này có dãy 5 nước hợp lệ không, nếu có thì trả về true - trò chơi kết thúc.

→ Đánh giá độ phức tạp: với độ sâu là d và độ phân nhánh b thì độ phức tạp về thời gian sẽ là O(bm)

→ Vấn đề đặt ra là chọn b và m cho phù hợp để vừa đảm bảo được về mặt thời gian mà độ thông minh của AI là chấp nhận được.

# Thử nghiệm

|  |  |  |
| --- | --- | --- |
| Độ sâu d | Độ phân nhánh b | Thời gian trung bình cho nước đi đầu tiên (ms) |
| 14 | 4 | 21236 |
| 11 | 4 | 17611 |
| 11 | 5 | 20395 |
| 9 | 6 | 28014 |
| 8 | 7 | 21312 |

# Các đề xuất cải tiến hệ thống

Thuật toán trò chơi còn điểm hạn chế, đó là mỗi lần gọi hàm MIN-VALUE hoặc MAX-VALUE thì lại phải phải tính lại điểm cho tất cả các ô bàn cờ. Việc này dẫn đến phải tính nhiều phép toán dư thừa, do những ô nằm xa so với vùng đang đánh có khả năng đánh vào rất thấp. Để cải tiến thì em sẽ chỉ tính điểm các ô chưa đánh mà nằm trong vùng bán kính của các ô đã đánh. Chúng em sẽ cải tiến theo hướng này trong tương lai.

# Kết luận

Dựa trên kết quả thử nghiệm và nhiều lần đánh thử, nhóm chúng em quyết định chọn độ sâu 7 và độ phân nhánh 8 . Độ thông minh và thời gian chạy của AI là chấp nhận được.

# Tài liệu tham khảo

[1] Slide Trí tuệ nhân tạo - Nguyễn Nhật Quang

[2] Một số lí thuyết tham khảo trên internet