12/29/2014

Lê Khả Sỹ

GV : Huỳnh THỊ THANH THƯƠNG

Báo cáo đồ án môn Nhập môn trí tuệ nhân tạo

Đề tài : Cờ vua

Mục Lục

[CHƯƠNG 1: GIỚI THIỆU BÀI TOÁN 3](#_Toc408313635)

[Nguồn gốc trò chơi 3](#_Toc408313636)

[Mô tả trò chơi 4](#_Toc408313637)

[Ứng dụng của trò chơi 5](#_Toc408313638)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 6](#_Toc408313639)

[ý tưởng của chiến lược tìm kiếm minimax 6](#_Toc408313640)

[Nguyên lí của cắt tỉa alpha-Beta 6](#_Toc408313641)

[CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ 8](#_Toc408313642)

[3.1. Phân tích bài toán 8](#_Toc408313643)

[3.2. Cấu trúc dữ liệu và cách biểu diễn các trạng thái của bài toán 8](#_Toc408313644)

[Lớp AI 8](#_Toc408313645)

[Cấu trúc dữ liệu của Node trên cây minimax 9](#_Toc408313646)

[Cấu trúc dữ liệu của trạng thái 9](#_Toc408313647)

[Cấu trúc dữ liệu của quân cờ 10](#_Toc408313648)

[3.3. Các vấn đề và thuật giải 12](#_Toc408313649)

[quản lí Value của mỗi node 12](#_Toc408313650)

[TÌM NODE CON TỐT NHẤT BẰNG THUẬT GIẢI MINIMAX VỚI ĐỘ SÂU GIỚI HẠN 12](#_Toc408313651)

[Lấy các status con có thể phát sinh 15](#_Toc408313652)

[Mở rộng một node 17](#_Toc408313653)

[Đánh giá một status bằng heuristic 19](#_Toc408313654)

[Kiểm tra cắt tỉa alpha-Beta trên một node 21](#_Toc408313655)

[3.4. Ví dụ minh họa thuật giải minimax, cắt tỉa alphabeta 23](#_Toc408313656)

[CHƯƠNG 4: ỨNG DỤNG 24](#_Toc408313657)

[4.1. Giới thiệu chương trình ứng dụng: 24](#_Toc408313658)

[4.2. Cài đặt 26](#_Toc408313659)

[4.3 Kết quả chạy chương trình. 26](#_Toc408313660)

[Nhận xét giao diện 26](#_Toc408313661)

[Kết quả chạy chương trình 26](#_Toc408313662)

[Hạn chế 26](#_Toc408313663)

[CHƯƠNG 5: KẾT LUẬN 28](#_Toc408313664)

[5.1. Kết quả đạt được 28](#_Toc408313665)

[5.2. Hạn chế 28](#_Toc408313666)

[5.3. Hướng phát triển 28](#_Toc408313667)

[Về Hình thức 28](#_Toc408313668)

[Về xử lí 28](#_Toc408313669)

[TÀI LIỆU THAM KHẢO 29](#_Toc408313670)

# CHƯƠNG 1: GIỚI THIỆU BÀI TOÁN

## Nguồn gốc trò chơi

Tiền thân của cờ vua xuất hiện ở đất nước của nhưng điệu múa bụng nổi tiếng là Ấn Độ, trong thời kỳ của đế chế Gupta, vào khoảng thế kỷ thứ 6 sau công nguyên. Vào thời đó, người ta gọi trò chơi này là chaturanga – trò chơi với nhiều loại binh chủng xuất hiện trên bàn cờ gồm bộ binh, kỵ binh, tượng binh và xa binh tương ứng với các quân cờ hiện đại là chốt, mã, tượng và xe. Ở đất nước Ba Tư khoảng 600 năm sau công nguyên, tên trò chơi được gọi là chatrang và những luật lệ đã bắt đầu phát triển cao hơn, đặc biệt trong số đó người ta gọi “Shāh!” khi tấn công vào Vua đối phương, và “Shāh māt!” khi quân Vua đó bị tấn công và không có cách nào để thoát khỏi. Điểm này tồn tại với cờ vua khi nó được mang đến nhiều vùng đất khác nhau trên thế giới.



Cuộc chinh phục của người Ba Tư đã đưa trò chơi này đến các đất nước Tây Á và tên của trò chơi được sửa lại đôi chút cho hợp với giọng đọc của người dân trong vùng là shatranj. Sau đó, người Maroc ở vùng Bắc Phi đổi tên thành shaterej, đọc theo tiếng Tây Ban Nha có thể là acedrez, axedrez hay ajedrez. Khi cờ vua đến với châu Âu, tùy theo âm điệu giọng nói của từng dân tộc, từng quốc gia mà thành nhiều tên khác nhau như scacchi (Italy), échecs (Pháp), schack (Thụy Điển)…

Cờ vua đã du nhập đến nhiều vùng miền trên thế giới bởi nhiều con đường khác nhau và cũng từ đó phát triển mạnh về số người chơi cờ vua. Vì có nhiều người chơi từ nhiều đất nước dân tộc nên nảy sinh ra nhu cầu phải có luật lệ thống nhất để mọi người có thể giao lưu với nhau dù khác biệt về ngôn ngữ, văn hóa. Các luật lệ được phát triển mạnh mẽ và sớm hình thành hệ thống ở châu Âu vào cuối thế kỷ 15 của thời Phục Hưng.

Chúng ta cùng điểm qua sự phát triển của các quân cờ qua thời gian và khi đến các vùng đất, dân tộc khác nhau.

Ở Ấn Độ ngày trước, quân Hậu chỉ có thể đi chéo 1 ô, quân Tượng thì đi chéo 2 ô không hơn không kém và quân chốt chưa thể đi 2 ô ngay từ đầu. Tuy đã có luật phong cấp nhưng chốt khi ấy cũng chỉ có thể phong thành Hậu, không thể thành quân cờ khác. Riêng Xe, Tượng, Mã thì không có nhiều thay đổi.

Ở châu Âu, người ta thêm 1 luật cho chốt có khả năng nhảy 2 ô ngay từ vạch xuất phát và cũng từ đó hình thành nên luật ăn chốt qua đường. Tiếp theo đó, Vua được cho thêm khả năng nhảy 2 ô 1 lần duy nhất trong ván, đó là tiền đề cho sự hình thành nước nhập thành “nhất cử lưỡng tiện” để Vua được an toàn và Xe ra tấn công ngay. Hậu đã được phép đi ngang dọc chéo trên bàn cờ, tuy cũng mới chỉ được 2 ô. Tuy nhiên, lại không cho một quân đang bị ghim được thực hiện nước chiếu.

Qua nhiều cải cách trên thế giới, cờ vua dần được hoàn thiện và đến với mọi người. Luật cờ vua cũng từ đó mà dần hợp nhất. Thế nhưng, các quân cờ vẫn chưa phát huy hết được sức mạnh của nó, cụ thể là Hậu và Tượng vẫn còn rất yếu. Điều đó cũng do nguyên nhân, khi cờ vua du nhập châu Âu người ta không có Tượng binh và khi đó phụ nữ vẫn chưa được xem trọng. Quân Tượng từ Ấn Độ nay biến thành quân cố vấn (Councilor) và cố vấn cũng chỉ quanh vẩn vua nên sức mạnh quân sự của nó không thể cao như kỵ binh (Mã) hay xa binh (Xe). Nhưng sau đó, người ta dần nhận ra tầm quan trọng của các quý bà cũng như muốn làm cho cờ vua càng trở nên hoàn thiện, Hậu và Tượng đã được nâng cao giá trị qua việc biến chúng trở thành quân tầm xa và đặc biệt là Hậu có thể ngang dọc khắp bàn cờ. Cũng từ đó, việc phong cấp cho chốt càng giá trị hơn và quân chốt yếu ớt ngày này, nay được người ta xem trọng hơn rất nhiều.

Những câu lạc bộ, tổ chức, trung tâm cờ xuất hiện. Đó là nơi nhiều tay cờ nổi tiếng tụ hội để thi đấu học hỏi lẫn nhau hay cũng chính là nơi để những người mới biết chơi rèn luyện khả năng tư duy của mình trên bàn cờ. Nổi tiếng nhất trong số các trung tâm cờ chính là quán cà phê Régence ở Pháp, nơi hai nhà chơi cờ kiệt xuất là Paul Morphy và Andre Philidor đặt chân đến thi đấu.

Các tổ chức cờ hình thành nhằm mục đích xây dựng các hệ thống thi đấu Quốc gia và Quốc tế, thống nhất làng cờ thế giới để tìm ra người giỏi nhất. Và hiện này, chúng ta có tổ chức cờ vua thế giới với tên viết tắt là FIDE.

## Mô tả trò chơi

**Cờ vua**, trước kia còn được gọi là **cờ quốc tế**, là một trò chơi trên bàn và là một môn thể thao trí tuệ cho 2 người chơi. Ngày nay, cờ vua là một trong những trò chơi phổ biến nhất thế giới với hàng triệu người tại nhà riêng, trong câu lạc bộ, trực tuyến, từ xa và trong các giải đấu.

Trò chơi này diễn ra trên một bảng hình vuông, gọi là bàn cờ, gồm 8 hàng (đánh số từ 1 đến 8) và 8 cột (đánh số từ a đến h), tạo ra 64 ô hình vuông với các màu đậm và nhạt xen kẽ nhau, với mỗi người chơi sẽ có ô màu nhạt ở hàng cuối cùng bên tay phải của mình khi ngồi vào bàn chơi cờ. Mỗi người sẽ bắt đầu ván cờ với 16 quân cờ và sẽ lần lượt đi các quân của mình sau khi đối phương đã đi xong một nước (hoàn thành nước đi).

Các quân cờ của mỗi bên bao gồm:  
8 Tốt  [Tốt đen](http://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin:Chess_pdl44.png)[Tốt trắng](http://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin:Chess_pld44.png)  
2 Mã  [Mã đen](http://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin:Chess_ndl44.png)[Mã trắng](http://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin:Chess_nld44.png)  
2 Tượng [Tượng đen](http://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin:Chess_bdl44.png)[Tượng trắng](http://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin:Chess_bld44.png)  
2 Xe  [Xe đen](http://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin:Chess_rdl44.png)[Xe trắng](http://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin:Chess_rld44.png)  
1 Hậu  [Hậu đen](http://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin:Chess_qdl44.png)[Hậu trắng](http://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin:Chess_qld44.png)  
1 Vua  [Vua đen](http://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin:Chess_kdl44.png)[Vua trắng](http://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin:Chess_kld44.png).

Người cầm quân trắng luôn là người đi đầu tiên; người kia cầm quân đen. Các quân Hậu và Xe được gọi là **quân nặng**, còn Tượng và Mã được gọi là **quân nhẹ**.

## Ứng dụng của trò chơi

Bản chất của trò chơi thì thường không có ứng dụng gì nhiều với thực tiễn, nhưng việc làm ra trò chơi đó trên máy tính hôm nay có ích khá nhiều cho kinh nghiệm lập trình và ứng dụng của các kĩ thuật áp dụng trên trò chơi này.

# CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

Để giải quyết bài toán này, nhóm đã sử dụng một thuật giải tìm kiếm có đối thủ khá kinh điểm là Minimax kết hợp với thuật toán cắt tỉa alpha-beta.

## ý tưởng của chiến lược tìm kiếm minimax

Chiến lược Minimax (được thể hiện bằng giải thuật minimax) dựa trên 2 giả thiết sau:   
– Cả 2 đối thủ có cùng kiến thức như nhau về không gian trạng thái của trò chơi – Cả 2 đối thủ có cùng mức cố gắng thắng như nhau

Hai đối thủ trong trò chơi có tên là MAX và MIN   
– Max: biểu diễn cho mục đích của đối thủ này là làm lớn tối đa lợi thế của mình – Min: biểu diễn cho mục đích của đối thủ này là làm nhỏ tối đa lợi thế của đối phương.

Trên cây tìm kiếm sẽ phân lớp thành các lớp Max và Min xen kẽ nhau

Nếu một node thuộc lớp max thì có nghĩa node đó được tạo ra bởi đối thủ max dựa trên một toán tử chuyển trạng thái nào đó từ một node Min là cha của nó. Ngược lại với node Min.

Với một node n bất kỳ:  
– Nếu nó thuộc lớp Max thì gán cho nó giá trị “**nhỏ nhất”** của các node con   
– Nếu nó thuộc lớp Min thì gán cho nó giá trị “**lớn nhất**” của các node con.  
– Nếu không có node con thì : Nếu là Max thì gán giá trị lớn nhất cho node đó và nhỏ nhất nếu đó là node min.  
Cuối cùng, một node có độ sâu = 1 sẽ được lựa chọn làm bước đi tiếp theo.  
nếu lựa chọn một node có giá trị là x, điều đó có nghĩa là nếu đi theo hướng đó thì sau max-1 bước nữa, chúng ta sẽ có được độ tốt tối thiểu là x;

## Nguyên lí của cắt tỉa alpha-Beta

Cắt tỉa Alpha-Beta là một cách để loại bỏ bớt các nhánh thừa trên cây tìm kiếm Minimax dựa trên các tính chất của cây tìm kiếm Minimax.  
Giả sử ta có một nhánh nhỏ của cây tìm kiếm minimax như sau:  
   
  
Trong đó các node B,D đã được định giá trị.  
  
Cắt tỉa Alpha-Beta được lí luận như sau:  
Theo tính chất của cây minimax, ta có: C D  
Nếu D B thì C B.  
Mà A lấy giá trị lớn nhất trong các node con, vì vậy, giá trị của C không còn quan trọng nữa, điều đó có nghĩa là không cần phát triển thêm các node con của C là E nữa.  
Tương tự với trường hợp A là node Max thì chỉ cần điều kiện D B là được.

# CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ

## 3.1. Phân tích bài toán

Nhận thấy đây là một bài toán tìm kiếm có đối thủ, nên ý tưởng đầu tiên là dùng giải thuật Minimax để đưa ra một trạng thái được đánh giá tốt nhất.  
Do số lượng các trạng thái quá lớn nên sẽ chỉ phát triển cây tìm kiếm tới một độ sâu max kết hợp với cắt tỉa alphabeta.  
Ý tưởng chính đi xuyên suốt quá trình phát triển là sẽ có 1 hàm chính, đưa vào một trạng thái, nó sẽ dùng giải thuật minimax với độ sâu max và nó sẽ trả về một trạng thái tối ưu cho độ sâu đó.

## 3.2. Cấu trúc dữ liệu và cách biểu diễn các trạng thái của bài toán

Về cấu trúc dữ liệu của bài toán, phần này chỉ giới thiệu các cấu trúc chính cần lưu ý, các cấu trúc nhỏ khác sẽ chỉ được nói sơ qua hoặc lược bỏ đi.

### Lớp AI

Lớp này đại diện cho trí tuệ của máy tính, với suy nghĩ :   
một chương trình chỉ có 1 trí tuệ, tất cả các phương thức, thuộc tính trong lớp này đềulà static.   
- Inteligent : Độ sâu tối đa của cây minimax  
- rdm : Một đối tượng ngẫu nhiên, giúp chọn ra một kết quả trong số các kết quả giống nhau, giúp ván cờ biến hóa hơn  
- workerThread : danh sách các thread dùng để mở rộng các node con của node gốc

### Cấu trúc dữ liệu của Node trên cây minimax

Để đáp ứng thuật giải minimax, mỗi node cần có  
 \_value : giá trị của node đó  
deep : độ sâu của nó trên cây  
Father : tham chiếu tới cha nó  
status : Status mà nó đại diện cho  
Subs : danh sách các node con của nó  
type : kiểu của node: là một enum gồm có hai kiểu là Min và Max.

### Cấu trúc dữ liệu của trạng thái

Một trạng thái được biểu diễn bằng một ma trận các Quân cờ.  
Mỗi trạng thái có hai giá trị quan trọng khác là   
 - Luot : xác định trạng thái này được tạo ra bởi Man hay Computer  
- Finished : xác định đây có phải là trạng thái kết thúc chưa  
Ngoài ra còn có một số các Static Fields dùng cho UI, các Field này chỉ là phần giao diện, không liên quan tới thuật giải.   
  
Toán tử chuyển trạng thái trên một trạng thái là việc di chuyển một quân cờ từ vị trí này sang vị trí kia.  
Trạng thái đích là trạng thái có Finished = true;

### Cấu trúc dữ liệu của quân cờ

Sau khi có được cái nhìn tổng quát về một Status, ta đi xây dựng một quân cờ tổng quát và từng dẫn xuất của nó.  
Quan cờ được thiết kế như một lớp cơ sở, dùng để đại diện cho một quân cờ bất kì trong một status.



  
  
Trong một Quân cờ có những thuộc tính sau:  
- một đối tượng lưu số dòng và cột của quân cờ này trên bàn cờ (Index)  
- một tham chiếu tới bàn cờ chứa quân cờ này (Owner)  
- một giá trị cho biết quân cờ này thuộc đội nào (Team)   
- để đánh giá độ tốt của một quân cờ, ta không thể gán giá trị trực tiếp cho quân cờ đó được, mà mỗi trường hợp quân cờ đó sẽ có một độ tốt khác nhau. Vì vậy, ta cần một hàm để đánh giá độ tốt của một quân cờ.

Các lớp dẫn xuất của Quanco khá tương tự nhau, đều có thêm một thuộc tính tĩnh là MoveIndex.  
MoveIndex là một mảng tĩnh các số nguyên gồn có 8 ô, được khởi tạo từ khi mới bắt đầu chương trình và không thay đổi cho đến hết chương trình.  
Mảng này lưu số ô tối đa mà một quân cờ có thể di chuyển được theo một hướng nào đó theo thứ tự các hướng như sau:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Trái | Phải | Trên | Dưới | 11h | 5h | 1h | 7h |

Mục đích tạo mảng này là để lấy tất cả các vị trí mà quân cờ đó có thể đi đến được bằng một thuật toán chung cho tất cả các quân cờ.  
Trường hợp quân cờ đó là Ngựa, sẽ có một đoạn code riêng dành cho nó.  
Đây là một ví dụ cho các lớp dẫn xuất từ lớp Quanco:

  
  
  
  
  
  
Nhưng trong số các lớp dẫn xuất này, có một lớp đặc biệt hơn các lớp còn lại, đó là Tốt.  
Bởi vì tốt là con cờ duy nhất trên bàn cờ có quy định chiều di chuyển, nước đi chéo chỉ thực hiện được khi ở đó có quân của đối phương và nó được di chuyển 2 ô ở lần di chuyển đầu tiên.  
  
Vì vậy, nó cần có một số thuộc tính khác nữa:  
 - moved : là true khi con tốt này đã di chuyển rồi  
 - Bốn mảng tĩnh MoveIndex thay vì 1 như các lớp khác.   
Bốn mảng này dùng cho các trường hợp khác nhau:  
MoveIndex : là quân của máy và chưa di chuyển.  
MoveIndex1 : quân của người và chưa di chuyển.  
MoveIndex2 : quân của máy và đã di chuyển.  
MoveIndex3 : quân của người và đã di chuyển.   
  
Nếu dùng các câu lệnh điều kiện so sánh thì hoàn toàn không cần tơi các mảng này, nhưng để tốc độ thực thi nhanh hơn, ta nên dùng những mảng này để giảm bớt các câu lệnh so sánh.

## 3.3. Các vấn đề và thuật giải

Dưới đây sẽ đưa ra các vấn đề để thực hiện thuật giải.  
Việc diễn giải sẽ đi từ cái tổng quát tới cái chi tiết, từ thuật giải ta đang muốn hướng đến cài đặt đầu tiên tới những chi tiết nhỏ hơn phát sinh khi cài đặt.

### quản lí Value của mỗi node

Để code được ngắn gọn hơn, ta quản lí giá trị của thuộc tính value bằng một Properties Value.  
Properties này sẽ chỉ nhận những giá trị phù hợp với loại node thông qua so sánh.  
code cũng khá đơn giản:

#### Code

public int Value

{

get { return \_value; }

set{

if (type == Nodetype.Max)

{// node này lưu giá trị min

if ((\_value > value && \_value!=5000) || \_value == 5000)

{ // giá trị phù hợp và đã được gán ít nhất 1 lần

// hoặc chưa bao h được gán  
 // 5000 là giá trị khởi tạo, bằng 5000 là chưa bị thay đổi

\_value = value;

}

}

else

{

if ((\_value < value && \_value!=5000) || \_value==5000)

{

\_value = value;

}

}

}  
 }

### TÌM NODE CON TỐT NHẤT BẰNG THUẬT GIẢI MINIMAX VỚI ĐỘ SÂU GIỚI HẠN

Như đã trình bày ở chương 2, AI là một lớp có toàn bộ là các thành phần tĩnh. Lớp này là sự mô tả trí tuệ nhân tạo.  
ta đi tìm hiểu thuật giải cho phương thức status Think(status input) của lớp này.

- Có sẵn : độ sâu tối đa (inteligent)  
- Input : một status do người thực hiện nên.  
- Output : một status được lựa chọn qua thuật giải minimax với độ sâu giới hạn.

#### Mã giả

Bước 1: //Khởi tạo

Node root = input; // sao chép input cho node root để xử lí

List<Status> rootsubsstatus = getSubStatus(root.status); // lấy tất cả các status con có thể phát sinh từ input

Bước 2 : // xử lí

Bước 2.1 : với mỗi status con N của input:  
 Bước 2.1.1: tạo một node mới tương ứng với status đó rồi thêm chúng vào tập node con của root (root.Subs)  
 Bước 2.1.2 : tạo một thread mới và chạy code của bước 2.1.3 bằng thread này  
 Bước 2.1.3 : if(!N.Finished)

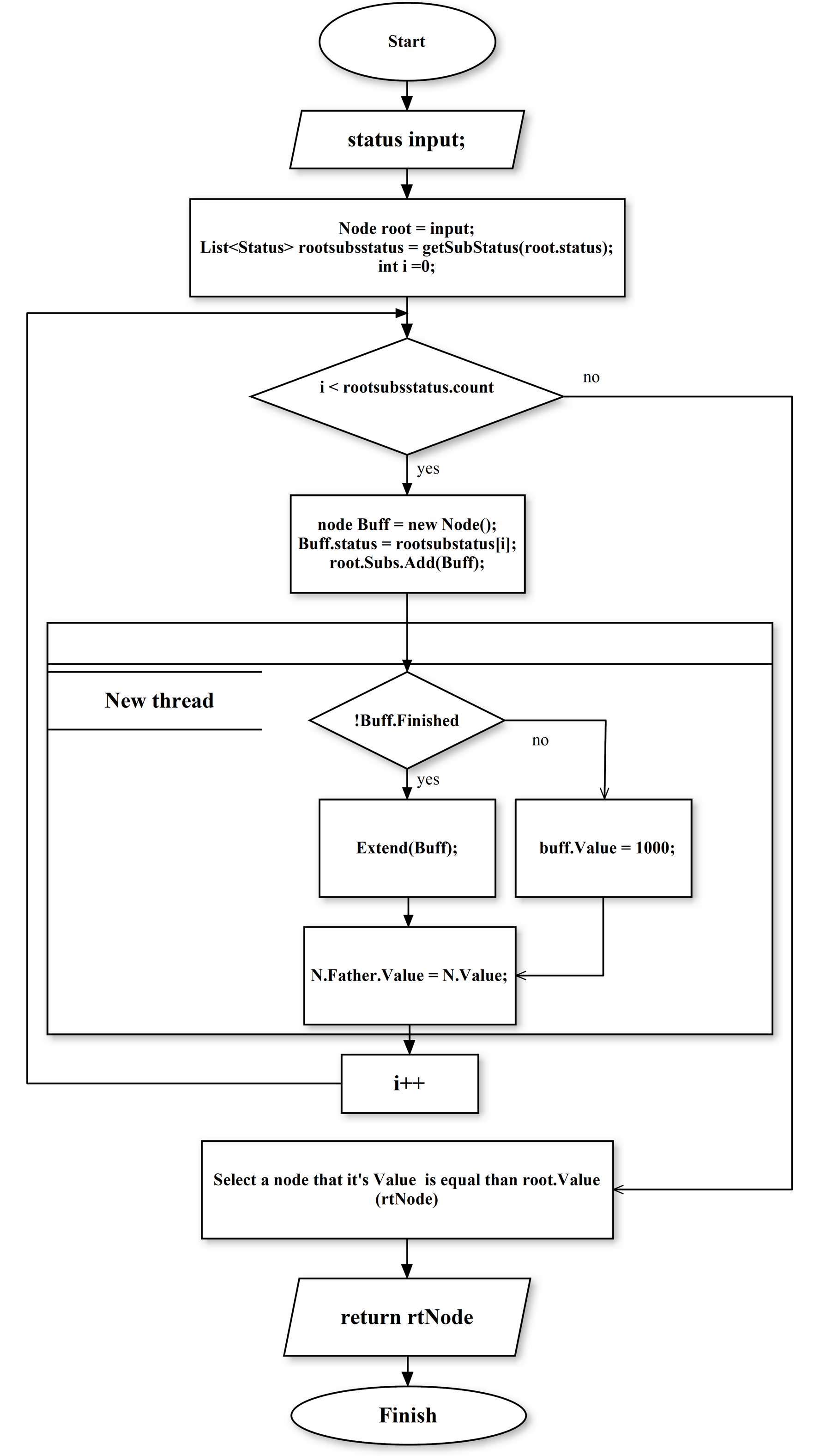
{

mở rộng N ra;  
 gán N.Value = Min(N.Subs);  
 xóa tất cả các con của N để giải phóng bộ nhớ;  
 }  
 else{  
 gán value của node này = 1000;  
 }

gán N.Father.Value = N.Value;

Bước 3 : trả về một trong những status con của root có Value = root.value;

#### Sơ đồ khối



### Lấy các status con có thể phát sinh

- Có sẵn :   
- Input : một status  
- Output : Danh sách các status có thể phát sinh

#### Mã giả

Bước 1 : List<Status> rtlst = new List<Status>();  
 Bước 2: Với mỗi vị trí trên bàn cờ

Bước 2.1 if (vị trí không trống và vị trí khác đội với input.Luot)

{

2.1.1 List<Index> buffarr;

Lấy tất cả vị trí mà quân cờ hiện tại có thể đi tới gán cho buffarr

2.1.2 Với mỗi Item trong buffarr

Status buff;  
 lấy một status từ việc di chuyển quân cờ từ vị trí hiện tại tới vị trí Item rồi gán cho buff

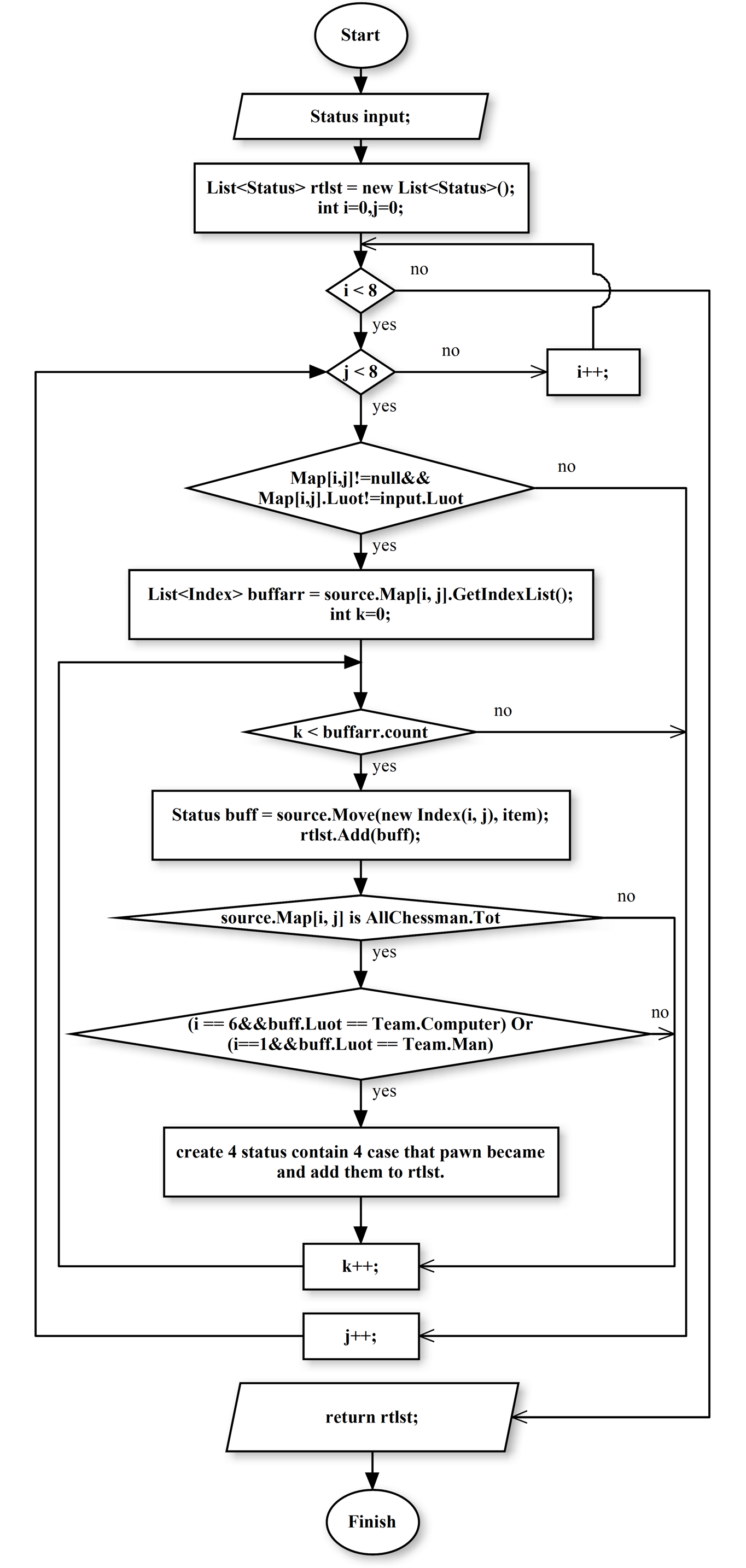
Thêm Buff vào rtlst  
  
 2.1.3 if(quân cờ là một con tốt và đã di chuyển tới vị trí cuối)  
 {  
 tạo nên 4 status mới cho việc phong tốt;  
 thêm 4 status mới đó vào rtlst;

}

}

Bước 3 : return rtlst;

#### Sơ đồ khối



### Mở rộng một node

- Có sẵn : hàm lấy status con có thể phát sinh : list<status> Getsubsstatus(status input);  
 độ sâu giới hạn inteligent;  
- Input : một status cần mở rộng  
- Output : void

#### Mã giả

Bước 1 : List<Status> SubsStatus = AI.getSubStatus(input.status);

Bươc 2 : với mỗi item trong SubsStatus  
 2.1 : Tạo một node buff tương ứng với item là status của nó  
 2.1.2 : if (input là node Max) {  
 gán loại node cho buff là min  
 }  
 else{  
 gán loại node cho buff là Max  
 }  
 2.1.3 : buff.deep = input.deep+1;  
 buff.Father = input;  
 thêm buff vào tập con của input;  
 2.2 : if (chưa tới độ sâu giới hạn)  
 {  
 2.2.1 : if (buff không phải trạng thái kết thúc)  
 {  
 mở rộng buff;  
 if ( buff là node max ) {  
 giá trị của Buff là giá trị nhỏ nhất trong số con của nó.  
 }  
 else {  
 giá trị của Buff là giá trị lớn nhất trong số con của nó.  
 }  
 Xóa tất cả con của Buff đi để tiết kiệm bộ nhớ.  
 }  
 else  
 {  
 if (buff.status.Luot == Team.Computer)

{

buff.Value = 1000;

}

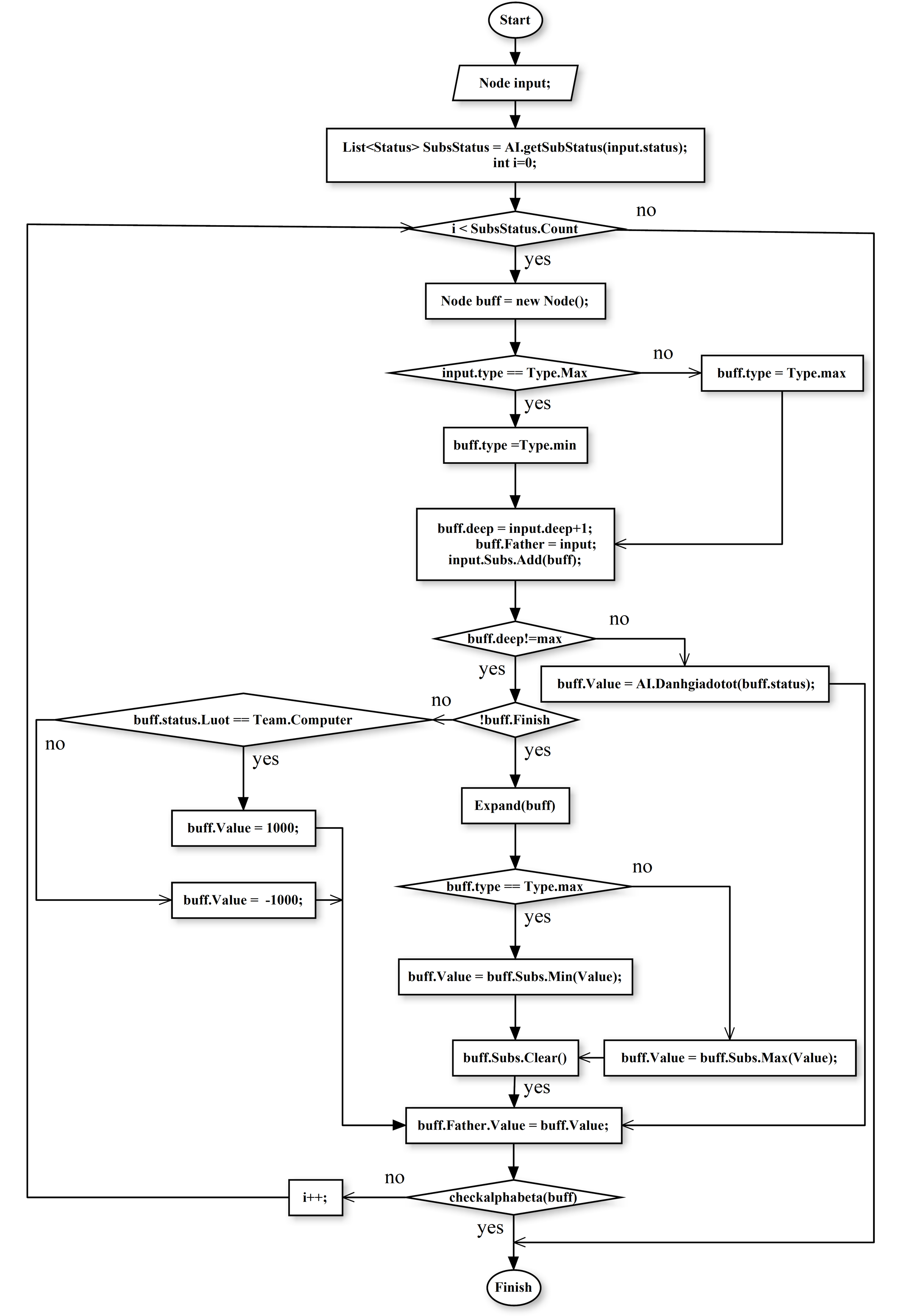
else

{

buff.Value = -1000;  
 }  
 }  
 }  
 else  
 {  
 Dùng hàm heuristic để tìm giá trị cho node này.  
 }

2.3 : gán Properties Value = Buff.Value  
 2.4 : if(Buff thỏa cắt tỉa alphabeta) break;  
Bước 3 : kết thúc

#### Sơ đồ khối



### Đánh giá một status bằng heuristic

Do dùng để phục vụ cho thuật giải minimax với độ sâu giới hạn nên hàm heuristic phải thỏa điều kiện sau:  
trả về giá trị càng lớn thì max càng có lợi, càng nhỏ thì min càng có lợi.  
- Có sẵn : Độ tốt của mỗi quân cờ  
- Input : một status cần đánh giá  
- Output : một giá trị thể hiện sự đánh giá đối với Input

#### Mã giả

Bước 1: int nguoi =0, may =0;  
Bước 2 : với mỗi vị trí index trên bàn cờ  
 2.1 : if(index có quân cờ trên đó)  
 {  
 if(quân cờ đó là của đội máy)   
 cộng độ tốt của quân cờ đó cho máy  
 else cộng độ tốt của quân cờ đó cho người  
 }  
Bước 3 : return may – nguoi;

#### C:\Users\Sy\Desktop\image.PNGSơ đồ khối

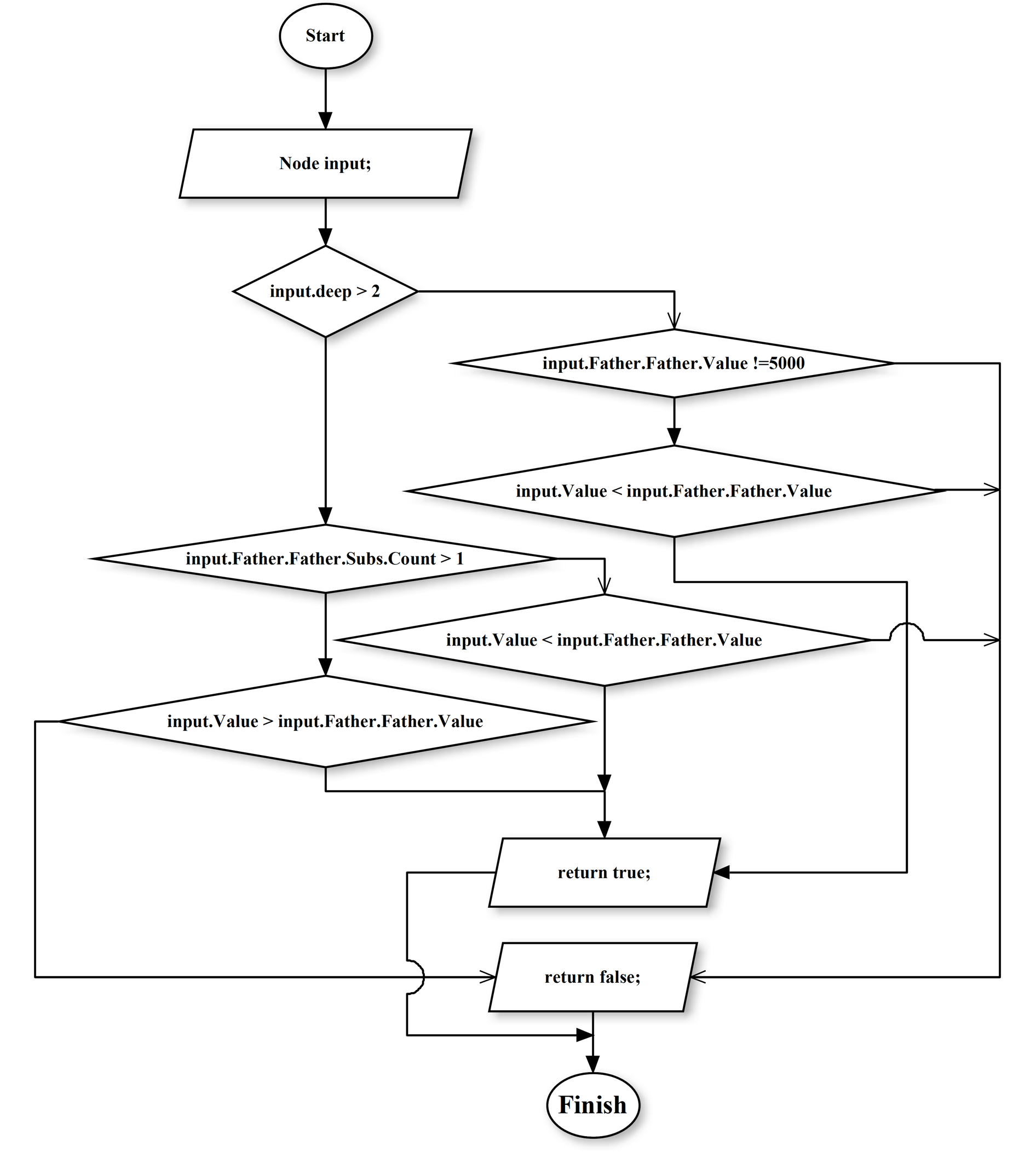
### Kiểm tra cắt tỉa alpha-Beta trên một node

Việc kiểm tra cắt tỉa alphabeta dựa trên lí thuyết về cắt tỉa alphabeta ở chương 2, phần này chỉ trình bày mã giả để cài đặt.  
- Input : một Node cần kiểm tra  
- Output : nếu là một trường hợp cắt tỉa alpha-beta, trả về true, ngược lại là false.

#### Mã giả

Bước 1 : if(độ sâu cuat input > 2 )  
 {  
 1.1 : if(input là node Max)   
 {  
 1.1.1 : if(input.Value > input.Father.Father.Value)  
 {  
 return true;  
 }  
 }  
 else   
 {  
 1.1.2 : if(input.Value < input.Father.Father.Value)  
 {  
 return true;  
 }  
 }  
 }  
 else {  
 1.2 : if(input.Father.Father.Value !=5000)  
 {   
 1.2.1 : if(input.Value < input.Father.Father.Value)  
 {  
 return true;  
 }  
 }  
 }  
Bước 2 : return false;

#### Sơ đồ khối



## 3.4. Ví dụ minh họa thuật giải minimax, cắt tỉa alphabeta

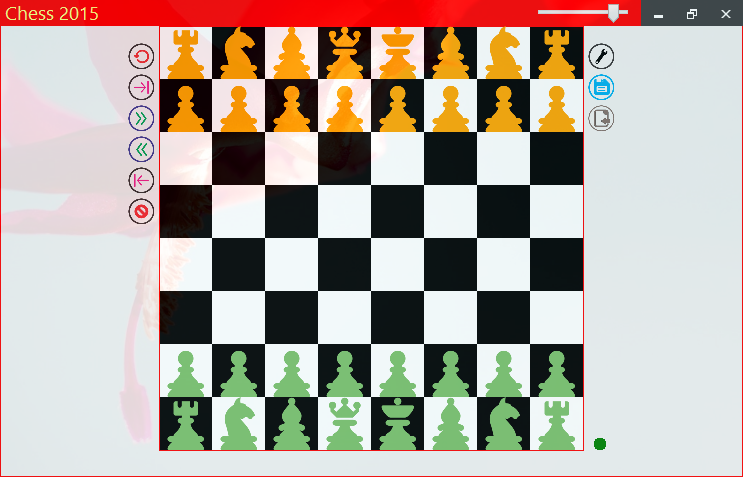
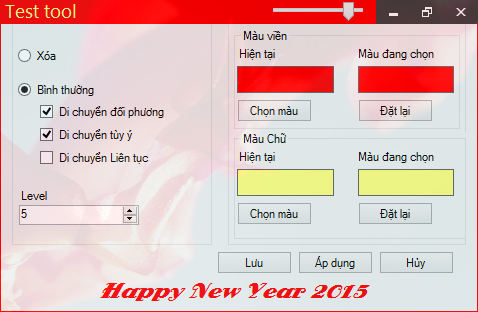
Cách làm việc của thuật giải cũng có phần giống với thuật toán duyệt đồ thị theo chiều sâu.  
khi đạt đến độ sâu max hoặc node đó chứa trạng thái kết thúc thì nó sẽ định giá trị cho node đó và gán giá trị ngược lại cho cha nó.  
các node màu vàng là các node thỏa điều kiện cắt tỉa alphabeta, node màu đỏ là đã đạt trạng thái kết thúc.  
  
VD1 : thuật giải với độ sâu max = 4;

VD2 : thuật giải với độ sâu max = 4; có sử dụng multithread.

Đây là trường hợp có kết hợp multithread, điều này khiến cách cây được mở rộng bị thay đổi, tuy nhiên kết quả vẫn cuối cùng không đổi.  
Lí do xảy ra chuyện này bởi vì mỗi node con của node gốc được mở rộng bằng một thread riêng.  
trong trường hợp này là node 55 mở rộng trước node 52 và nó đã gán giá trị cho node gốc trước node 52, vậy khi node 52 kiểm tra cắt tỉa alphabeta và so sánh giá trị với node gốc thì nó đã thỏa điều kiện.  
Như hình, chắc chắn lần này cây sẽ phải mở rộng ít hơn so với không có multi thread.  
Nhưng thực tế thì chưa chắc, vì đây chỉ là một trường hợp, và không thể biết trước được thread nào sẽ hoàn thành trước thread nào.  
Ngược lại, còn có những trường hợp đáng lẽ sẽ được cắt tỉa, nhưng khi đưa multithread vào thì chúng lại không bị cắt tỉa.  
Tuy nhiên việc không xác định được này chỉ xảy ra ở các node có độ sâu là 2, các node còn lại không bị ảnh hưởng.  
Khi cài đặt đã có một thử nghiệm kiểm tra xem bên nào cắt tỉa tốt hơn, và kết quả là hai bên có mức cắt tỉa trung bình sau 100 lần không lệch nhau nhiều.  
Lí do để chúng ta phải dùng multithread là chúng giúp chương trình chạy nhanh hơn do tận dụng tối đa tài nguyên của CPU.

# CHƯƠNG 4: ỨNG DỤNG

## 4.1. Giới thiệu chương trình ứng dụng:

Ứng dụng gồm có các chức năng :  
ngoài các chức năng cơ bản mà đương nhiên phải có là:  
 - di chuyển các quân cờ  
 - chỉ ra các nước có thể đi của quân cờ đó  
 - kiểm tra tính hợp lệ khi di chuyển  
 - quay lại nước cờ trước đó  
 - bắt đầu với bất kì một thế nào đó trong lịch sử  
 - bắt đầu ván mới với việc reload lại bàn cờ  
ứng dụng có thêm những chức năng sau:  
Về giao diện :   
- thay đổi kích thước tùy ý mà vẫn giữ được tỉ lệ của các đối tượng trên UI.  
- Thay đổi màu sắc của Form lẫn của chữ trên form bất cứ lúc nào.  
Về thao tác :  
- Có một số tool hỗ trợ tester, giúp đặt ra những thế cờ mà tester muốn: Xóa một quân bất kì, di chuyển quân của đối phương, di chuyển liên tục, di chuyển không kiểm tra hợp lệ.  
- Chức năng lưu lại : có thể lưu lại các thiết đặt và ván cờ của người dùng trên ứng dụng để lần sau có thể tiếp tục.  
- Ngắt xử lí : là chức năng được ưng ý nhất của ứng dụng : khi một thao tác phải chờ xử lí quá lâu, thường do người dùng chọn level quá cao, người dùng có thể hủy việc đi vừa rồi để giảm level và tiếp tục chơi.  
Giao diện chính của trò chơi :  
  
  
Giao diện cài đặt :  


## 4.2. Cài đặt

Ứng dụng được viết bằng ngôn ngữ C#.  
Các công cụ:  
 - .Net 4.5  
 - Visual studio 2013  
Nhìn tổng thể ứng dụng thì có một số hàm chính cần chú ý sau:  
 - Status.Getcopy() : lấy một bản sao của status gọi hàm này.  
 - Status.Move(Index From, Index To) : lấy một Status được tạo ra sau khi di chuyển một quân cờ từ vị trí From tới vị trí To.  
 - AI.Think(Status input) : Thực hiện giải thuật minimax với độ sâu max trên input và trả về một trong số các Status có thể được phát sinh từ input có lợi nhất cho máy.  
 - AI.Expand( Node input) : mở rộng một node theo giải thuật minimax.  
 - AI.checkalphabeta(Node input) : Kiểm tra input có phải là một trường hợp cắt tỉa alphabeta hay không.  
 - AI.getSubsStatus(Status input) : Trả về tất cả các trạng thái có thể phát sinh từ input.  
 - AI.danhgiadotot(Status input) : Trả về giá trị của input được đánh giá bằng Heuristic.

## 4.3 Kết quả chạy chương trình.

### Nhận xét giao diện

Giao diện khá đẹp và dễ dùng, có thể điều chỉnh độ trong suốt, các button có hình ảnh kèm theo và tooltip hướng dẫn sử dụng khi chỉ chuột vào.  
Với những người mới làm quen với cờ vua, họ sẽ dễ dàng nắm bắt được luật di chuyển các quân cờ nhờ vào các chấm đỏ khi người chơi ấn vào 1 quân cờ.  
Việc kiểm soát thời điểm được ấn các nút được thực hiện khá tốt.  
Tester được hỗ trợ khá nhiều bởi các chức năng giúp tạo ra các thế cờ mong muốn để xem ứng xử của máy.

### Kết quả chạy chương trình

Chương trình cho ta một kết quả đúng như thuật toán đề ra.  
Tốc độ thực thi được cải thiện khá nhiều nhờ việc sử dụng multithread.  
Video kết quả :  
<http://youtu.be/RAsGE01EoPk>

### Hạn chế

**Về giao diện** : có một bug với các button khi showdialog(). Bug này chưa xử lí được do trình độ chưa cao, chưa nắm được kĩ thuật HOOK. Bug sẽ được khắc phục sớm.  
Việc resize form chưa đạt yêu cầu, sẽ chỉnh lại sớm.  
Các button chỉ đẹp nhất khi ở chế độ toàn màn hình, các chế độ khác, hình ảnh bị thu nhỏ đi và không còn đẹp nữa.  
Đáng lẽ khi máy di chuyển, nên cho người ta thấy máy đã di chuyển từ đâu tới đâu, nhưng để làm được việc đó, ta cần thêm 1 thuộc tính vào Status, điều đó sẽ làm thuật toán chậm đi đáng kể, vì vậy, em đã quyết định lược bỏ đi thuộc tính đó. Thay vào đó, sẽ sử dụng một danh sách khác ở ngoài để lưu giữ, việc này cần thêm thời gian. Hiện tại, nếu người chơi bỏ lỡ, không biết máy đã đi nước nào thì có thể ấn nút quay lại để xem.  
Hình ô vuông màu đỏ trông khá thô kệch, nếu có thời gian em sẽ vẽ một hình ảnh khác mềm hơn.  
  
**Về xử lí** : máy còn rất lúng túng khi đến những nước cuối ván, khi mà hai bên còn lại ít quân, máy thường rất khó khăn trong việc dồn ép vua đối phương vào góc, và nhiều trường hợp đi lung tung vì trong vòng 4 - 5 nước đi, quân máy không có cách nào làm thay đổi chỉ số cả. Việc này em vẫn chưa tìm được cách xử lí thích hợp, có lẽ phải thêm vào đó một vài heuristic khác, nhưng để xử lí nó thì rất phức tạp.

# CHƯƠNG 5: KẾT LUẬN

## 5.1. Kết quả đạt được

Đã thiết lập được nền tảng của trò chơi cờ vua, giao diện lẫn xử lí.

## 5.2. Hạn chế

Có thể nói kết quả chưa có gì đáng chú ý bởi hiệu suất thực thi còn quá thấp, thời gian chờ xử lí quá lâu, heuristic nghèo nàn.  
Một số vấn đề về giao diện chưa ổn định.  
  
Khi hoàn thành sản phẩm, em có cho chơi thử với một ứng dụng khác được tải về, kết quả là máy của em thua thảm hại cả về kết quả lẫn hiệu suất.  
Khi bật taskmanager lên để theo dõi, ứng dụng của em mất hơn 2 giây, trong khi ứng dụng của họ chỉ mất 0.3 giây.  
Mức dùng CPU của họ chỉ là 5%, trong khi ứng dụng của em là 100%.  
Cứ như là ứng dụng của họ chỉ phải duyệt qua 1 list và chọn 1 từ list đó vậy.  
Hiệu suất đó gần như là không thể với cây minimax có độ sâu là 4 và đó mới chỉ là mức nornmal của họ.  
Đến giờ em vẫn ko hiểu họ làm cách nào.

## 5.3. Hướng phát triển

### Về Hình thức

Sau môn lập trình mạng và lập trình web, em sẽ thử đưa ứng dụng này lên web và cho chế độ multiplayer.  
khắc phục một số Bug giao diện.  
Thêm chức năng gợi ý.

### Về xử lí

Tiếp tục tìm hiểu các giải thuật khác để áp dụng.

# TÀI LIỆU THAM KHẢO

[Artificial Intelligence - A Modern Approach - 3rd Edition](http://51lica.com/wp-content/uploads/2012/05/Artificial-Intelligence-A-Modern-Approach-3rd-Edition.pdf)  
[THE QUEST FOR ARTIFICIAL INTELLIGENCE](http://ai.stanford.edu/~nilsson/QAI/qai.pdf)