VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

———————— * ————————

BÁO CÁO MÔN

TRÍ TUỆ NHÂN TẠO

ĐỀ TÀI

XÂY DỰNG GAME CỜ CARO

Sinh viên thực hiện:

Nguyễn Thị Diệu Mơ	**20162763**
Công Việt Hùng	**20161922**
Thoeun Rathana	**20167988**

Lớp: **111572**

Giảng viên hướng dẫn: **Nguyễn Nhật Quang**

Hà Nội, tháng 12 năm 2019

MỤC LỤC

Contents

LỜI NÓI ĐẦU

Game cờ caro là một trong những trò chơi yêu thích và phổ biến thời học sinh, sinh viên. Luật chơi của cờ caro khá đơn giản và cách chơi cũng vậy, chỉ cần có tờ giấy kẻ các ô vuông và vài chiếc bút thì 2 người có thể chơi cờ với nhau. Tuy nhiên, việc chơi cờ caro giữa 2 người với nhau thì hoàn toàn đơn giản, ở cấp độ khó hơn đó là phải xây dựng một trò chơi có khả năng chơi với con người ở các mức độ khác nhau. Để làm được điều đó đòi hỏi phải xây dựng một thuật toán khá tốt và xây dựng một giao diện có thể đáp ứng được. Đề tài này khá thú vị và cũng là sở thích của chúng em, vì vậy chúng em đã chọn làm game cờ caro cho bài tập lớn môn Trí tuệ nhân tạo.

Để hoàn thành được bài tập lớn này, nhóm chúng em xin được gửi lời cảm ơn chân thành đến thầy giáo hướng dẫn đề tài Thầy Nguyễn Nhật Quang, Giảng viên Bộ môn Hệ thống thông tin, Viện Công nghệ Thông tin và Truyền thông, Trường Đại học Bách Khoa Hà Nội, đã hết lòng giúp đỡ, hướng dẫn, chỉ dạy tận tình để nhóm em hoàn thành được đề tài này.

Hà Nội, tháng 12 năm 2019

PHÂN CÔNG THÀNH VIÊN TRONG NHÓM

STT	Họ tên	MSSV	Công việc đóng góp
1	Nguyễn Thị Diệu Mơ	20162763	Xây dựng thuật toán cắt tỉa Alpha – Beta, xây dựng các lớp, thiết kế báo cáo
2	Công Việt Hùng	20161922	Xây dựng thuật toán MINIMAX, xây dựng giao diện chương trình, thiết kế slide.
3	Thoeun Rathana	20167988	Làm báo cáo, slide

1. Khái quát trò chơi cờ Caro

Cờ Caro là một trò chơi phổ biến và khá ưa thích. Cờ caro không chỉ được chơi ở Việt Nam mà còn ở rất nhiều nước trên thế giới. Ở mỗi nước nó lại có cái tên khác nhau. Ở Nhật Bản, người ta gọi cờ caro là **Gomoku**. Ở Anh, Cờ Caro được gọi là **Connect5**. Người Trung Quốc gọi cờ caro là **Guziqi**. Người vùng Đông Âu có cách gọi cờ caro bằng **Five in a row** nghĩa là Một Hàng Năm. Người Hàn Quốc gọi cờ Caro là **Omok**.

Luật chơi: Hai bên lần lượt đánh các quân cờ vào ô vuông, một bên đánh quân X, một bên đánh quân O. Người thắng là người đánh được 5 quân cờ cùng kiểu trên hàng dọc, hàng ngang hoặc đường chéo mà không bị chặn 2 đầu. Hai bên hòa nhau khi đánh hết ô vuông mà chưa đánh được 5 quân cùng kiểu như trên hoặc tiếp tục đánh phân thắng bại trên bàn cờ mở rộng.

2. Phương pháp giải quyết

2.1. Xây dựng các hàm lượng giá dùng để tính điểm cho các nước đi.

Mỗi nước đi sẽ có 2 loại điểm: điểm tấn công và điểm phòng ngự

Điểm tấn công đại diện cho khả năng tạo các quân cờ liền nhau theo hàng ngang/dọc/chéo. Số quân cờ liền nhau tạo được càng nhiều thì điểm sẽ càng cao.

Điểm phòng ngự đại diện cho khả năng ngăn chặn đối phương tạo các quân cờ liền nhau. Số quân cờ của đối phương bị chặn càng nhiều thì điểm sẽ càng cao.

Mỗi loại điểm (tấn công/phòng ngự) sẽ bằng tổng điểm của 4 hàm lượng giá tương ứng với 4 hướng đi (ngang/dọc/chéo xuôi (hướng Tây Bắc – Đông Nam)/chéo ngược (hướng Đông Bắc – Tây Nam) của loại điểm đó.

Điểm số sau cùng đánh giá nước đi đó được tính bằng điểm lớn hơn trong 2 loại điểm là tấn công và phòng ngự.

2.2. Ý tưởng xây dựng hàm lượng giá

Đối với mảng điểm tấn công, xét theo 4 hướng là ngang, dọc, chéo xuôi, chéo ngược, mỗi hướng xét theo 2 phía.

Với mỗi hướng, duyệt lần lượt từng ô xuất phát từ ô đang xét theo 4 hướng trên.

Mỗi lần gặp 1 quân ta thì tăng chỉ số tấn công thêm 1 đơn vị (lúc đầu là 0)

Nếu gặp quân địch hoặc nước chưa có ai đánh thì dừng, tăng chỉ số phòng ngự thêm 1 đơn vị và chuyển sang xét phía còn lại.

Nếu gặp quân ta thì tiếp tục tăng chỉ số tấn công.

Nếu gặp quân địch thêm 1 lần nữa (chặn 2 đầu) thì chỉ số tấn công quay về 0.

Nếu gặp ô trống thì thoát vòng lặp.

2.3. Giải thuật Minimax

Minimax là giải thuật là một thuật toán đệ quy lựa chọn bước đi kế tiếp trong một trò chơi có hai người bằng cách định giá trị cho các Node trên cây trò chơi sau đó tìm Node có giá trị phù hợp để đi bước tiếp theo.

Như chúng ta đã biết thì có rất nhiều thuật toán tìm kiếm để làm AI trong game như A*, Heuristic... Mỗi thuật toán thì sẽ phù hợp với từng loại game cho nó. Những game đối kháng, người chơi luân phiên đánh như cờ vua, cờ tường, caro... Khi chơi ta có thể khai triển hết không gian trạng thái nhưng khó khăn chủ yếu là chúng ta phải tính toán được phản ứng và nước đi của đối thủ mình như thế nào? Cách xử lý đơn giản: giả sử đối thủ cũng sử dụng kiến thức về không gian trạng thái giống ta. Giải thuật Minimax áp dụng giả thuyết này để

tìm kiếm không gian trạng thái của trò chơi. Trường hợp này thuật toán minimax sẽ đáp ứng những gì mình cần.

* Các khái niệm:

- Cây trò chơi (Game tree) - Một sơ đồ hình cây thể hiện từng trạng thái, từng trường hợp của trò chơi theo từng nước đi.
- Mỗi node biểu diễn 1 trạng thái của trò chơi hiện tại trên cây trò chơi.
- Node được gọi nút lá là tại đó trò chơi kết thúc (trạng thái trò chơi lúc đó có thể thắng, thua hoặc hòa).

Giải thuật Minimax: Hai người chơi trong game được đại diện là MAX và MIN. MAX đại diện cho người chơi luôn muốn chiến thắng và cố gắng tối ưu hóa ưu thế của mình còn MIN đại diện cho người chơi cố gắng cho người MAX giành số điểm càng thấp càng tốt. Giải thuật Minimax thể hiện bằng cách định trị 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. Từ các giá trị này người chơi sẽ lựa chọn cho mình nước đi tiếp theo hợp lý nhất.

* Giải thuật MINIMAX

```
function MINIMAX-DECISION(state) returns an action
    v ← MAX-VALUE(state)
    return the action in SUCCESSORS(state) with value v

function MAX-VALUE(state) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← −∞
    for a, s in SUCCESSORS(state) do
        v ← MAX(v, MIN-VALUE(s))
    return v

function MIN-VALUE(state) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← ∞
    for a, s in SUCCESSORS(state) do
        v ← MIN(v, MAX-VALUE(s))
    return v
```

2.4. Sử dụng giải thuật cắt tỉa alpha-beta để tìm nước đi tối ưu cho máy tính

Thuật toán Alpha-beta là một cải tiến của thuật toán Minimax nhằm tỉa bớt nhánh của cây trò chơi, làm giảm số lượng nút phải sinh và lượng giá do đó có thể tăng độ sâu của nút.

Thuật toán cắt tỉa alpha – beta

---

function ALPHA-BETA-SEARCH(*state*) **returns** *an action*
  inputs: *state*, current state in game
  $v \leftarrow$ MAX-VALUE($state, -\infty, +\infty$)
  return the *action* in SUCCESSORS(*state*) with value v

---

function MAX-VALUE($state, \alpha, \beta$) **returns** *a utility value*
  inputs: *state*, current state in game
    α, the value of the best alternative for MAX along the path to *state*
    β, the value of the best alternative for MIN along the path to *state*
  if TERMINAL-TEST(*state*) **then return** UTILITY(*state*)
  $v \leftarrow -\infty$
  for a, s in SUCCESSORS(*state*) **do**
    $v \leftarrow$ MAX(v, MIN-VALUE(s, α, β))
    if $v \geq \beta$ **then return** v
    $\alpha \leftarrow$ MAX(α, v)
  return v

function MIN-VALUE($state, \alpha, \beta$) **returns** *a utility value*
  inputs: *state*, current state in game
    α, the value of the best alternative for MAX along the path to *state*
    β, the value of the best alternative for MIN along the path to *state*
  if TERMINAL-TEST(*state*) **then return** UTILITY(*state*)
  $v \leftarrow +\infty$
  for a, s in SUCCESSORS(*state*) **do**
    $v \leftarrow$ MIN(v, MAX-VALUE(s, α, β))
    if $v \leq \alpha$ **then return** v
    $\beta \leftarrow$ MIN(β, v)
  return v

Áp dụng vào bài toán tìm kiếm nước đi cho máy tính:

- State: gồm một mảng 2 chiều đại diện cho bàn cờ, phần tử có giá trị bằng 0 khi ô trống, bằng 1 nếu đó là nước đi của máy tính, bằng 2 nếu đó là nước đi của người chơi, và một biến depth đại diện cho độ sâu muốn xét.

```
1 reference
private OCo maxValue(OCo[,] _MangOCo, long depth, long alpha, long beta)...
2 references
private OCo minValue(OCo[,] _MangOCo, long depth, long alpha, long beta)...
```

- Terminal-Test: hàm Max-Value và Min-Value sẽ trả về giá trị luôn nếu depth = 0 nghĩa là đã xét tới độ sâu mong muốn. Giá trị trả về là một đối tượng ô cờ, cụ thể là ô cờ có điểm cao nhất với state nhập vào.

```
1 reference
private OCo maxValue(OCo[,] _MangOCo, long depth, long alpha, long beta)
{
    if (depth == 0)
    {
        OCo oCoMax = new OCo();
        oCoMax = TimKiemNuocDi();
        return oCoMax;
    }
```

- Successors(state): với state nhập vào, successors(state) là mảng 2 chiều đại diện cho bàn cờ và CÓ THÊM 1 nước đi đang xét ngoài những nước đã có trong state nhập vào.

```
for (int i = 0; i < _BanCo.SoDong; i++)
{
    for (int j = 0; j < _BanCo.SoCot; j++)
    {
        if (_MangOCo[i, j].SoHuu == 0)
        {
            long currScore = TinhDiem(i, j);
            _MangOCo[i, j].SoHuu = 1;
```

- Điểm của mỗi state được định nghĩa như sau: đối với MAX, điểm của state đang xét bằng điểm của nước đi đang xét trừ đi điểm của nước đi tốt nhất mà MIN sẽ đi nếu MAX đi nước đi đó. Đối với MIN, điểm của state sẽ bằng số đối của điểm của nước đi đang xét cộng với điểm của nước đi tốt nhất mà MAX sẽ đi nếu MIN đi nước đó. Nói cách khác, điểm của state được đánh giá bằng hiệu nước đi của MAX với nước đi của MIN. Điểm số này càng cao, nghĩa là nước đi của MAX có điểm cao hơn MIN, thì càng có lợi cho MAX và ngược lại.

```
long currScore = TinhDiem(i, j);
_MangOCo[i, j].SoHuu = 1;
OCo oCoMin = new OCo();
oCoMin = minValue(_MangOCo, depth - 1, alpha, beta);
if(currScore - TinhDiemPlayer(oCoMin.Dong, oCoMin.Cot) > v)
{
    oCoResult = new OCo(_MangOCo[i, j].Dong, _MangOCo[i, j].Cot, _MangOCo[i, j].ViTri, _MangOCo[i, j].SoHuu);
    v = currScore - TinhDiemPlayer(oCoMin.Dong, oCoMin.Cot);
}
```

- Sau khi tìm được state tối ưu, hàm Max-Value và Min-Value sẽ trả về một đối tượng ô cờ, tức nước đi, mà sinh ra state tối ưu đó.

2.5. Bảng so sánh số node cần duyệt của 2 thuật toán Alpha – Beta Pruning và Minimax

Độ sâu	Alpha – Beta Pruning	Minimax
2	795	157608
3	15761	62097552
4	312041	Vô cùng lớn

3. Cài đặt

Chương trình là một Windows Form App được viết bằng C#, muốn chạy chương trình chỉ cần chạy file .exe.

4. Các khó khăn gặp phải

- Trong quá trình thực hiện bài tập lớn, nhóm em khó khăn trong việc sắp xếp thời gian họp nhóm vì các thành viên trong nhóm có lịch học không giống nhau.
- Khi xây dựng hàm lượng giá nhóm chưa thể tối ưu được thuật toán đưa ra dẫn đến chương trình chạy chưa thực sự tốt như thuật toán không thể duyệt được hết các nút.
- Xảy ra mâu thuẫn trong quá trình đưa ra thuật toán giữa các thành viên trong nhóm.
- Nhóm không có nhiều thời gian tìm hiểu về thiết kế giao diện nên giao diện chưa thật sự thân thiện.

5. Đề xuất phát triển

- Thiết kế giao diện chuyên nghiệp hơn
- Xây dựng thuật toán tìm kiếm nước đi tối ưu nhất, có thể duyệt được độ sâu lớn nhất.

6. Tài liệu tham khảo

- Slide bài giảng Trí tuệ nhân tạo, TS. Nguyễn Nhật Quang
- Source code tham khảo: https://sharecode.vn/source-code/source-code-game-caro-viet-bang-c-1670.htm
- Thuật toán minimax và cắt tỉa alpha- beta: http://doc.edu.vn/tai-lieu/luan-van-giai-thuat-tim-kiem-minimax-va-ung-dung-trong-cac-tro-choi-co-tong-bang-khong-22012/, https://viblo.asia/p/thuat-toan-minimax-ai-trong-game-APqzeaVVzVe