THE UNIVERSITY OF HONG KONG

Department of Computer Science

CSIS0270/COMP3270 Artificial Intelligence

Chinese Chess Project, by Haibin Lin

1. Project Description

Write a program for playing either chess or Chinese chess. The game will be between a human and the computer. Graphical User Interface is not required. You can accept text input for moves by human. After every move, the board configuration will be printed out using text graphics, e.g. you can use C for Castle, B for Bishops etc.

Write a report describing in detail your methodology, such as the search engine, whether α-β pruning is used, heuristics applied, and the evaluation function you use. Also state the number of look ahead steps that your program uses, and whether you use any database for beginning of games and end games.

You can use C++ or Java for implementation. If you want to use other programming languages, contact the tutor first.

2. Getting started

To run the game, type the following command in terminal:

```
$    cd bin
$    java com.linhaibin.chess.GameLauncher
```

Then you'll see a welcome screen:

```
Welcome to Eric Haibin Lin's Chinese Chess.
K => King,       A => Advisor,   B => Bishop,

y\x   0  1  2  3  4  5  6  7  8    x/y
    ==============================
0 ||  r  n  b  a  k  a  b  n  r   || 0
1 ||  +  +  +  +  +  +  +  +  +   || 1
2 ||  +  c  +  +  +  +  +  c  +   || 2
3 ||  p  +  p  +  p  +  p  +  p   || 3
4 ||  +  +  +  +  +  +  +  +  +   || 4
    ==============================
5 ||  +  +  +  +  +  +  +  +  +   || 5
6 ||  P  +  P  +  P  +  P  +  P   || 6
7 ||  +  C  +  +  +  +  +  C  +   || 7
8 ||  +  +  +  +  +  +  +  +  +   || 8
9 ||  R  N  B  A  K  A  B  N  R   || 9
    ==============================
y\x   0  1  2  3  4  5  6  7  8    x/y
```

<= Computer's side

<=   Your side

K => King(将/帅),	A => Advisor(士),	B => Bishop(相),
N => Knight(马),	R => Rook(车),	P => Pawn(兵)

To move your piece, type the x and y position of your piece:

```
Please select a piece to move...
x(0~8), y(0~9): 1 7
Please specify the target position...
x(0~8), y(0~9): 4 7
```

Then Computer will move accordingly.

3. Move generation

In the programme, all kinds of pieces(*KingPiece, KnightPiece, PondPiece*, etc) implement the *Piece* interface.

In interface *Piece*, the method *generateAllMove* generates a list of possible moves. Moves are generated according to the rule of Chinese Chess.

4. Evaluation function

To evaluate the state, three aspects are examined:

a) The existence of pieces on board
Assign values if the piece still exists on board.
K => 10000, A => 40, B => 40, N => 80, R => 180, P => 30

b) The position of pieces
Assign values according to the position of the piece on board. Details are attached in Appendix.

c) The mobility of pieces
Assign values according to the mobility of the piece on board, proportional to the number of moves that piece is able to move.

5. Search Algorithms

The programme uses MinMax Search algorithm to explore the state space. Alpha-beta pruning is applied to make the search more efficient. In the game, 4 levels of state space is searched ahead. No begin/end game database is used.

6. Appendix

The position value for evaluation function is shown as follows:

```
redKingPositionValue = Arrays.asList(
0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,
0,0,0,1,1,1,0,0,0,
0,0,0,10,10,10,0,0,0,
0,0,0,15,20,15,0,0,0);

blackKingPositionValue = Arrays.asList(
0,0,0,15,20,15,0,0,0,
0,0,0,10,10,10,0,0,0,
0,0,0,1,1,1,0,0,0,
0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0);

redRookPositionValue = Arrays.asList(

160,170,160,150,150,150,160,170,160,
170,180,170,190,250,190,170,180,170,
170,190,200,220,240,220,200,190,170,
180,220,210,240,250,240,210,220,180,
180,220,210,240,250,240,210,220,180,

180,220,210,240,250,240,210,220,180,
170,190,200,220,240,220,200,190,170,
170,180,170,170,160,170,170,180,170,
160,170,160,160,150,160,160,170,160,
150,160,150,160,150,160,150,160,150);

blackRookPositionValue = Arrays.asList(
150,160,150,160,150,160,150,160,150,
160,170,160,160,150,160,160,170,160,
170,180,170,170,160,170,170,180,170,
170,190,200,220,240,220,200,190,170,
180,220,210,240,250,240,210,220,180,

180,220,210,240,250,240,210,220,180,
180,220,210,240,250,240,210,220,180,
170,190,200,220,240,220,200,190,170,
170,180,170,190,250,190,170,180,170,
160,170,160,150,150,150,160,170,160);

redAdvisorPositionValue = Arrays.asList(
 0,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0,0,0,

 0,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0,0,0,
 0,0,0,30,0,30,0,0,0,
 0,0,0,0,22,0,0,0,0,
 0,0,0,30,0,30,0,0,0);

blackAdvisorPositionValue = Arrays.asList(
 0,0,0,30,0,30,0,0,0,
 0,0,0,0,22,0,0,0,0,
 0,0,0,30,0,30,0,0,0,
 0,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0,0,0,

 0,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0,0,0);

redBishopPositionValue = Arrays.asList(
0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,

0,0,25,0,0,0,25,0,0,
0,0,0,0,0,0,0,0,0,
20,0,0,0,35,0,0,0,20,
0,0,0,0,0,0,0,0,0,
0,0,30,0,0,0,30,0,0);

blackBishopPositionValue = Arrays.asList(
0,0,30,0,0,0,30,0,0,
0,0,0,0,0,0,0,0,0,
20,0,0,0,35,0,0,0,20,
0,0,0,0,0,0,0,0,0,
0,0,25,0,0,0,25,0,0,

0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0);

redKnightPositionValue = Arrays.asList(
70,80,90,80,70,80,90,80,70,
80,110,125,90,70,90,125,110,80,
90,100,120,125,120,125,120,100,90,
90,100,120,130,110,130,120,100,90,
90,110,110,120,100,120,110,110,90,

90,100,100,110,100,110,100,100,90,
80,90,100,100,90,100,100,90,80,
80,80,90,90,80,90,90,80,80,
70,75,75,70,50,70,75,75,70,
60,70,75,70,60,70,75,70,60);

blackKnightPositionValue = Arrays.asList(
60,70,75,70,60,70,75,70,60,
70,75,75,70,50,70,75,75,70,
80,80,90,90,80,90,90,80,80,
80,90,100,100,90,100,100,90,80,
90,100,100,110,100,110,100,100,90,

90,110,110,120,100,120,110,110,90,
90,100,120,130,110,130,120,100,90,
90,100,120,125,120,125,120,100,90,
80,110,125,90,70,90,125,110,80,
70,80,90,80,70,80,90,80,70);

redPondPositionValue = Arrays.asList(
10,10,10,20,25,20,10,10,10,
25,30,40,50,60,50,40,30,25,
25,30,30,40,40,40,30,30,25,
20,25,25,30,30,30,25,25,20,
15,20,20,20,20,20,20,20,15,

10,0,15,0,15,0,15,0,10,
10,0,10,0,15,0,10,0,10,
0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0);
```

```
blackPondPositionValue = Arrays.asList(
0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,
10,0,10,0,15,0,10,0,10,
10,0,15,0,15,0,15,0,10,

15,20,20,20,20,20,20,20,15,
20,25,25,30,30,30,25,25,20,
25,30,30,40,40,40,30,30,25,
25,30,40,50,60,50,40,30,25,
10,10,10,20,25,20,10,10,10);

redCannonPositionValue = Arrays.asList(
125,130,100,70,60,70,100,130,125,
110,125,100,70,60,70,100,125,110,
100,120,90,80,80,80,90,120,100,
90,110,90,110,130,110,90,110,90,
90,110,90,110,130,110,90,110,90,

90,100,90,110,130,110,90,100,90,
90,100,90,90,110,90,90,100,90,
90,100,80,80,70,80,80,100,90,
80,90,80,70,65,70,80,90,80,
80,90,80,70,60,70,80,90,80);

blackCannonPositionValue = Arrays.asList(
80,90,80,70,60,70,80,90,80,
80,90,80,70,65,70,80,90,80,
90,100,80,80,70,80,80,100,90,
90,100,90,90,110,90,90,100,90,
90,100,90,110,130,110,90,100,90,

90,110,90,110,130,110,90,110,90,
90,110,90,110,130,110,90,110,90,
100,120,90,80,80,80,90,120,100,
110,125,100,70,60,70,100,125,110,
125,130,100,70,60,70,100,130,125);
```