CSE 537 Project 2
Connect K: Game Search Report

[Image Source: http://ecx.images-amazon.com/images/I/61QZQl1GGAL.jpg]

Submitted By:

Vinayak Mittal **Alpit Kumar Gupta**

SBU ID: 110385943 **SBU ID: 110451714**

PREFACE

We have ***implemented all the mentioned requirements*** as described in Homework 2 namely:

1. **Minimax Search**
   Implementation of minimax search algorithm is performed in function minimax() of basicplayer.py

2. **Better Evaluation Function**
   Implementation of new evaluation function to find utility value. The function is described in new_evaluate() function of basicplayer.py

3. **Alpha-Beta Search**
   Implementation of alpha-beta search algorithm is performed in function alpha_beta_search() of lab3.py .

4. **Generalization of the Game**
   - **Connect-k Problem**
     The Connect four game is generalized to play for multiple k values. Default value is set to 4. For k=3, 4 or 5 the user has to pass kVal parameter in run_game as follows:

     run_game(new_player, basic_player, ConnectFourBoard(kVal=5))

   - **Longest Streak to Win Problem**
     In this model, both the player plays equal number of tokens and the winner is decided based on who has the longest streak after the given number of rounds. To use this functionality, the user has to run the following command:

     run_game(new_player, basic_player, ConnectFourBoard(longStreak=True))

     Default value of this is set to 20 as described in Homework. However, we have added flexibility to provide variable rounds which can be accessed as:

     run_game(new_player,basic_player, ConnectFourBoard(longStreak=True,streakLen=30))

5. **Project Report**

   Project report defines the usage of the project and findings of the game.

1. Introduction

Connect four is a two-player game where one player at a time plays, players have all the information about the moves taken place, and all moves that can take place, for a given game state. Connect four also belongs to the classification of an adversarial, zero-sum game, since a player's advantage is an opponent's disadvantage.

A Connect-Four board is a matrix, laid out as follows having height = 6 and width =7:

```
  0 1 2 3 4 5 6 7
0 * * * * * * * *
1 * * * * * * * *
2 * * * * * * * *
3 * * * * * * * *
4 * * * * * * * *
5 * * * * * * * *
6 * * * * * * * *
```

Two players take turns alternately adding tokens to the board. Tokens can be added to any column that is not full. When a token is added, it immediately falls to the lowest unoccupied cell in the column. The game is won by the first player to have four tokens lined up in a row, either vertically or horizontally, or along a diagonal.

More information of the game can be found at Wiki: https://en.wikipedia.org/wiki/Connect_Four

This project comprises the implementations of following adversarial search techniques:

- *Minimax Search*
  Minimax is a decision rule used in adversarial game search strategies for minimizing the possible loss for a worst case scenario. More information can be found at Wiki: https://en.wikipedia.org/wiki/Minimax

- *Alpha-Beta Search*
  Alpha-Beta search is a special case of minimax search that seeks to decrease the number of nodes that are evaluated by minimax algorithm in its search tree. More about can be found at Wiki: https://en.wikipedia.org/wiki/Alpha%E2%80%93beta_pruning

This project uses following heuristics to evaluate the utility of any possible move in board:

- *Basic Evaluate*
  Basic evaluate is the utility function which checks for the longest streak in the game for the current player and tries to put the token at center locations.

- *New Evaluate*
  New evaluate is the utility function which checks for the number of streaks available in the game board and calculates the winning position based on that streak. If the current player as 3 consequent tokens placed in the board, then it adds high score ensuring better chances for success.

2. FINDING STATS

This project is tested under different circumstances, mainly two of below:

- **New Player vs Basic Player**
- **Alpha-Beta Player vs Basic Player**

Every game has different variants for different values of k in Connect-k and streak length.

New Player Vs Basic Player

In this implementation, new player uses new evaluate utility and implements through minimax search algorithm, whereas the basic player uses the basic evaluate utility and is implemented through minimax algorithm.
At the successful completion of the game, total time taken and number of nodes counted are displayed

1. **run_game(new_player, basic_player)**
   This mode is run for new player vs basic player when connect four is played. Default kVal is set to 4 in constructor. Number expanded is generated for the first player only.

   Output:

```
Win for X!

  0 1 2 3 4 5 6
0 X   X O O X
1 X   O X X O
2 O   X O O X X
3 X X O X X O O
4 X O X O O O X
5 X O O O X O X

Execution Time: 32.7779873286
Total number of nodes Expanded: 24339
```

2. **run_game(new_player, basic_player, ConnectFourBoard(kVal=3))**
   This mode is run for new player vs basic player when connect-k is played for k=3.

   Output:

```
Win for X!

  0 1 2 3 4 5 6
0 O
1 X
2 X
3 O     O
4 X X X O
5 X O O X O X
```

Execution Time: 14.0933356427
Total number of nodes Expanded: 17197

3. **run_game(new_player, basic_player, ConnectFourBoard(kVal=5))**
   This mode is run for new player vs basic player when connect-k is played for k=2.

   Output:

```
Win for  !

  0 1 2 3 4 5 6
0 O X O O X O O
1 X O X X O X X
2 O X O O O X O
3 X X O X X O X
4 X X O O X O X
5 X X O O O O X
```

Execution Time: 44.6356210723
Total number of nodes Expanded: 27897

4. **run_game(new_player, basic_player, ConnectFourBoard(longStreak=True))**
   This mode is run for new player vs basic player when connect four is played for given token set to 20.

   Output:

```
Win for O!

  0 1 2 3 4 5 6
0     X X
1     X O
2     O X
3     X O O
4 X   X O O
5 X O O O X O X
```

Execution Time: 10.6278039716
Total number of nodes Expanded: 24957

Alpha-Beta Player vs Basic Player

In this implementation of game, alpha-beta player uses the new evaluate utility function and have been implemented through alpha beta pruning search technique. This player plays the first turn against the basic player which use basic evaluate utility and implemented through minimax algorithm.

At the successful completion of the game, total time taken and number of nodes count are printed:

5. **run_game(alphabeta_player, basic_player)**
   This mode is run for alphabeta player vs basic player when connect four is played.
   Output:

```
Win for X!

  0 1 2 3 4 5 6
0 X   X O O X
1 X   O X X O
2 O   X O O X X
3 X X O X X O O
4 X O X O O O X
5 X O O O X O X
```

Execution Time: 23.6668308861
Total number of nodes Expanded: 7185

6. **run_game(alphabeta_player, basic_player, ConnectFourBoard(kVal=3))**
   This mode is run for alphabeta player vs basic player when connect-k is played for k=3.

   Output:

```
Win for X!

  0 1 2 3 4 5 6
0 O
1 X
2 X
3 O     O
4 X X X O
5 X O O X O X
```

Execution Time: 9.6299349751
Total number of nodes Expanded: 3141

7. **run_game(alphabeta_player, basic_player, ConnectFourBoard(kVal=5))**
   This mode is run for alphabeta player vs basic player when connect-k is played for k=5.

   Output:

```
Win for  !

  0 1 2 3 4 5 6
0 O X O O X O O
1 X O X X O X X
2 O X O O O X O
3 X X O X X O X
4 X X O O X O X
5 X X O O O O X
```

Execution Time: 32.137486716
Total number of nodes Expanded: 6279

8. **run_game(alphabeta_player, basic_player, ConnectFourBoard(longStreak=True))**
   This mode is run for alphabeta player vs basic player when connect four is played for defined number of token which is by default set to 20.

   Output:

```
Win for O!

  0 1 2 3 4 5 6
0     X X
1     X O
2     O X
3     X O O
4 X   X O O
5 X O O O X O X
```

Execution Time: 8.01462023192
Total number of nodes Expanded: 6605

3. **NEW EVALUATION UTILITY**
   New evaluation utility is more aggressive utility as compared to basic evaluate utility and works on the principle of finding favorable streaks in the board for the current player.

   This utility function check number of streaks present on the board which have more chances of winning the game as opposed to just playing randomly on the board. It finds two major streaks for connect K i.e. continuous streak of k-1 and k-2.

   The value of k is defined as the connect-k problem. If no value is explicitly specified then it is by default set to 4. Then it evaluates the following information

1. A = number of consecutive k-1 streaks found in the board for the current player.
2. B = number of consecutive k-2 streaks found in the board for the current player.
3. C = number of consecutive k-1 streaks found in the board for opposite player.
4. D = number of consecutive k-2 streaks found in the board for opposite player.

All these values are looked for all possible winning directions as described by the rule of playing connect-k. Therefore, it checks for consecutive streaks in following directions:

1. Horizontal
2. Vertical
3. Diagonal
    - Left Diagonal
    - Right Diagonal

After calculating all these values, the utility value is set by the following formula:

$$Utility\text{-}Value = A* 10000 + B*100 – C*100000 – D*100$$