Assignment 1 Report **Yunjie Cao and Minjia Jin**

Search: The Othello Game

Introduction

This program allows the user to play a full Othello game against a computer. The computer's moves are made according the search result, which is obtained using the Minimax algorithm together with Alpha-Beta pruning. The person's moves are provided to the program through keyboard input. We can select for the computer the colour it will play and set a time limit for its response. When it is the person's turn to move, the program will list all the legal moves for the person. After every move, the program will show the current state of this game. In the end, the program will announce the winner and exit gracefully.

Program Input

Upon running this program, the user selects the colour the computer should play as. This is done by entering a number: 1 means the computer plays as the white player and 2 means it is the dark player. After that, the user defines the time limit within which the computer should respond. The input is an integer and it determines the search depth.

Now, the Othello game starts. In every round, the user's move is provided to the program using the keyboard, where the input is the standard move notation, i.e. ”X Y” where X is one of a-h (column) and Y is one of 1-8 (row). For example, ”a 5” is a legal input. Note that there is a space between the character (a-h) and the integer (1-8).

Program Output

There are two parts to the output of the program.

First, before the user makes a move, the program will present all the possible moves for the user.

Second, after every move, the current state of the game will be shown in the form of a grid. The symbols used are '#', which means blank, 'd', which means a dark piece, and 'w', which means a white piece. An example of the game state visualisation is shown in Fig 1.

Design and Implementation

0.1 Algorithm and Evaluation Function

In this program, the computer's optimal move is calculated by the search algorithm. Minimax adversarial search algorithm with Alpha-Beta pruning introduced is used to generate the computer's moves. When it is the computer's turn to move, it will iterate

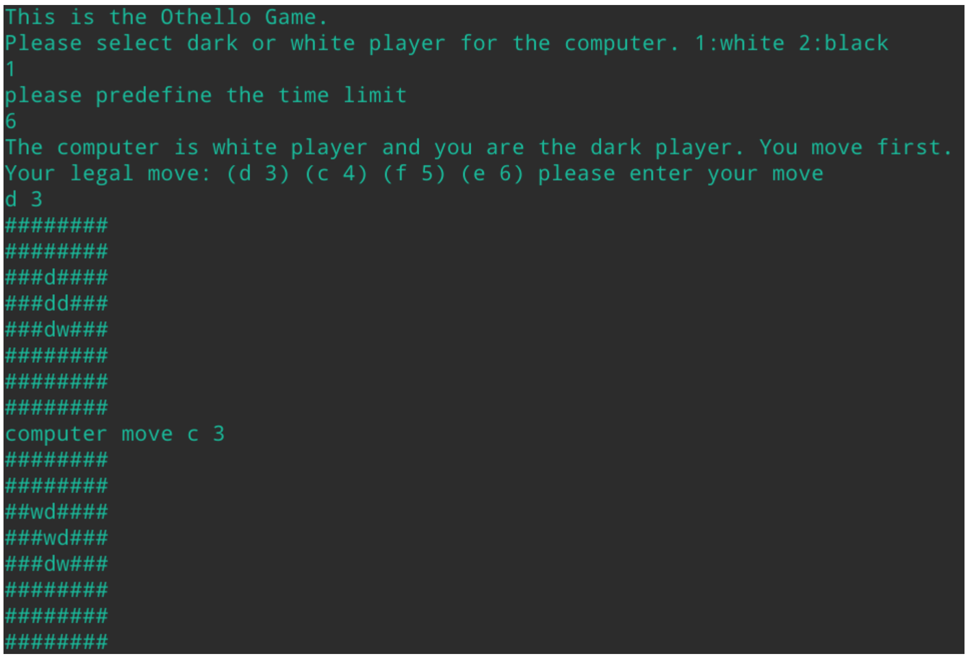
```
This is the Othello Game.
Please select dark or white player for the computer. 1:white 2:black
1
please predefine the time limit
6
The computer is white player and you are the dark player. You move first.
Your legal move: (d 3) (c 4) (f 5) (e 6) please enter your move
d 3
########
########
###d####
###dd###
###dw###
########
########
########
computer move c 3
########
########
##wd####
###wd###
###dw###
########
########
########
```

Figure 1: Output example

through all its possible moves and use the search algorithm to obtain the heuristic value of this move. It will then choose the move that gives it the best value - the largest value obtained - as its next move. The evaluation function is defined as the following:

$$V(node) = N_{computer} - N_{person} \tag{1}$$

$N_{computer}$ means the number of pieces held by the computer and N_{person} means the number of pieces held by the person.

0.2 Implementation

Assume that the computer is the white player and the person is the dark player. As the dark player starts first, the person moves first and the computer moves next. The implementation of this program can be shown in algorithm 1

Algorithm 1: Game procedure

```
repeat
    GenerateLegalMoves(); //generate possible moves for the person
    PersonInput(); //person inputs his move
    ChangeBoard(); //update the board according to person's move
    Visualize(); //visualise the current state of the board
    ComputerMove(); //search for a move for the computer
    ChangeBoard(); //update the board according to computer's move
    Visualize(); //visualise the current state of the board
until End of the game;
```

The most important elements used in best computer move calculation are implemented in the function *ComputerMove*(). In the program, the computer is always the maximiser, in that it selects moves that maximise the score of the game at the depth limit. This is facilitated by the evaluation function as above in *evaluation*(). To do so, the computer analyses the potential moves by the user in response to its own potential move, and derives the minimum score among the user's N potential moves, say x_n (for n = 1..N). Then the computer chooses the move that gives it the highest score among the x_n scores. This is how the Minimax algorithm used in this program. Alpha-Beta pruning is also introduced to improve the efficiency of the search and a depth limit is set according to the predefined time limit to limit the depth of search.

Details of this program can be found in the Assignment1.cpp file, which is in the directory: /h/d3/x/mi3774ji-s/Documents/AppliedAI/assignment1. Using the command ./Assignment1, this program can be run and tested in the case that the input is as described in the Program Input section and all the inputs are valid, i.e. the character is between a-h and it is lowercase.