Universidade de São Paulo

Engenharia da Computação

Introdução à Ciência de Computação I

C Quarto!

Aluno:
Felipe Tiago De Carli

Professor:
Dr Adenilso da Silva Simão

April 19, 2018

1 Introdução

Quarto! é um jogo de soma-zero onde há 16 peças distintas a serem jogadas em 16 posições diferentes. Originalmente, as peças são representadas por serem altas ou baixas, brancas ou pretas, com furo ou sem furo e redondas ou quadradas. O jogo é dito ganho quando quatro peças com pelo menos uma mesma caracterísitica são alinhadas em um tabuleiro 4x4.

Devido ao fato das peças serem representadas por características binárias, as peças podem ser representadas por números binários, e o jogo é ganho quando uma linha, coluna ou diagonal tem a operação AND retornando 1 ou a operação OR retornando 0.

2 Descrição do projeto

2.1 Ambiente de desenvolvimento

O código foi escrito tanto na plataforma Linux quanto em Windows. Foi utilizado a IDE Sublime-Text nos ambos sistemas operacionais. Foi também utilizado a plataforma Git para controle de versão.

2.2 Compilador usado

Em Windows, foi utilizado o compilador GCC 5 Series do MinGW x86.

Em Linux, foi utilizado o compilador GCC 7.3.

Não houve nenhum problema quanto a portabilidade do código entre os dois compiladores.

2.3 Códigos Fonte

Os seguintes arquivos fazem parte do código fonte do programa:

1. main.c
2. play.c
3. board.c
4. ai.c

5. play.h
6. board.h
7. ai.h
8. options.h

3 Tutorial

3.1 Tutorial de Compilação

Com o GCC instalado, execute o seguinte comando na pasta do projeto:

```
$ gcc -o main main.c play.c board.c ai.c
```

Também é possível compilar o programa utilizando o CodeBlocks, desde que haja um compilador de C instalado na máquina.

3.2 Tutorial de Execução

O programa aceita duas entradas (flags) que definem se o jogo será jogado contra uma IA e se a IA ou o jogador começara jogando. De modo padrão, o programa iniciará no modo multiplayer. Se a flag para que a IA comece jogando não estiver presente, o jogador começara o jogo por padrão.

O seguinte comando executa o programa no modo Jogador vs IA com a IA responsável pela primeira jogada:

```
$ ./main -ai -aib
```

É possível utilizar o CodeBlocks para compilar e executar o programa. Basta que todos os arquivos estejam na mesma pasta do projeto e que um compilador de C esteja instalado. Para executar com argumentos, siga Project - Set programs' arguments...

4 Entrada teste

A entrada teste inicia o jogo com a primeira jogada dada para a IA, e com o tabuleiro representado pela Figura 1. Nesse teste, a IA deve encontrar uma posição otimizada para a peça 0011.

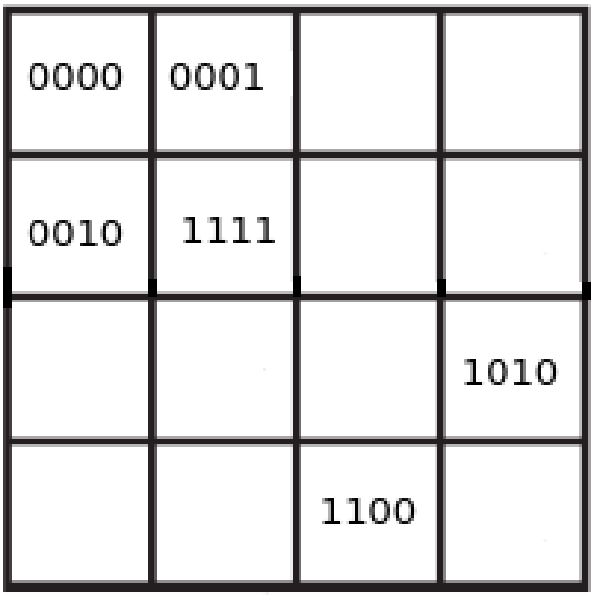

Figure 1: Demonstração do tabuleiro com uma entrada teste do programa.

5 Análise da Inteligência Artificial

O algoritmo utilizado para a inteligência artificial foi o algoritmo de busca minimax. Esse algoritmo percorre toda a árvore de jogo, assumindo que o jogador também jogue posições otimizadas.

Por ser um algoritmo de complexidade O(n!), o tempo de processamento pode ser muito longo se for utilizado logo nas primeiras jogadas.

Foram feitas medições e gráficos do tempo de processamento do minimax utilizando a entrada teste definida na ultima seção. Foi utilizado um processador i5 5500U para a execução. O gráfico foi gerado com Python utilzando a biblioteca matplotlib. O código que gera o gráfico está na pasta utils do diretório do projeto. O gráfico Tempo x Jogadas Faltantes é mostrado mais abaixo.

6 Limitações

O código da AI foi desenvolvido utilizando uma simples tomada de decisão até que o jogo atinja um determinado nível. A partir daí, a AI passará a uti-

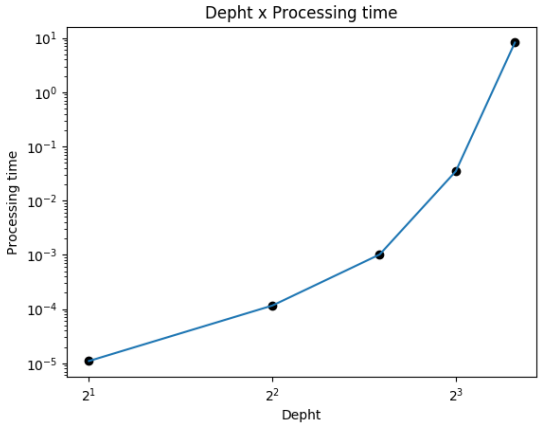

Figure 2: Gráfico Tempo x Jogadas Faltantes em escala logarítmica, utilizando o algoritmo minimax.

lizar o algoritmo minimax para decidir a próxima jogada. A razão para que o minimax não seja utilizado desde a primeira jogada é de que seria computacionalmente inviável calcular e avaliar todas as jogadas possíves (Minimax possuí complexidade O(n!)). Logo, o algoritmo da AI não é perfeito, e vale-se somente da sua tomada de decisão por simples decisão até que comece a criar jogadas otimizadas. Pode-se melhorar o algoritmo da AI diminuindo esse nível em que o algoritmo minimax é utilizado, ao custo de um tempo maior para que o algoritmo tome uma decisão.

A primeira tomada de decisão utilizando o algoritmo do minimax ocorre quando há menos de 11 posições vazias no tabuleiro. Nessa primeira decisão, o algoritmo pode tomar mais de um minuto para fazer a jogada. A partir daí, o algoritmo toma uma decisão quase que instantaneamente.