Trabalho Final de IA

Jefferson Stoffel, João Gross

Características Gerais do Programa

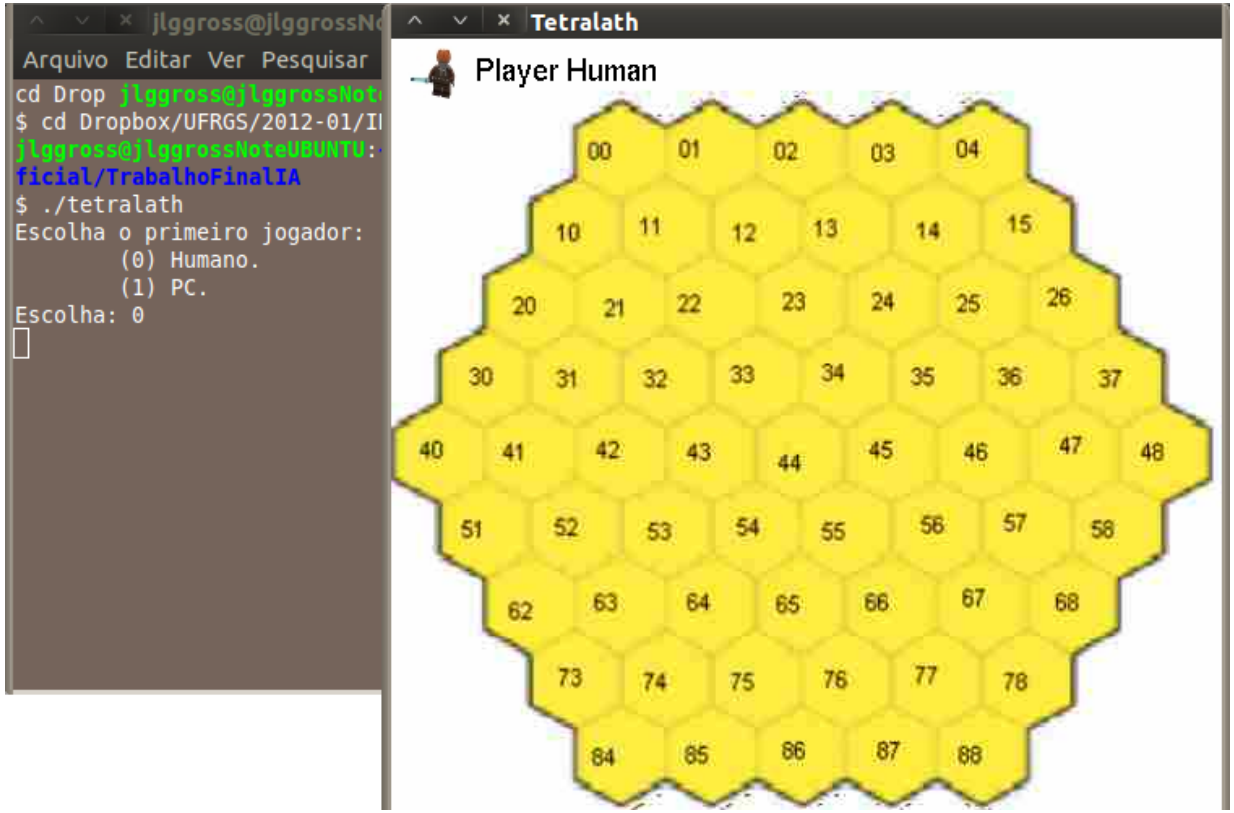
Interface

- Simples e intuitiva
- Fácil utilização

```
$ ./tetralath
Escolha o primeiro jogador:
        (0) Humano.
        (1) PC.
Escolha: 0
```

Escolha do primeiro jogador

Características Gerais do Programa



Interface carregada após a escolha do primeiro jogador

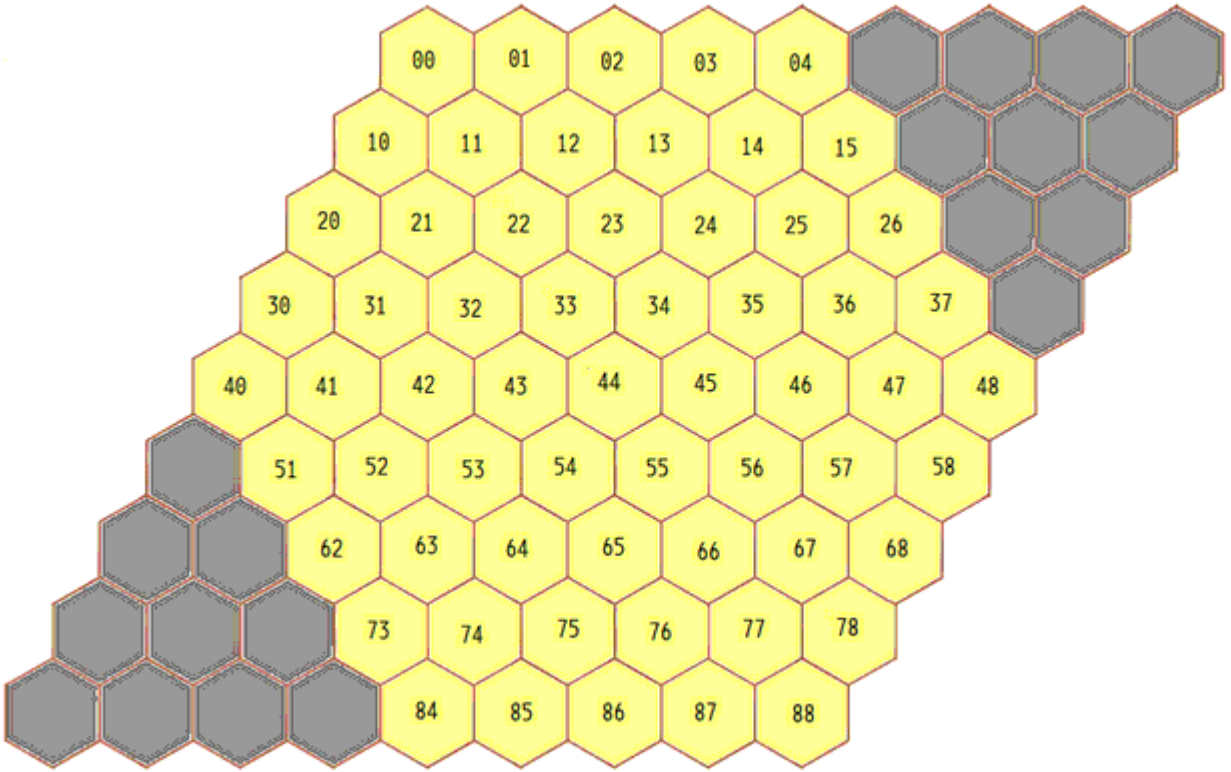
Estruturas de dados do tabuleiro

Posição do tabuleiro

```
typedef struct {
    int color;
    int neighbors[NEIGHBORS];
} Position;
```

- color: inicializada com NO_COLOR para posições válidas e INVALID para invávlidas.
- neighbors[]: vetor com os ids das posições das peças vizinhas.

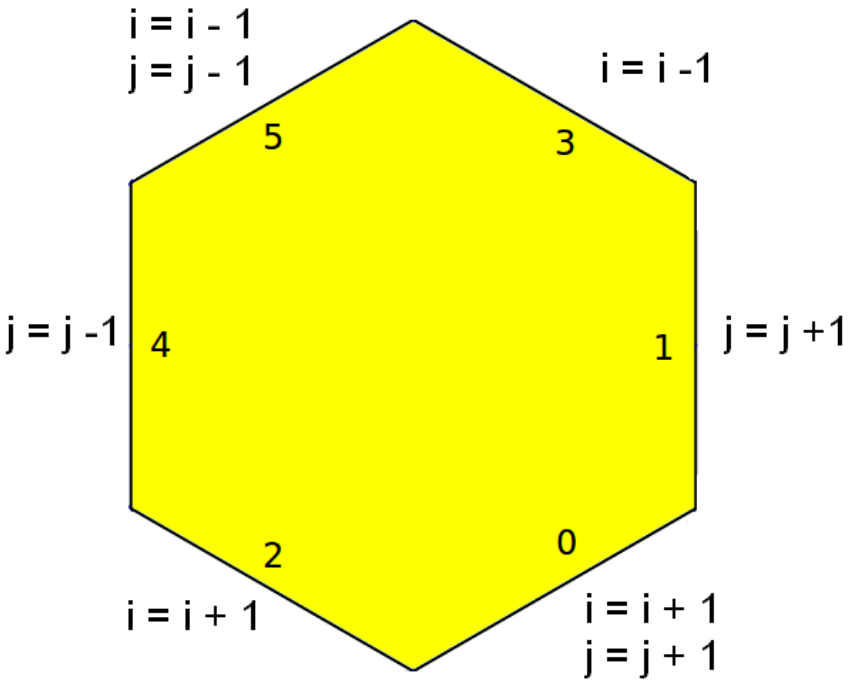
Estruturas de dados do tabuleiro



Matriz 9 x 9 para representação do tabuleiro

Lógicas de manipulação das posições

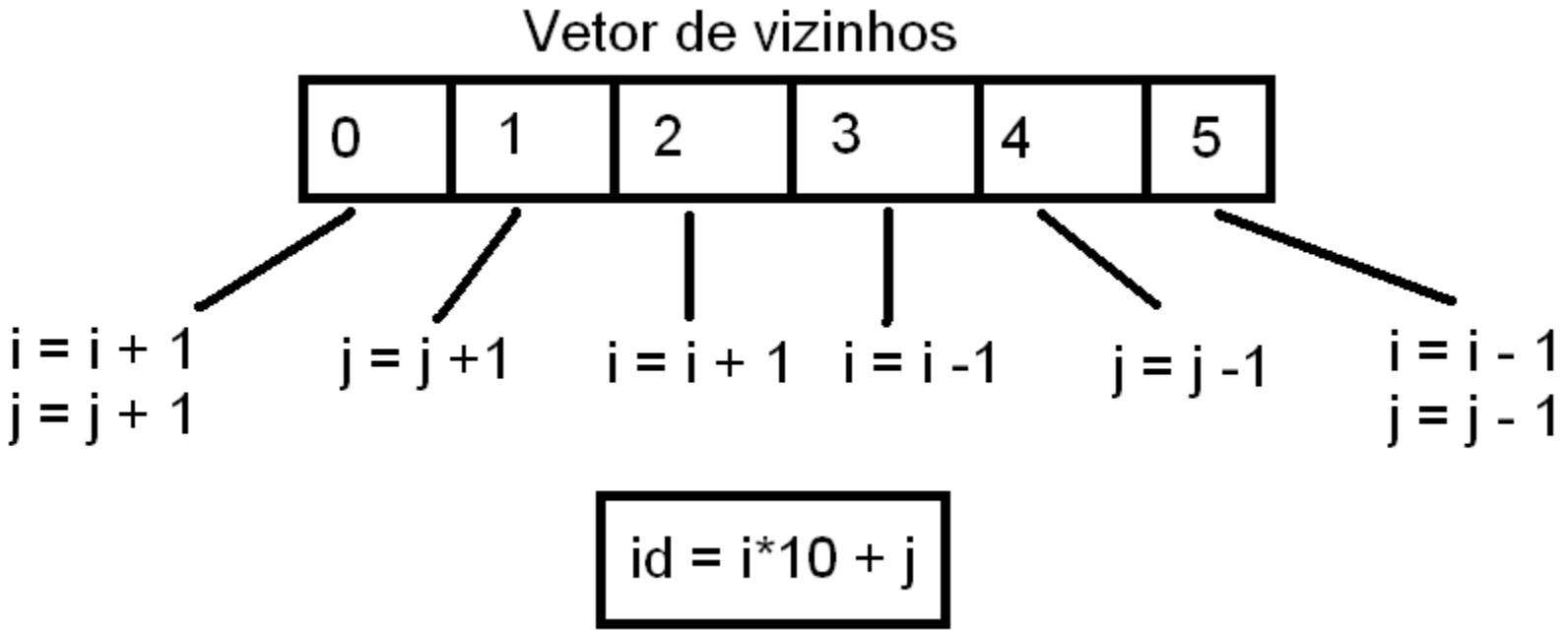
Vizinhos de uma posição



i e j são os índices da posição no tabuleiro

Lógicas de manipulação das posições

Vetor 'neighbors[]'



Representação dos vizinhos no vetor de vizinhos

Funções de manipulação do tabuleiro

Função de criação

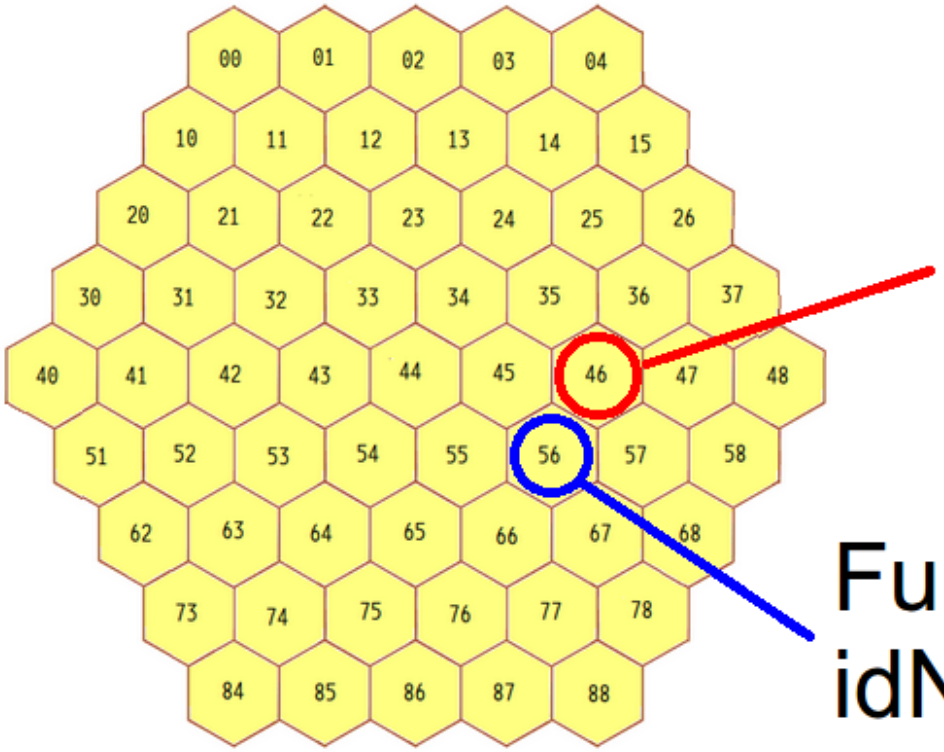
- Aloca-se memória para a matriz 9x9.
- Aqui não é feita nenhum tipo de inicialização.

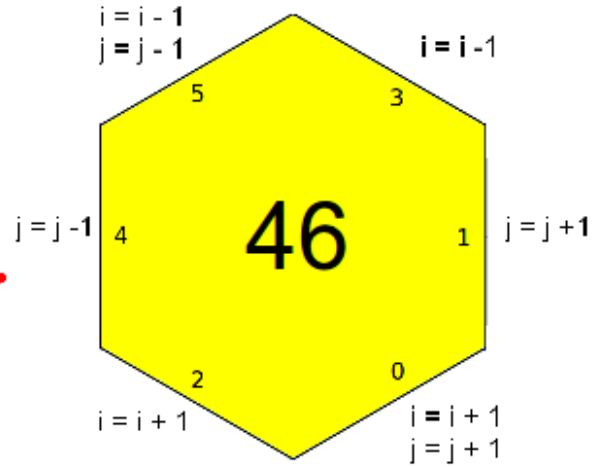
Função de inicialização

- variável 'color' dos elementos válidos inicializada com NO_COLOR. Posições inválidas com INVALID.
- o vetor 'neighbors' é inicializado com o id do vizinho de uma posição. Se um vizinho for uma posição inválida o valor nicializado é INVALID.

Funções de manipulação do tabuleiro

Função auxiliar para obtenção do id



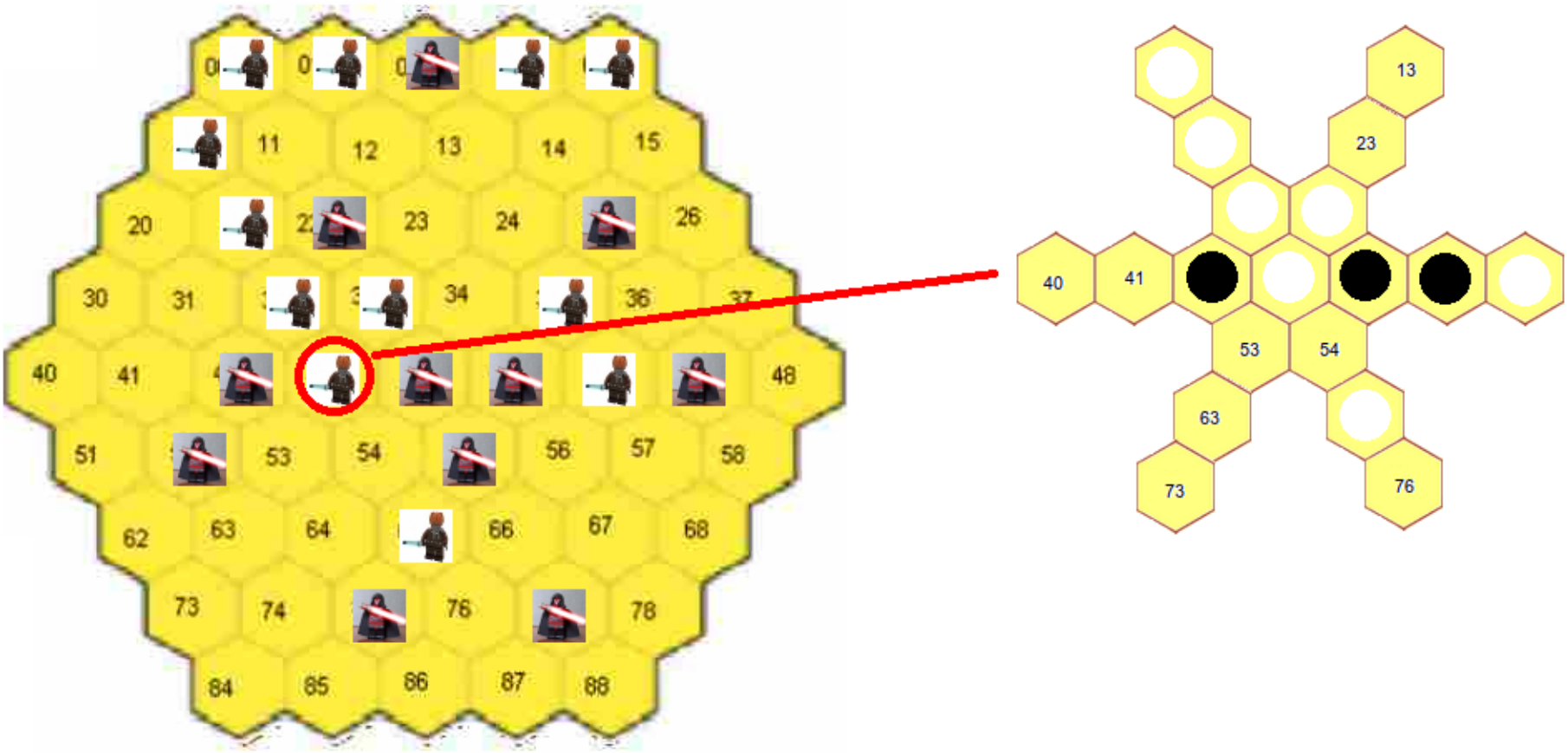


Função:
idN = idNeighbor(46, 2)

Função de Avaliação

- Estratégia de fácil entendimento.
- Dado o conjunto de peças marcadas de um jogador, ela pega cada peça deste jogador, marcada no tabuleiro, e varre os seus vizinhos.
- Cada vizinho é considerado no algoritmo como uma caminho de varredura.
- Cada caminho é varrido em linha, retornando um valor associado a jogada mais favorável à IA.

Função de Avaliação



como a função de avaliação ‘vê’ uma posição

Minimax

```
seedMinimax{
    se testTwoEmptyOne
        retorna jogada
    senão
        α ← -∞
        para cada espaço livre no tabuleiro faça
            α ← max(α, expandTree(tabuleiro, profundidade, adversário)))
        fim para
        retorna α
    fim senão
}
```

Minimax

- testTwoEmptyOne: função que verifica se um dos jogadores está prestes a fechar uma linha de quatro peças consecutivas.
  - Se a IA estiver nesta situção, sua próxima jogada será de vitória.
  - Se o adversário esteja nesta situação, a próxima jogada da IA será uma ação de bloqueio.

- A função expandTree é efetivamente o algoritmo de minimax.

Minimax

- A função expandTree é efetivamente o algoritmo de minimax

```
expandTree(tabuleiro, profundidade, jogador) {
    avalia se alguém ganhou
    se alguém ganhou retorna vitória para aquele jogador
    senão
        se profundidade > 0
        … (próximo slide) …
        senão
            realiza função de avaliação
            retorna resultado da função
        fim senão
    fim senão
```

Minimax

- expandTree (cont.)

```
se é a jogada do adversário
        α ← +∞
        para cada espaço livre no tabuleiro faça
                marca tabuleiro
                α ← min(α, expandTree(tabuleiro, profundidade - 1, ¬jogador)))
        fim para
        retorna α
fim se
```

… (próximo slide) …

Minimax

- expandTree (cont.)

```
senão
    α ← -∞
    para cada espaço livre no tabuleiro faça
        marca tabuleiro
        α ← max(α, expandTree(tabuleiro, profundidade - 1, ¬jogador)))
    fim para
    retorna α
fim senão
```

Minimax

Profundidade alcançada

- Para permanecer dentro do limite de 5 segundos, a máxima profundidade alcançada é três. Com o evoluir do jogo, esse número poderia aumentar devido a queda do número de ramificações, porém não é o caso do programa que construímos, que trabalha com um valor constante.

Minimax

Tipo de otimização utilizada

- Poda alpha-beta

evita que sejam calculados ramos que não trarão nenhum novo resultado para o minimax.

- função “testTwoEmptyOne”
  - avalia se existe um jogador prestes a fechar um linha de quatro peças consecutivas.
  - evita que o minimax seja aberto para uma jogada óbvia.

Análise do desempenho no campeonato

Primeira rodada

- algoritmo do minimax não estava funcionando
- não conseguimos arrumá-lo a tempo.
- nesta fase a IA voluntariamente marcava 3 peças em linha, gerando uma situação de derrota, o que nos gerou a maioria das derrotas.

Análise do desempenho no campeonato

Segunda rodada

- minimax foi arrumado apenas em parte.
- IA não marcava mais voluntariamente situações de derrota, porém a marcação das peças seguia em ordem sequencial no tabuleiro.
- Conseguimos também implementar bloqueios de vitória do adversário e também a gerar a vitória da IA.

Análise do desempenho no campeonato

- Após o campeonato analisamos os códigos e descobrimos a causa do problema.
- No algoritmo de minimax houve uma troca nas funções de chamadas: quando era calculado Max, era retornando Min com seu valor de inicialização e vice-versa.
- Desta forma, todos os nós folhas tinham o mesmo valor, o que ocasionava a escolha sempre do primeiro valor, e por essa razão o algoritmo marcava o tabuleiro de forma sequencial.