Relatório Bots

Algoritmo minMax

- Estrutura de Dados:

Para processar e guardar informação relativa às posições do tabuleiro, foi escolhida uma estrutura de lista ligada - *listTpl* - que contem uma estrutura *pair e* um apontador para a *listTpl* seguinte. (Fig. 1)

A estrutura *pair* tem 2 inteiros, x e y, que guardam as linhas e as colunas das posições, respetivamente, de acordo com o critério existente nas funções que usam estas estruturas.

```
typedef struct pair{
   int x;
   int y;
}coordenadas;

typedef struct listTpl *Ltpl;
typedef struct listTpl {
   struct pair valor;
   Ltpl next;
}nodo;
```

Fig. 1 – Estrutura listTpl

- Algoritmo:

A função *minmax* (Fig. 2) recebe uma lista com as posições em que é possível jogar.

Para cada uma das posições, é determinado o número de peças que são viradas ao jogar na posição.

De seguida, é simulado as consequências da jogada, e são analisadas todas as jogadas possíveis do oponente.

É depois subtraído ao número calculado anteriormente o número de peças viradas pela melhor jogada do adversário.

É escolhida a posição com a maior diferença entre as peças viradas pelo jogador e as peças viradas pela melhor jogada do oponente.

```
coordenadas minmax (ESTADO *e, Ltpl jogadas_possiveis){
   STACK stack;
   struct estado inicial = *e;
   int acc, res;
   coordenadas resultado,posicao;
  minmax_inicializacao(&res,&acc,&resultado,e,jogadas_possiveis,&posicao,stack);
   resultado = jogadas_possiveis->valor;
   *e = inicial;
   if (jogadas_possiveis != NULL)
      jogadas_possiveis = jogadas_possiveis->next;
   while (jogadas_possiveis != NULL){
      minmax_loop(&res,&acc,&resultado,&posicao,e,jogadas_possiveis,stack);
      *e = inicial;
      jogadas_possiveis = jogadas_possiveis->next;
   }
   return resultado;
}
```

Fig. 2 – Função minmax

Bot Nível 1

O bot de nível 1 **(Fig. 3)** usa o algoritmo *minmax* para escolher em um quarto das suas jogadas, das posições onde é possível o bot o jogar (usando a função *posicoes_possiveis),* a posição em que vai ser colocada a peça.

De seguida, guarda em dois valores inteiros, i e j, os valores produzidos, incrementa-os (para atualizá-los para o input de 1 a 8, como os utilizadores usam) e atualiza o ESTADO na posição escolhida pelo algoritmo através da função *executa_jogada*.

```
void bot_facil(ESTADO *e,STACK *pointer){
  Ltpl aux = posicoes_possiveis(e);
  int i,j;
  if(nmr_jogadas(e)%4 == 0) {
    coordenadas minmax_xy = minmax(e, aux);
    i = minmax_xy.x;
    j = minmax_xy.y;
  } else {
    i = aux->valor.x;
    j = aux->valor.y;
  }
  i++, j++;
  executa_jogada(e, i, j, pointer);
}
```

Fig. 3 – Função bot_facil

Bot Nível 2

O bot de nível 2 **(Fig. 4)** usa o algoritmo *minmax* para escolher em metade das suas jogadas, das posições onde é possível o bot o jogar (usando a função *posicoes_possiveis),* a posição em que vai ser colocada a peça.

De seguida, guarda em dois valores inteiros, i e j, os valores produzidos, incrementa-os (para atualizá-los para o input de 1 a 8, como os utilizadores usam) e atualiza o ESTADO na posição escolhida pelo algoritmo através da função *executa_jogada*.

```
void bot_medio(ESTADO *e,STACK *pointer){
  Ltpl aux = posicoes_possiveis(e);
  int i,j;
  if(nmr_jogadas(e)%2 == 0) {
    coordenadas minmax_xy = minmax(e, aux);
    i = minmax_xy.x;
    j = minmax_xy.y;
  } else {
    i = aux->valor.x;
    j = aux->valor.y;
  }
  i++, j++;
  executa_jogada(e, i, j, pointer);
}
```

Fig. 4 – Função bot_medio

Bot Nível 3

O bot de nível 3 **(Fig. 5)** usa o algoritmo *minmax* para escolher, das posições onde é possível o bot o jogar (usando a função *posicoes_possiveis),* a posição em que vai ser colocada a peça.

De seguida, guarda em dois valores inteiros, i e j, incrementa-os (para atualizá-los para o input de 1 a 8, como os utilizadores usam) e atualiza o ESTADO na posição escolhida pelo algoritmo através da função *executa_jogada*.

```
void bot_minmax(ESTADO *e,STACK *pointer){
  Ltpl aux = posicoes_possiveis(e);
  int i,j;
  coordenadas minmax_xy = minmax(e, aux);
  i = minmax_xy.x;
  j = minmax_xy.y;
  i++, j++;
  executa_jogada(e, i, j, pointer);
}
```

Fig. 5 – Função bot_minmax