Terceiro EP
Paper Boxing

Alfredo e Gubi

2 de dezembro de 2009

Este EP consiste na implementação de um jogo de lápis e papel chamado "*Paper Boxing*", criado pelo especialista american Sid Sackson [1]. Apesar de simples, não conheço uma estratégia segura para vencer. Uma descrição detalhada encontra-se no livro "*A Gamut of Games*"

A tarefa é fazer um programa capaz de jogar *Paper Boxing* de forma autônoma, de acordo com as regras descritas na seção 1 e seguindo a interface em 2. O projeto pode ser feito em **duplas** e o time vencedor ganhará **média 10**. Ao final faremos uma contenda entre os campeões de cada classe, pelo cinturão de ouro.

1 O jogo

Descrição retirada e adaptada do livro original:

É um jogo para dois jogadores, um é o **PAR** e o outro **ÍMPAR**. Cada jogador constrói uma matriz 4×4, sendo que o elemento superior esquerdo é o de partida. No jogo de lápis e papel, esta quadrícula é marcada com um 'S' (*Start*), na nossa versão, pode ser um 0. O restante da matriz contém os números de 1 a 15, na ordem que o jogador achar mais conveniente.

Depois de derminadas as matrizes, cada jogador expõe sua escolha ao outro. Desta forma os dois jogadores tem a informação completa da distribuição sua e do adversário.

Cada jogador "se posiciona" sobre sua casa 'S' (como dissemos, na posição (0,0)). Os valores das casas inferiores direitas (posição (3,3)) são somados e o

[1] `http://en.wikipedia.org/wiki/Sid_Sackson`

resultado indica qual jogador iniciará a partida, se o resultado por par, **PAR** começará, senão **ÍMPAR** inicia. Cada rodada se desenvolve da seguinte forma:

- O primeiro jogador então escolhe uma das casas vizinhas *ainda não visitadas*, incluindo as diagonais.
- O segundo jogador, após ver a jogada do primeiro, faz o mesmo.
- Aquele que estiver sobre o maior valor ganha o *round*, marcando um ponto.
- O jogador que ganhar começa o *round* seguinte. No caso de empate, o mesmo jogador que começou o *round* começará o próximo.

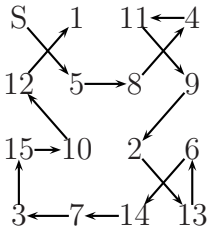
Ao final, quem fizer o maior número de pontos vence. No caso de empate, ganha aquele que começou a partida, já que isso é uma pequena desvantagem.

Se um jogador se descuidar e atingir uma posição em que não tenha jogadas possíveis, com todas as casas vizinhas já visitadas, ele perderá por *knock-out*, mesmo que tenha vantagem nos pontos[2].

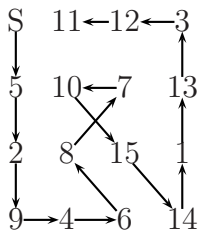
1.1 Exemplo

Acompanhe as jogadas para entender.

PAR



ÍMPAR



[2]Óbviamente no 15º round esta condição acontecerá para os dois jogadores e não deve ser levada em conta.

Placares parciais:

round	**PAR**	**ÍMPAR**	Resultado
1	5	5	Empate
2	8	2	PAR
3	4	9	ÍMPAR
4	11	4	PAR
5	9	6	PAR
6	2	8	ÍMPAR
7	13	7	PAR
8	6	10	ÍMPAR
9	14	15	ÍMPAR
10	7	14	ÍMPAR
11	3	1	PAR
12	15	13	PAR
13	10	3	PAR
14	12	12	Empate
15	1	11	ÍMPAR

Resultado final 7 a 6 para **PAR**.

2 Interface em C

Seu programa deve ler um caractere da entrada para saber se é PAR ou ÍMPAR, se o caractere for 'p', é PAR.

O programa PAR deve imprimir sua matriz primeiro e em seguida ler a do adversário. Para imprimir a matriz, basta colocar os números na ordem (0,1) (0,2) .. (3,2) (3,3). Não é necessário imprimir o elemento (0,0).

Se o programa for ÍMPAR, a situação é invertida, primeiro lê a matriz do adversário e depois imprime a sua.

Em cada rodada, o programa que deve iniciar imprime sua jogada e lê a do adversário. Cada programa deve contabilizar os pontos e no final imprimir uma mensagem dizendo que ganhou ou perdeu. Se o programa desistir ou perder por *knock-out* deverá imprimir o valor especial '-1' na sua jogada.

As funções que tratarão da entrada e saída serão fornecidas, para garantir a uniformidade e devido ao pouco tempo restante.

Crie um tipo `typedef int TABULEIRO[4][4];` para representar as matrizes. Crie também um tipo que guarda uma posição:

```
typedef struct {
    int x, y;
} POS;
```

Se você preferir, use um vetor de duas posições.

Construa pelo menos as seguintes funções:

- `void geraTabuleiro(TABULEIRO t);` cria um novo tabuleiro válido;
- `int jogadaValida(TABULEIRO t, POS p);` verifica se a jogada na posição `p` é válida.
- `POS jogada(TABULEIRO meu, TABULEIRO outro, POS minha, POS deles);` retorna a sua próxima jogada,ou o valor `{-1,-1}` caso esteja em *knock-out* ou haja desistência.

Que vença o melhor.

3 Interface em Java

Crie uma classe Tabuleiro que implemente os seguintes métodos da Interface TabuleiroPaperBxing:

- `void geraTabuleiro();` cria um novo tabuleiro válido;
- `void recebeTabuleiro(int [][]t);` a partir de uma matriz 4x4 guarda uma representação do tabuleiro;
- `boolean estaValido();` verifica se o tabuleiro é válido;
- `void imprimeTabuleiro();)` imprime o tabuleiro atual de forma clara, mostrando as jogadas já realizadas;
- `boolean jogada(int [][]j);` recebe duas coordenadas, sendo a primeira correspondente ao movimento horizontal e a segunda ao vertical. Cada coordenada pode ter (-1,0,1). Devolve verdadeiro e efetua a jogada se for válida;

- `boolean estaEmDeadLock();` Devolve verdadeiro se a posição atual é *deadlock*;
- `int posiçãoAtual();` Devolve o valor da posição atual;
- `int [][] devolveJogadas();` Devolve as jogadas já feitas no tabuleiro.

A classe tabuleiro irá necessitar de atributos adicionais a serem definidos por vocês.

Crie uma classe Jogada com o seguinte método:

- `int [][] efetuaJogada(Tabuleiro meu, Tabuleiro adversario);` recebe dois tabuleiros e devolve a próxima jogada.

Algumas classes adicionais serão necessárias para que possa se jogar com um EP. Serão necessários por exemplo métodos para ler um tabuleiro de um jogador e para controlar de quem é a jogada.

Para o torneio será usada apenas a classe Jogada e o seu método `efetuaJogada`.

4 Dicas

Para a competição entre os programas em C e em Java, não é necessário que um programa consiga jogar diretamente com o outro. Um humano ou um programa externo fará a passagem de dados de um programa a outro.

Procure ter certeza que não vai cair em *knock-out*. Procure alguma coisa nos escritos do Djikstra ou olhe o problema do ratinho e do queijo para achar uma saída.

As mesmas regras de controle de cópia que foram usadas para os EPs anteriores continuam válidas para este.