MAC 374 NOÇÕES DE ESTRUTURAS DE DADOS PARA ENGENHARIA

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO – USP

PRIMEIRO EXERCÍCIO-PROGRAMA PRAZO DE ENTREGA: ATÉ 28/01/98

Cruel!

1. Introdução

O objetivo desse exercício-programa é fazer um programa em `Java` que simule o `Cruel` que é um jogo de paciência para uma jogador. Você deve entregar um disquete contendo o programa fonte e uma listagem do seu programa até o dia 28 de janeiro (guarde cópias do programa que você entregar).

2. Cruel

Cruel é um jogo solitário de cartas jogado com 1 baralho. As regras a seguir correspondem ao jogo `Cruel` de Ken Sykes, conforme descrito na implementação do jogo no *Windows Entertainment Pack for Windows*. Essas regras são quase uma tradução literal das regras encontradas no *help* do jogo.

3. Descrição do jogo

A área de jogo é dividida em duas regiões: uma com 4 *pilhas de naipes* e outra com 12 *pilhas de cartas*. O objetivo do jogo é mover todas as cartas das 12 pilhas de cartas para as pilhas de naipes.

No início de cada partida, os 4 ases são colocados nas pilhas de naipes e as 48 cartas restantes são distribuídas aleatoriamente nas 12 pilhas de cartas, com cada pilha contendo exatamente 4 cartas.

(Pilha de carta → pilha de naipe) Podemos mover uma carta do topo de uma pilha de cartas para uma pilha de naipes, desde que a carta movida seja do naipe correspondente e um 'número' maior do que a carta no topo da pilha de naipes. Por exemplo, se o 4 de ♠ (espadas) estiver no topo da pilha do naipe de espadas estão podemos mover o 5 de ♠ do topo de uma 12 pilhas para o topo da pilha do naipe de espadas.

(Pilha de carta → pilha de carta) Podemos mover uma carta do topo de uma das 12 pilhas de cartas para o topo de uma das outras pilhas cartas, desde que as cartas tenham o mesmo naipe e a carta movida seja um 'número' menor do que a carta do topo da outra pilha. Por exemplo, podemos mover

Date: 6 de janeiro de 1998.

o Valete de $\heartsuit$ (copas) do topo de uma das pilhas de cartas para uma pilha de cartas que tenha a Rainha de $\heartsuit$ no topo.

Quando você não conseguir mais movimentar as cartas, você pode *redistribuir* nas 12 pilhas. A operação *redistribuir* rearranja as cartas nas pilhas de cartas, mantendo 4 cartas no maior número possível de pilhas, sem mudar a ordem consecutiva de nenhuma das cartas. Por exemplo, se você moveu cartas de uma pilha a outra de forma a que 3, 4 e 5 de Ouros ficaram juntas, uma redistribuição irá manter essa ordem intacta. Assim, após a operação resdistribuir, no máximo uma pilha ficará com menos do que 4 cartas e as demais pilhas terão 4 ou 0 (zero) cartas. Numa redistribuição o número de pilhas com cartas pode diminuir ou aumentar.

O jogo termina quando todas as cartas foram movidas para as pilhas de naipes ou quando nenhuma carta pode ser movimentada entre as pilhas depois de uma redistribuição.

4. Estratégia

As seguintes estratégias são recomendadas:

- Mova as cartas para as pilhas de naipes tão cedo quanto possível; e
- Quando movimentar cartas entre pilhas de cartas, tente mover as cartas maiores primeiro.

5. Pontuação

Cruel scoring is simple. Your score is the number of cards remaining in the stacks at the end of the game.

6. Especificações do programa

Faça um programa em `Java`, que simula o jogo `Cruel`. O seu programa deve permitir que diversas partidas sejam jogadas.

Num baralho as cartas estão divididas em 4 naipes:

$\diamondsuit$: ouros (vermelho);
$\heartsuit$: copas (vermelho);
$\spadesuit$: espadas (preto); e
$\clubsuit$: paus (preto).

Para cada naipe existe uma seqüência de 13 cartas, a saber: A (Ás), 2, 3, 4, 5, 6, 7, 8, 9, 10, J (Valete), Q (Dama) e K (Rei). Escolha uma implementação adequada para representar as cartas.

As 12 pilhas de cartas devem ser necessariamente armazenadas em um único vetor com 48 posições. Para movimentação das cartas, use a estratégia descrita nas sugestões acima (ou invente algumas outras estratégias).

Para o tratamento de overflow em cada uma das pilhas você pode utilizar, com as devidas adaptações, o *algoritmo de realocação de tabelas seqüenciais*, descrito nas páginas 245 e 246 do livro de Knuth [1]. Observe que poderão ocorrer muitas colisões entre as pilhas, portanto, seu tratamento deve ser o mais eficiente possível.

Seu programa deve mostrar na tela o conteúdo das pilhas a cada jogada, assim como uma mensagem dizendo qual o movimento que foi realizado. No final da partida, imprima a pontuação obtida. As pilhas de jogo deverão ser mostradas em forma de pilhas, uma ao lado da outra.

Seu programa deverá ser bem documentado, e isso também será avaliado.

Para embaralhar as 'cartas' você vai precisar de um gerador de números aleatórios. Você pode usar o método `random()` da classe `java.lang.Math`.

20.11 The Class `java.lang.Math`

20.11.20 `public static double random()`

The result is a double value with positive sign, greater than or equal to zero but less than 1.0, chosen pseudorandomly with (approximately) uniform distribution from that range.

When this method is first called, it creates a single new pseudorandom-number generator, exactly as if by the expression

`new java.util.Random()`

This new pseudorandom-number generator is used thereafter for all calls to this method and is used nowhere else.

This method is properly synchronized to allow correct use by more than one thread. However, if many threads need to generate pseudorandom numbers at a great rate, it may reduce contention for each thread to have its own pseudorandom number generator.

7. Comentários

Lendo as regras é capaz de você não entender como funciona o jogo. Talvez você queira jogar o `Cruel` no Windows para tentar esclarecer algumas dúvidas. Depois de você jogar no Windows, ainda é capaz de você não entender exatamente como rearranjar as cartas ...

Referências

1. D.E. Knuth, *The art of computer programming, vol 1: Fundamental algorithms*, Addison-Wesley, 1968, QA758 K74a.