Caça aos Pokémons

Data limite de entrega: 04 de outubro de 2016, 23:55

Valor: 10 pontos + 1 para desafio

1 Introdução

O objetivo desse trabalho é rever conceitos básicos de programação e explorar os conceitos de Tipos Abstratos de Dados (TADs) e análise de complexidade, além de exercitar a preparação e submissão de trabalhos práticos em AEDS II.

Iremos implementar uma versão simplificada de um jogo de Pokémons. Em conhecidos jogos de Pokémons, o jogador deve buscar por Pokémons com o objetivo de capturá-los através de um objeto chamado Pokébola. Se o Pokémon não escapar da Pokébola, ele é considerado oficialmente do jogador. O jogador que obtiver o maior número de Pokémons se torna o Mestre Pokémon e pode dominar o mundo!

Você será o responsável pela escolha do(s) Mestre(s) Pokémon em sua região. Assim, você deve simular uma partida entre um número pré-definido de jogadores, e determinar o vencedor. Você deverá criar uma estratégia de jogo em que os treinadores que estão competindo pelo título de Mestre Pokémon deverão maximizar um score. Esse score representa a quantidade de pontos acumulada por cada jogador de acordo com os pokémons capturados, e define o vencedor.

Inicialmente, tem-se um mapa que representa a região de competição. Neste mapa, tem-se vários Pokémons espalhados, bem como Pokéstops. Os Pokéstops são utilizados para a recarga de Pokébolas quando o número de Pokébolas de um jogador estiver se esgotado. Repare que a recarga de Pokébolas só é possível quando um jogador tiver ZERO Pokébolas para a captura de Pokémons. Cada Pokémon possui um *Combat Power*(CP) associado. O CP de um Pokémon mostra o quão forte ele é. O score de um jogador é definido pela soma de todos os CPs de todos os Pokemons que ele capturou. Assim, quanto mais Pokémons de CP altos forem capturados, mais eles contribuem para o score de um jogador.

A Tabela 1 mostra os itens encontrados no mapa (Pokémons e Pokéstops) com seus respectivos códigos. Assim, para valores de 1 a 5 temos os Pokémons e seus respectivos CPs, e o código 6 mostra que existe um Pokéstop no local. Além desses códigos, algumas posições do mapa representam perigo para um determinado jogador, onde este corre o risco de machucar seus Pokémons. No mapa, estes perigos são representados por números negativos que devem ser subtraídos do score de um jogador, caso o mesmo tenha decidido se deslocar para aquela posição. Seu objetivo então é sempre escolher as posições do mapa onde a perda é menor.

Cada jogador possui inicialmente 3 Pokébolas, e 1 Pokébola é gasta na captura de cada Pokémon, independente de seu CP. Na recarga de Pokébolas realizadas nos Pokéstops, cada jogador recebe 1 Pokébola. O número máximo de jogadas, isto é, o número máximo de vezes que um jogador pode se deslocar no mapa é igual a [$3N - 1$, onde N representa a dimensão do mapa quadrado. Assim, para um mapa de tamanho 3, o jogador tem 9 locais para visitar e 8 deslocamentos possíveis]. Ao fim do jogo, o(s) Mestre(s) Pokémon vencedores são aqueles que possui(em) o maior score, dado pela soma dos CPs dos Pokémons capturados menos os scores presentes em regiões perigosas do mapa que o jogador tenha acessado. Em caso de empate, deve-se observar o(s) jogador(es) que possui(em) o maior número de Pokémons de CP alto. Caso ainda tenha-se jogadores empatados, deve-se utilizar o número de passos dados por cada jogador, onde o jogador que tiver concluído o jogo se deslocando um número menor de vezes, é o vencedor. Se não for possível desampatar o jogo com nenhum dos critérios acima, todos os jogadores devem ser considerados vencedores.

Note que um jogador não pode passar por um mesmo local do mapa mais de uma vez, uma vez que tanto os Pokémons capturaddos como os Pokéstops utilizados ficarão indisponíveis após a primeira passagem. Porém, tanto os Pokémons quanto os Pokéstops continuam disponíveis para os próximos jogadores da competição de Mestre Pokémon. [Se a célula inicial onde o jogador inicia o jogo contiver um Pokémon, o mesmo pode ser capturado. Se as pokébolas do jogador acabarem e todas as células eligíveis

Tabela 1: Códigos do mapa de entrada

Item	CP
Célula vazia	0
Dragonite	1
Alakazam	2
Magma	3
Eevee	4
Pikachu	5
Pokéstop	6

para locamoção tiverem um pokémon, o mesmo pode se locomover (desde que a célula ainda não tenha sido visitada) e não capturar o pokémon (isso não aumenta nem diminui seu score). Quando um jogador está rodeado de Pokéstops, mas possui Pokébolas, ele pode se mover para qualquer um dos locais, porém sem pegar Pokébolas (isso também não altera o seu score)] Quando um jogador estiver em um local onde não possa mais se locomover (porque já visitou todas as célular vizinhas) ~~ou por nao ter Pokebolas para capturar um Pokemon em sua vizinhanca~~, o jogo deve terminar para o mesmo. [Os jogadores competem de maneira independente (um por vez) e um jogador não tem qualquer informação sobre os outros.]

2 Tipos Abstratos de Dados

O único TAD obrigatório na sua implementação corresponde ao jogador. No entanto, outros TADs podem ser criados para organizar melhor o seu código, como o TAD Pokémon. Preste atenção as boas práticas de programação ao organizar seu código.

O TAD Jogador deve armazenar uma tabela Pokédex com os Pokémons capturados e seus respectivos CPs, e implementar no mínimo os seguintes métodos:

- explorar(vizinhos) - Retorna uma lista contendo tudo que pode ser encontrado no mapa ao redor do jogador.
- andar(vizinhanca) - Possibilita que um jogador caminhe para as regiões vizinhas. O jogador tem a possibilidade de ir para qualquer uma das 8 regiões ao seu redor (vide Figura 1). Porém, só é permitido que ele dê um passo por jogada.
- caminho_percorrido() - Retorna uma lista com o caminho percorrido pelo jogador.

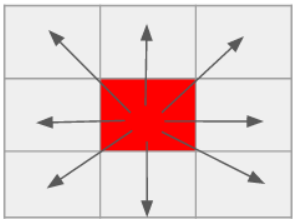

Figura 1: Movimentos Possíveis

Se necessário, acrescente novas funções ao seu TAD detalhando-as na documentação do trabalho.

3 Arquivo de Entrada

Os dados de entrada esatão disponíveis em um arquivo de nome **entrada.txt**, na seguinte ordem:

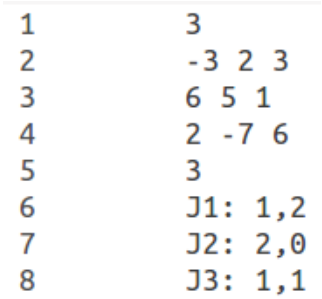

Figura 2: Exemplo de um arquivo de entrada

- Na primeira linha do arquivo temos o tamanho do mapa a ser percorrido. No exemplo temos um mapa de 3 linhas e 3 colunas. **O mapa será sempre representado por uma matriz quadrada.**
- Na segunda linha do arquivo de entrada, inicia-se o mapa em si. Cada linha no arquivo indica uma linha do mapa, e os dados das colunas são separados por espaço. (vide exemplo)
- Na linha que se segue após os dados do mapa (Figura 2), temos o número de jogadores que irão competir entre si. No exemplo, temos um total de 3 jogadores.
- Após o número de jogadores, temos os dados referentes a cada jogador (Figura 2). Cada jogador encontra-se em uma linha do arquivo na seguinte ordem: *NOME: LINHA,COLUNA*. Na linha 6 do exemplo, temos um jogador de nome $J1$ e sua localização inicial no mapa é linha 1 e coluna 2.

Note que os números das linhas são mostrados no arquivo apenas para ilustração, e não estarão no arquivo de entrada.

3.1 Arquivo de Saída

O arquivo de saída deverá ser nomeado **saida.txt**, e deverá conter informações obre um jogador por linha. Em cada linha, deve-se escrever o nome do jogador seguido de seu *score* e o caminho no mapa por ele percorrido. O caminho deve indicar a sequencia de células que o jogador percorreu na matriz, na notação linha,coluna. [Note que a casa inicial deve ser considerada para o estado do jogador.] A última linha deve mostrar o jogador(es) vencedor(es). Na Figura 3 temos como vencedor o jogador $J3$ com um *score* de 6 e um total de 7 passos dados para finalizar o jogo. Note que, embora os jogadores J2 e J3 obtiveram o mesmo *score*, o jogador $J3$ finalizou o jogo com o menor número de passos. No caso do jogo apresentar mais de um vencedor, o nome do segundo vencedor deve aparecer após o do primeiro separado por um espaço, por exemplo, "VENCEDOR J2 J3". [A ordem de impressão dos jogadores no arquivo de saída deve ser a mesma apresentada no arquivo de entrada.]

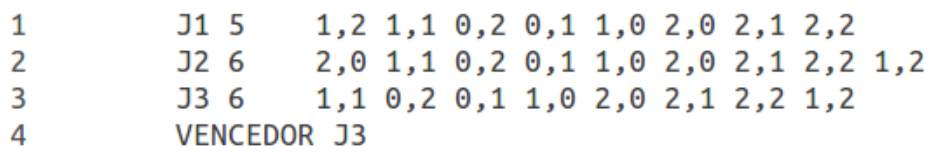

Figura 3: Exemplo de um arquivo de saída

O programa criado não deve conter "menus interativos" ou "paradas para entrada de comandos" (como o system("PAUSE") por exemplo). Ele deve apenas ler os arquivos de entrada, processá-los e gerar os arquivos de saída. Os TPs serão corrigidos em um ambiente Linux. Portanto, o uso de bibliotecas do Windows está PROIBIDO.

[Quanto ao uso de bibliotecas, está liberado o uso de stdio.h e stdlib.h. Qualquer outra biblioteca que você precise de usar, pergunte previamente no fórum nomeado "Bibliotecas Permitidas no TP0".]

4 O que deve ser entregue:

- Código fonte do programa em C (todos os arquivos .c e .h), bem identado e comentado.

- [A pasta de entrega do Trabalho deve ser nomeada "MATRICULA_tp0", onde MATRICULA é seu número de matrícula na universidade. Dentro da pasta de entrega, o arquivo principal (o que contiver o main) deverá receber o nome de "main.c".]
- Documentação do trabalho. Entre outras coisas, a documentação deve conter:

  1. Introdução: descrição sucinta do problema a ser resolvido e visão geral sobre o funcionamento do programa.
  2. Implementação: descrição sobre a implementação do programa. Deve ser detalhada a estrutura de dados utilizada (de preferência com diagramas ilustrativos), o funcionamento das principais funções e procedimentos utilizados, o formato de entrada e saída de dados, compilador utilizado, bem como decisões tomadas relativas aos casos e detalhes de especificação que porventura estejam omissos no enunciado.
  3. Estudo de Complexidade: estudo da complexidade do tempo de execução dos procedimentos implementados e do programa como um todo (notação O), considerando conjuntos de tamanho N. [Devido ao número de jogadores ser sempre menor que o tamanho do mapa, a ordem de complexidade deve ser analisada levando-se em consideração o tamanho do mapa (N).]
  4. Você deve indicar claramente no início da documentação como o seu programa deve ser compilado para gerar o executável. Por exemplo, "gcc main.c jogador.c -o main
  5. Conclusão: comentários gerais sobre o trabalho e as principais dificuldades encontradas em sua implementação.
  6. Bibliografia: bibliografia utilizada para o desenvolvimento do trabalho, incluindo sites da Internet se for o caso.
  7. [Não há um limite máximo de páginas, mas 10 páginas de documentação é suficiente.]
  8. [O pdf entregue deve ser nomeado "MATRICULA.pdf"e será entregue dentro da pasta "MATRICULA_tp0".]

Obs: Apesar desse trabalho ser simples, a documentação pedida segue o formato da documentação que deverá ser entregue nos próximos trabalhos. Um exemplo de documentação está disponível no Moodle.

5 Desafio

Modifique seu programa para que nos métodos EXPLORAR e ANDAR do jogador possam considerar uma vizinhança maior, que considera 3 células em cada uma das direções (*lookahead* de 3)~~, podendo observar e se deslocar para qualquer local dentro deste raio.~~ [No método EXPLORAR cada jogador pode observar 3 células em cada uma das direções. O método ANDAR, NÃO pode saltar. Devendo o jogador percorrer todas as células do caminho para chegar ao destino escolhido.] Note que agora seu processo de decisão é muito mais complexo que no caso anterior. [Os códigos implementados no desafio devem ser entregues em uma pasta chamada "desafio"e esta deve estar inserida na pasta de nome "MATRICULA_tpo".]

6 Comentários Gerais:

1. Comece a fazer este trabalho logo, enquanto o problema está fresco na memória e o prazo para terminá-lo está tão longe quanto jamais poderá estar.
2. Clareza, indentação e comentários no programa também serão avaliados.
3. O trabalho é individual.
4. A submissão será feita pelo Moodle.

5. Trabalhos copiados, comprados, doados, etc. serão severamente penalizados com nota igual a zero.
6. Na segunda prova, uma das questões será relacionada à implementação do trabalho. A nota do trabalho será ponderada pelo seu rendimento nessa questão.
7. Penalização por atraso: (2d - 1) pontos, onde d é o número de dias de atraso. O prazo máximo de atraso é de 3 dias (50% da nota).