

Relatório

Exercício-Programa 4

2º Semestre 2003

1 Introdução

Com base em uma história quase fictícia, o exercício-programa 4 propõe a determinação da realização de um caminho entre dois pontos.

Para a execução do programa é disponibilizado um mapa com trechos seguros de ruas, pontos de referência em que há atiradores, um ponto de entrada e um ponto de saída seguros.

O objetivo é caminhar da entrada à saída alvejando os atiradores (supondo que nunca se erre nem seja atingido), gastando no máximo um número fixo de balas.

Para resolver o problema, foi utilizado um algoritmo fortemente baseado no *Bellman-Ford-Moore* (BFM), com a diferença de que não necessariamente o melhor caminho é o possível, e que os pontos de referência (vértices) predecessores não são armazenados.

Arquivos componentes do EP:

- Celula.java
- FilaPrioridade.java
- InformacaoPrincipal.java
- InformacaoPrioridade.java
- LeituraArquivo.java
- ListaLigada.java
- PontoReferencia.java
- ProcuraCaminho.java
- TabelaHash.java

2 Algoritmo

O algoritmo de *Bellman-Ford-Moore* utiliza uma busca em largura e o método de “procura e rotulação”.

Se considerarmos o mapa a ser percorrido como um grafo, a busca em largura aliada à estratégia de “procura e rotulação” consiste em:

1. Tomar um vértice de partida;
2. Marca-se este vértice como visitado e define-se seu custo total como nulo;
3. Inicializa-se uma fila de prioridades com este vértice;

4. Retira-se o primeiro elemento da fila de prioridades, determinando-se todos os seus vizinhos (filhos) e marcando-o como visitado;
5. Enfileira-se todos os vizinhos para então descobrir qual deles conduz a um caminho possível e/ou otimizado;
6. Repetem-se os passos 4 e 5 até que a fila de prioridades esteja vazia.

O processo só devolve uma resposta afirmativa se, durante, ou ao final do mesmo, for detectado que algum dos caminhos atingiu a saída.

Uma vez atingido o ponto de saída, verifica-se se ao alcançar o mesmo, foi gasto um número de balas menor ou igual ao disponível. A possibilidade de haver um caminho entre os pontos fornecidos depende desta verificação (mesmo que haja um caminho possível, pode ser que hajam menos balas disponíveis que o necessário).

A funcionalidade do algoritmo é devida ao fato de que diversos caminhos são testados, na busca pela saída. Caso não haja nenhum caminho que se ligue à saída, todos os caminhos terão sido analisados e o algoritmo responderá que não encontrou a saída. Na ocasião do primeiro encontro da saída, significa que algum caminho é capaz de atingi-la, sendo esta condição encaminhada para o teste final, o do número de balas disponíveis.

3 Dificuldades encontradas

Logo de início a maior dificuldade é por optar por um modelo completamente baseado em vetores (*arrays*) ou por algo mais orientado a objetos.

A fim de dar continuidade com a experiência na linguagem Java, optou-se pelo segundo modelo.

Sem dúvida, as dificuldades iniciais e que mais atrasaram o desenvolvimento do exercício foram as rotinas de criação e ligação de pontos de referências do mapa da cidade.

Resultante disso, tem-se um código não muito otimizado, principalmente nas classes `LeituraArquivo` e `ProcuraCaminho`.

Uma vez suposto que as tais rotinas estivessem funcionando, a implementação do algoritmo revelou o contrário, o que levou a mais horas de depuração.

Foi preciso notar que estavam sendo feitas inserções repetidas na fila de prioridades, que caminhos estavam apontando pra si mesmo, outros caminhos não faziam as devidas ligações, estouros de fila, dentre muitos outros.

4 Prós e contras

À primeira vista, a implementação parece resolver o problema sugerido.

A orientação a objetos foi utilizada extensivamente, de modo similar aos exercícios anteriores de forma que, foi até mesmo possível aproveitar **integralmente** classes dos EPs anteriores (`ListaLigada`, `TabelaHash`, etc).

Um problema é que talvez o modo como foi feita essa implementação tornou o código um pouco poluído (excesso de *casts*, muitos encadeamentos em chamadas de métodos, recuos forçados em linhas de comando).

A documentação apresentada junto ao código visa minimizar ao máximo eventuais confusões que isso poderia acarretar.

Não foram feitos testes exaustivos que garantam a corretude do algoritmo, portanto, o mesmo pode falhar para casos pontuais ou mesmo para caso geral.

5 Considerações finais

O exercício foi uma ótima oportunidade para um contato inicial e superficial com o que, convencionalmente, deve se chamar de “Teoria dos Grafos”.

Deve servir de introdução para aprendizados futuros, dadas as dificuldades enfrentadas e eventuais dúvidas que permaneçam.