Trabalho Prático 2: Andando na Física

Algoritmos e Estruturas de Dados III – 2016/2

Bruno Varella Peixoto

1 Introdução

Com o objetivo de praticar os novos conhecimentos sobre grafos, este trabalho propõe um problema clássico de encontrar o caminho mais curto em um labirinto. Mais especificamente, dada a posição inicial de um garoto (Vinícius) no departamento de Física do ICEx, deve-se encontrar a distância do caminho mais curto até a saída. Há algumas restrições como portas trancadas e buracos de minhoca que teletransportam para uma posição específica do mapa.
A entrada é um vetor de posições que podem ser: posições livres, chaves, portas, buracos de minhoca ou paredes. A posição inicial também é especificada, assim como a de saída do departamento.
A saída é a distância mínima do início até a saída. Caso seja impossível sair do departamento, imprime-se -1.

2 Solução do Problema

Para resolver o problema de forma simples, aproveitando os conceitos vistos em sala, o mapa do departamento de Física foi modelado utilizando grafos. Foi utilizado um grafo direcionado, não ponderado, implementado com listas de adjacências. Cada vértice representa uma posição no mapa e guarda uma lista para os adjacentes. Além dessa lista, as seguintes informações são armazenadas:

- level: distância da posição inicial até o vértice atual.
- color: marcação utilizada no BFS.
- status: tipo de posição (livre, chave, porta, etc…).
- code: nome da chave ou porta (caso posição seja chave ou porta).
- wormhole_position: vértice ao qual deve-se teletransportar (caso posição seja buraco de minhoca).

Para encontrar a menor distância, foi implementado um BFS seguindo a indicação do livro *Introduction to Algorithms* do Thomas Cormen. O que muda basicamente é que a cada vértice, seu status é checado. Caso seja chave ou buraco de minhoca, um novo BFS é chamado, partindo deste novo vértice inicial. A ideia é calcular level.exit_vertex. Uma diferença da implementação do Cormen, é que não é calculado o parent de cada vértice, ou seja, não é armazenado qual o caminho mais curto, apenas sua distância.

3 Análise de Complexidade de Tempo e Espaço

A solução dada ao problema é de complexidade linear de espaço referente ao número de vértices (posições) no mapa.
A complexidade total de tempo se dá pela implementação recursiva do BFS, que é normalmente O(V+E) sendo V o número de vétices e E o número de arestas do grafo. Porém, como o BFS é chamado recursivamente para cada chave e buraco de minhoca, a

complexidade fica em ordem exponencial O(V+E)^(T+W) sendo T igual ao número de chaves e W o número de buracos de minhoca.

4 Análise de Experimental

Para testar a análise teórica, foi fixado um mapa de dimensões 8x8 e o número de buracos de minhoca e chaves foi variado. Os testes foram executados em um Core i7 6700k 4.5Ghz, 64GB de RAM DDR4 3000Mhz, sistema operacional Fedora 24.

BURACOS DE MINHOCA	CHAVES	TEMPO (s)
0	0	0,01s
1	1	0,01s
2	2	0,01s
3	3	0,01s
4	4	0,01s
4	5	0,01s
4	6	0,01s
4	7	0,01s
4	8	0,02s

Infelizmente não foi possível perceber o comportamento exponencial do algoritmo pelos testes. Isso provavelmente se deu pelo tamanho reduzido do grafo (apenas 64 vértices).

5 Conclusão

Com este trabalho foi possível consolidar os conhecimentos sobre grafos e a implementação de algoritmos como o BFS. É interessante notar que não foi necessário utilizar um algoritmo como Dijkstra pois o grafo não é ponderado, portanto o BFS consegue computar o menor caminho de cada vértice até o inicial. Tudo ocorreu como esperado. O mais difícil foi entender como a recursão iria ocorrer e evitar que um caminho mais longo substituísse um mais curto.