UNIVERSIDADE FEDERAL DO VALE DO SÃO FRANCISCO
CAMPUS - JUAZEIRO

ALLISSON PIERRE LINO GOMES
CAROLINE CARVALHO MACHADO
ESRON DTAMAR DA SILVA
PEDRO HENRIQUE DUARTE SANTANA

IMPLEMENTAÇÃO DO ALGORITMO DO MENOR CAMINHO DE DIJKSTRA: O CASO DO METRÔ DE PARIS

JUAZEIRO - BAHIA
2013

ALLISSON PIERRE LINO GOMES
CAROLINE CARVALHO MACHADO
ESRON DTAMAR DA SILVA
PEDRO HENRIQUE DUARTE SANTANA

IMPLEMENTAÇÃO DO ALGORITMO DO MENOR CAMINHO DE DIJKSTRA: O CASO DO METRÔ DE PARIS

Projeto apresentado como requisito para avaliação da disciplina Estrutura de Dados II, do curso de Engenharia de Computação, solicitado pela professora Ana Emilia de Melo Queiroz.

JUAZEIRO - BAHIA
2013

1. RESUMO

Neste relatório é apresentada uma aplicação do algoritmo de Dijkstra na solução de um problema clássico de cálculo do trajeto mais curto entre estações do metrô de Paris e as consequências de um levantamento restritivo ou errôneo das variáveis envolvidas no problema. Para visualizar os resultados dos cálculos foi desenvolvido um software de em linguagem de programação C que executa o cálculo da rota e exibe o melhor trajeto em um mapa.

Algoritmos de busca são ferramentas utilizadas na resolução de problemas complexos e que necessitam, normalmente, de uma abordagem diferente da oferecida pela programação convencional. São muito comuns nas soluções de problemas pesquisados pela Inteligência Artificial. Segundo Russell e Norvig (2004), os algoritmos de busca foram estabelecidos como as principais armas do arsenal dos pesquisadores de IA na década de 1960.

Muitas aplicações do mundo real requerem o auxílio destes algoritmos. Neves (2007) cita vários exemplos de aplicações que utilizam estes algoritmos como, por exemplo, o roteamento de veículos em um sistema de transporte, que se integram a sistemas de GPS, computadores portáteis e celulares. Outra aplicação deste tipo de sistema e que utiliza o algoritmo de Dijkstra pode ser obtida em Derekenaris et al. (2000) onde os autores propõem um sistema de gerenciamento de ambulâncias e emergências baseado em SIG, GPS e GSM. Outros exemplos clássicos são a transmissão de pacotes em redes de computadores e projetos de circuitos integrados em que os componentes precisam ser ligados por um menor caminho possível.

Em contraste com a importância e grande utilização destes algoritmos, a compreensão do seu funcionamento, em geral, não é imediata e trivial durante o processo ensino-aprendizagem. Isto ocorre devido as suas particularidades e principalmente a mudança no método de programação utilizado.

Este trabalho visa ilustrar a aplicação do algoritmo de Dijkstra na solução de um problema clássico de cálculo do trajeto mais curto entre estações de um metrô, bem como as consequências de uma definição errônea das variáveis envolvidas no problema. Como metodologia, foi desenvolvido um software que executa o cálculo das rotas e exibe o trajeto em um mapa.

2. PROBLEMA DA BUSCA EM GRAFOS

Vários problemas representados por um grafo podem ser resolvidos efetuando uma busca nesse grafo. Às vezes é preciso visitar todos os vértices de um grafo, às vezes o problema pode ser resolvido visitando somente um subconjunto dos vértices, por exemplo, o problema do caminho mais curto.

Os algoritmos apresentados para resolver esse problema fazem um percurso exaustivo de todos os vértices, o que não é necessário se, por exemplo, procura-se apenas o caminho mais curto até um vértice em particular. Nesse caso, assim que ele se encontra no conjunto dos vértices já visitado, não é preciso continuar o algoritmo.

Há basicamente duas maneiras de se realizar a busca em grafos: a busca em largura e a busca em profundidade:

a. Busca em largura - O algoritmo de Busca em Largura é uma proposta que trabalha sob o critério FIFO. É também denominada de busca em amplitude.Funciona da seguinte forma: aplicam-se todos os operadores possíveis à raiz (isto é, ao estado inicial), em seguida, a todos os filhos da raiz (da esquerda para a direita), depois, a todos os "netos" da raiz (da esquerda para a direita) e assim por diante. A busca termina ao se descobrir o nó objetivo.Existem duas filas de trabalho usadas pela busca em largura.Neste algoritmo, todos os vértices de certo nível da árvore são examinados antes do nível abaixo. Se existe uma solução e se o grafo é finito, então por este método ela certamente será encontrada.

b. Busca em profundidade - O algoritmo de Busca em Profundidade é uma proposta que trabalha sob o critério LIFO. É também denominada de primeiro-em-profundidade. Quanto à estrutura do algoritmo, a única diferença em relação à busca em largura, é a utilização de uma pilha ao invés de fila. Esta técnica explora o caminho para o objetivo, dando preferência aos nós que estão mais distantes da raiz da árvore de busca. Funciona bem, quando as soluções são total e igualmente desejadas ou quando anteriormente foi feita uma varredura, detectando direções incorretas.

3. PROBLEMA DO CAMINHO MAIS CURTO

Um grafo ***G (V, E)*** é um conjunto finito não vazio ***V*** e um conjunto ***E*** de pares não ordenados de elementos distintos de ***V***. Os elementos de ***V*** são os vértices e os de ***E*** são as arestas de ***G*** respectivamente. Cada aresta ***e € E*** será denotada pelo par de vértices ***e = (v, w)*** que a forma. Nesse caso, os vértices ***v*** e ***w*** são os extremos da aresta ***e***, sendo denominados adjacentes. A aresta ***e*** é dita incidente a ambos ***v*** e ***w***. Duas arestas que possuem um extremo comum são chamadas de adjacentes.

Seja um grafo ***G (V, E)*** e uma função distância ***L*** que associe cada aresta ***(v, w)*** a um número real não negativo ***L (v, w)*** e também um vértice fixo ***v0*** em ***V***, chamado fonte. O problema consiste em se determinar o caminho de ***v0*** para um vértice ***vf*** de ***G***, de tal forma que a somatória das distâncias das arestas envolvidas no caminho seja mínima. Dessa forma, seja p o peso do caminho c de ***v0*** até um ***vf*** (v0, v1, ..., vf), então p será a soma de todos os pesos das arestas do caminho:

$$P(c) = \sum p(vi\text{-}1, vi)$$

No trabalho proposto, podemos representar o metrô de Paris através de um grafo ***G (V, E)***, onde as estações são os vértices e as linhas do metro são as arestas, contendo as baldeações nas trocas das linhas. O problema do menor caminho consiste em determinar por quais vértices se deverá passar para seguir de um vértice ***v0*** até um vértice ***vf***, percorrendo-se a menor distância possível.

O PMC (problema do menor caminho) é um importante problema encontrado na Teoria dos Grafos, pela sua aplicação prática. Assim como os grafos podem ser usados para representar matematicamente uma enorme quantidade de problemas, o PMC pode ser usado para resolver estes problemas.

Sua aplicação pode ser vista em alguns exemplos citado por diversos autores, como o problema do caixeiro viajante: suponha que se tenha um mapa de rede rodoviária em que as cidades são representadas por pontos e as estradas por linhas. O mapa, sem dúvida, pode ser considerado um grafo, em que as distâncias são ponderações das arestas, podendo ser consideradas também como custo ou tempo. O Problema do Caixeiro Viajante consiste em encontrar o caminho de menor custo que o caixeiro viajante deve percorrer, partindo de uma cidade de origem escolhida e visitando todas as outras uma única vez e retornando à cidade de origem.

O Problema do Caixeiro Viajante é um problema mais teórico que prático, mas existem problemas reais onde se necessita passar por vários vértices e retornar ao vértice de origem com o menor custo possível. Podem ser citados aqui os problemas de entrega de mercadorias, onde se deseja saber qual o caminho que deve ser percorrido a um custo mínimo.

O custo para percorrer certo trecho de um caminho pode ter uma série de variáveis, como o comprimento; as condições da estrada, o nível de tráfego; o número de oficinas mecânicas no trecho, ou qualquer outra variável que possa ser relevante, no caso do metrô de Paris existem como variáveis a distância real, a distância direta, as linhas que existem entre uma estação e outra.

Para resolver o problema do caminho mais curto foi então usado o algoritmo de Dijkstra, que mantem um conjunto S de vértices cujos caminhos mais curtos até um vértice origem já são conhecidos. Ao final da sua execução o algoritmo produz uma arvore de caminhos mais curtos de um vértice origem para todos os vértices que são alcançáveis a partir do vértice origem.

4. ALGORITMO DIJKSTRA

O algoritmo Dijkstra, assim chamado devido ao seu criador EdsgerDijkstra, surgiu para solucionar o problema do caminho mínimo entre vértices de um grafo dirigido ou não dirigido. Esse algoritmo não consegue encontrar o menor caminho em um grafo com arestas de pesos negativos, para esse propósito, pode-se usar o algoritmo de Bellman-Ford, que possui uma maior complexidade de tempo de execução do que o Dijkstra.

O Dijksta é considerado um algoritmo guloso, pois toma a decisão que parece ótima no momento, apesar disso ele é considerado ótimo pois encontra sempre o caminho de menor custo da raiz para todos os demais nós do grafo. A estratégia gulosa é conveniente já que em um menor caminho C entre 2 vértices do grafo, todo sub-caminho entre esses vértices é um menor caminho entre 2 vértices pertencentes ao caminho C, desta forma são construídos os melhores caminhos dos vértices alcançáveis pelo vértice inicial determinando todos os melhores caminhos intermediários, caso existam 2 menores caminhos apenas um será descoberto.

Um exemplo prático de um problema que pode ser resolvido pelo algoritmo de Dijkstra é quando alguém quer se deslocar de uma cidade para outra, em que existem várias estradas com quilômetros diferentes, assim o Dijkstra procuraria o caminho mais curto, um outro exemplo seriam as estações de metrô, em que ele avaliaria qual é melhor pegar para se deslocar de um ponto a outro na cidade.

Os pesos dos caminhos (arestas), podem ser compostos por informações que representam a distância entre determinadas localidades, os custos operacionais, a quantidade de recursos, o tempo gasto, ou qualquer outro elemento passível de ser ponderado entre os determinados vértices.

A relação do Dijkstra com a busca em largura, é que a busca em largura é a base para esse algoritmo do caminho mais curto. O algoritmo da Busca em Largura calcula a distância (menor número de arestas) desde o vértice raiz até todos os vértices acessíveis, é um método de busca não-informada que expande e examina sistematicamente todos os vértices de um grafo direcionado ou não-direcionado.

Quando se tem arestas não valorizadas em um grafo, é mais fácil encontrar a distância para o vértice raiz com o algoritmo de busca em largura, já quando se tem pesos nas arestas, ou seja, quando a aresta passa a ser valorizada, utiliza-se o algoritmo Dijkstra para encontrar o menor caminho de um vértice a outro.

5. ESTRUTURA DE DADOS UTILIZADA

No projeto, o grafo foi armazenado em uma lista de adjacências simplesmente encadeada e ordenada. O vetor base da lista será carregado com o número de todas as estações (campo vert). Cada item do vetor é uma lista de adjacências com variáveis que apontam para o início e o final da lista. As variáveis tempo e distancia serão utilizadas ao se armazenar o caminho percorrido, que se utiliza da mesma estrutura.

A lista de adjacências é formada por células encadeadas, cada célula contém variáveis que armazenam o número da estação, a cor da linha pela qual ela é ligada a estação do vetor base, e a distancia entre a estação do vetor base e a estação da célula. Uma variável apontará para outra célula que também se liga a estação do vetor base.

Uma matriz foi utilizada para armazenar as distâncias diretas entre as estações, ambas as dimensões são a quantidade de estações.

```
//Declaração da lista de adjacências

typedefstructcelula_str
{
        intestacao;
        char linha;
        floatd_real;
        LINK prox;

}celula;

typedefstruct
{
        LINK head;
        LINK tail;
        intvert;
        float tempo;
        float distancia;

}TipoLista;

TipoLista listadj[LIM];
floatm_dis[LIM][LIM];
```

6. MATERIAL E MÉTODO

A linguagem utilizada foi a linguagem C. As bibliotecas utilizadas foram a <stdlib.h> devido ao uso de funções do sistema, a <stdio.h> para manipulação de entrada e saída, e por fim foi utilizada a <windows.h> para a construção da interface gráfica. Os dados que formaram o grafo foram lidos de quatro arquivos:

```
estacao.txt       // Arquivo contendo as estações
linha.txt         // Arquivo contendo as linhas
d_real.txt        // Arquivo contendo as distâncias reais entre as estações
d_direta.txt      // Arquivo contendo as distâncias diretas entre as estações
```

7. RESULTADOS

Ao término do projeto, foi adquirira experiência para resolver o problema do caminho mais curto com vários algoritmos de busca, Dijkstra, busca em largura e busca em profundidade. Também foi adquirida experiência sobre criação da GUI – Graphical User Interfaceem ambiente WIN32.

A tabela abaixo faz uma comparação entre os algoritmos de busca em largura e busca em profundidade. O teste foi feito em um grafo dirigido simples e conexo.

Tabela 1: Comparação entre algoritmos de busca

Vértices	Quando o vértice de destino foi atingido	
	BFS	DFS
10	7	5
30	13	11
60	25	34
100	37	76
200	129	128
500	239	318

A Tabela 1 mostra a comparação feita entre os algoritmos de busca em largura e busca em profundidade. Como comparação foi medida quantos vértices foram procurados até que o vértice de destino fosse atingido. Observa-se que ambas as buscas tiveram médias semelhantes, o que demonstra que muito do desempenho dos dois algoritmos deve-se a localização do vértice que está sendo procurado. Caso o vértice esteja num nível mais profundo, então a busca em profundidade leva vantagem, enquanto que se o vértice estiver em níveis iniciais, a busca em largura mostra-se muito eficiente.

REFERÊNCIAS BIBLIOGRÁFICAS

CORMEN, T. H. et al. **Algoritmos:** Teoria e Prática. Rio de Janeiro: Editora Campus Elsevier, 2002.

SEDGEWICK, R. **Algorithms in C.** 3. ed. Boston: Addison-Wesley, 1998.

STEWART, J. **Cálculo.** 4. ed. São Paulo: Pioneira Thomson Learning, v. 1, 2001.

TENENBAUM, A. M.; LANGSAM, Y.; AUGENSTEIN, M. **Estruturas de dados usando C.** São Paulo: Pearson Makron Books, 1995.

ZIVIANI, N. **Projetos de Algoritmos:** com implementações em Pascal e C. 2. ed. São Paulo: Pioneira Thomson Learning, 2004.