ALGORITMO DE DIJKSTRA

por

Bruno Miguel Pacheco Saraiva de Carvalho

Departamento de Engenharia Informática

Universidade de Coimbra

3030 Coimbra, Portugal

brunomig@student.dei.uc.pt

Resumo – *Descreve-se o funcionamento do algoritmo de Dijkstra através do cálculo do caminho mais curto entre duas cidades e do recurso aos grafos para a sua representação. Este algoritmo resolve o problema de qualquer cálculo do caminho mínimo num grafo ponderado apenas com pesos positivos.*

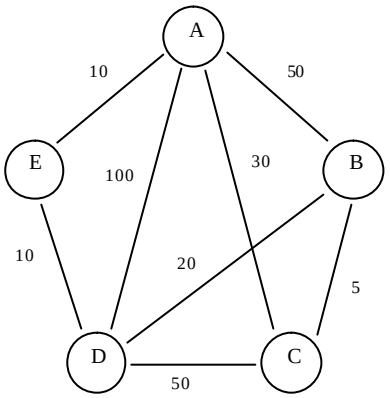
Palavras chave – caminho mais curto, algoritmo de Dijkstra, grafo.

1 – INTRODUÇÃO

Das mais diversas utilizações dos grafos (análise de circuitos, análise e planeamento de projectos, genética, linguística, ciências sociais, robótica...), uma delas é a optimização de percursos. Assim, uma rede de estradas que liga diversas cidades pode ser representada através de um grafo. Através da conjugação do uso de um grafo e do Algoritmo de Dijkstra é possível calcular o caminho mais curto para realizar determinado percurso. Deste modo, pretende-se com este artigo descrever o funcionamento do referido algoritmo, permitindo mesmo a sua implementação a nível informático, fazendo notar algumas vantagens e limitações deste algoritmo quando comparado com algoritmos semelhantes. É feita a descrição passo a passo, através do uso de um exemplo.

2 – EXECUÇÃO DO ALGORITMO DE DIJKSTRA

Um grafo G é um tuplo em que G = (V,A), onde V é um conjunto não vazio de vértices e A é um conjunto de pares de vértices denominados de arcos e que representam uma ligação entre dois vértices, arcos esses que podem ter um determinado peso (grafo ponderado), que será o custo necessário para percorrer o arco, pelo que uma rede de estradas que liga diversas cidades pode ser representada através de um grafo ponderado com pesos positivos. O grafo seguinte representa o conjunto de cidades V={A,B,C,D,E} com as respectivas estradas e distâncias que as ligam:



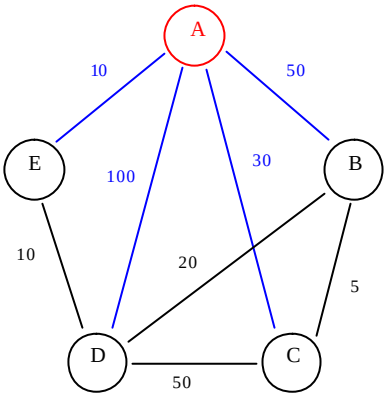
Pretende-se calcular, de entre os caminhos possíveis, aquele que é o caminho mais curto entre a cidade *A* e as restantes cidades. Para resolver este problema usou-se o Algoritmo de Dijkstra. O Algoritmo de Dijkstra (E.W. Dijkstra) é um dos algoritmos que calcula o caminho de custo mínimo entre vértices de um grafo. Escolhido o vértice *A* como raiz da busca (cidade origem), este algoritmo calcula a distância mínima deste vértice para todos os demais vértices do grafo, ou seja, as restantes cidades.

Este algoritmo parte de uma estimativa inicial para a distância mínima, que é considerada infinita (∞), e vai sucessivamente ajustando esta distância. Ele considera que uma cidade estará "fechada" quando já tiver sido obtido um caminho de distância mínima da cidade tomada como origem da busca até ela. Este procedimento pode ser armazenado na seguinte tabela que irá sendo alterada à medida que vão sendo percorridos todos os caminhos possíveis:

Cidades	A	B	C	D	E
Distância	0	∞	∞	∞	∞
Precedente	-	-	-	-	-
Fechado	N	N	N	N	N

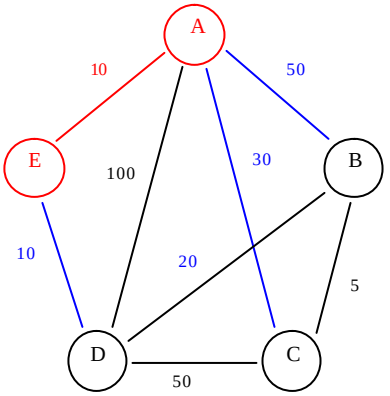
Inicialmente, todos os vértices têm uma distância infinita, excepto a cidade de origem (*A*) que tem, logicamente, distância zero. Posteriormente, é seleccionada a cidade que tem uma distância menor em relação a todas as outras e que se encontra "aberta" que é, no caso inicial, a cidade *A*. A cidade seleccionada é "fechada" e são recalculadas, a partir de *A*, todas as distâncias às cidades ainda "abertas", somando à distância da cidade seleccionada as distâncias dos arcos respectivos. Nos casos em que as novas distâncias obtidas são inferiores às que se encontram armazenadas nas tabelas, procede-se à substituição das mesmas pelas novas distâncias e é alterada também a cidade precedente. Após o primeiro passo, a tabela seria a seguinte:

Cidades	A	B	C	D	E
Distância	0	50	30	100	10
Precedente	A	A	A	A	A
Fechado	S	N	N	N	N



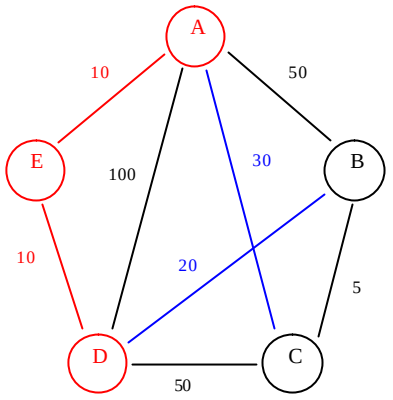
De forma análoga, o segundo passo será seleccionar a cidade ainda "aberta" com a menor distância na tabela (a cidade *E* com distância 10), "fechá-la" e, a partir de *E*, recalcular as distâncias, alterando aquelas que sejam menores que as da tabela (a distância de *D* é alterada para 20 com precedente *E*), que ficaria da seguinte forma:

Cidades	A	B	C	D	E
Distância	0	50	30	20	10
Precedente	A	A	A	E	A
Fechado	S	N	N	N	S



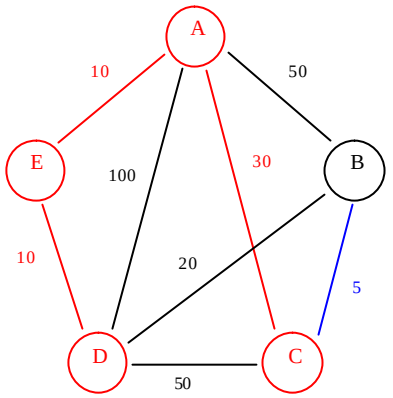
Repete-se o algoritmo até que todas as cidades tenham sido “fechadas”. As tabelas após as sucessivas operações são as seguintes:

Cidades	**A**	**B**	**C**	**D**	**E**
Distância	0	40	30	20	10
Precedente	A	D	A	E	A
Fechado	S	N	N	S	S



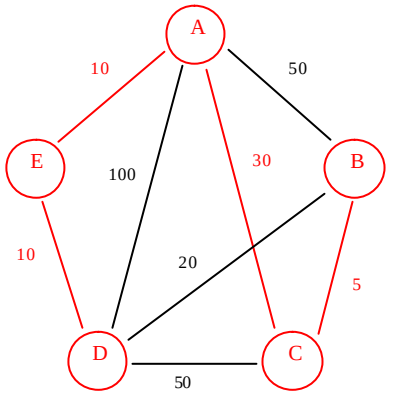
- *D* é “fechada”;
- Distância e Precedente de *B* são alterados.

Cidades	**A**	**B**	**C**	**D**	**E**
Distância	0	35	30	20	10
Precedente	A	C	A	E	A
Fechado	S	N	S	S	S



- *C* é “fechada”;
- Distância e Precedente de *B* são alterados.

Cidades	**A**	**B**	**C**	**D**	**E**
Distância	0	35	30	20	10
Precedente	A	C	A	E	A
Fechado	S	S	S	S	S



- *B* é “fechada”.

Quando todas as cidades tiverem sido “fechadas”, os valores obtidos são a distância mínima dos caminhos que partem da cidade *A* para as restantes. O caminho percorrido nesse trajecto é obtido a partir dos valores colocados no campo “Precedente”. Por exemplo, considere-se o caminho que vai de *A* até *B*, cuja distância mínima é 35. A cidade precedente

de *B* é *C*. Sendo assim, o caminho é *A* ? ?? ? *C* ? *B*. Por sua vez, o precedente de *C* é *A*. Portanto, o caminho final é *A* ? *C* ? *B*, com uma distância mínima de 35.

É muito importante referir que o Algoritmo de Dijkstra só pode ser utilizado em grafos ponderados e unicamente com pesos positivos e permite calcular as distâncias entre uma cidade específica e todas as outras, ao contrário do Algoritmo de Floyd que calcula as distâncias entre todas as cidades.

De forma computacional, o Algoritmo de Dijkstra pode ser traduzido da seguinte forma:

- Seja G(V,A) um grafo orientado e *a* um vértice de G:

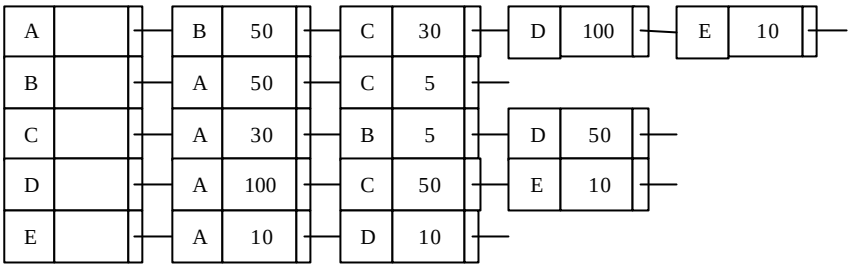
1. Atribui-se valor zero à estimativa do custo mínimo do vértice *a* (a raiz da busca) e infinito às demais estimativas;
2. Atribui-se um valor qualquer aos precedentes (o precedente de um vértice *t* é o vértice que precede *t* no caminho de custo mínimo de *a* para *t*);
3. Enquanto houver vértice aberto:
    - seja *k* um vértice ainda aberto cuja estimativa seja a menor dentre todos os vértices abertos;
    - fecha-se o vértice *k*
    - Para todo vértice *j* ainda aberto que seja sucessor de *k* faz-se:
        - soma-se a estimativa do vértice *k* com o custo do arco que une *k* a *j*;
        - caso esta soma seja melhor que a estimativa anterior para o vértice *j*, substitui-se e anota-se *k* como precedente de *j*.

A implementação computacional deste algoritmo tem uma complexidade $O(n^2)$, enquanto, por exemplo, o Algoritmo de Floyd tem uma complexidade superior de $O(n^3)$, o que é uma grande desvantagem quando o número de vértices e arcos (ou sejam, cidades e estradas) vai aumentando, portanto, o computador levará menos tempo a realizar os cálculos necessários se for usado o Algoritmo de Dijkstra.

Como os grafos não existem como estrutura de dados pré-definida nas linguagens de programação, a sua implementação pode ser feita através de uma matriz de adjacência ou de uma lista de adjacência:

	A	B	C	D	E
A		50	30	100	10
B	50		5		
C	30	5		50	
D	100		50		10
E	10			10	

Matriz de adjacência



Lista de adjacência

3 – CONCLUSÕES

Pretendeu-se que este artigo explicasse, de forma sucinta, objectiva e simples, o funcionamento do Algoritmo de Dijkstra com a utilização dos grafos ponderados de pesos positivos. Optou-se por uma descrição sequencial dos passos a seguir na execução do referido algoritmo, tendo antes sido definidos alguns conceitos necessários a uma correcta compreensão do mesmo. Apesar de se limitar a grafos com pesos positivos e calcular apenas a distância entre uma cidade e todas as outras, este algoritmo é bastante mais rápido na sua execução que outros algoritmos existentes, pelo que o seu uso é adequado em grande parte das situações de cálculo do caminho mais curto.

REFERÊNCIAS

1. Acetatos da cadeira de Comunicação e Profissão, 2002/2003.
2. Acetatos da cadeira de Programação e Algoritmos 3, 2001/2002.
3. Mark Allen Weiss *Data Structures & Problem Solving Using JAVA*, Addison-Wesley Pub. Co., 1998.
4. David M. Arnow and Gerald Weiss *JAVA - An Object Oriented Approach*, Addison-Wesley Pub. Co., 1998.