

UNIVERSIDADE FEDERAL DE SERGIPE

Iuri
Greguinho
Raul Rodrigo
Rafael Castro Nunes
Ruan
Vitor Bonfim

Grafos & Garfos

São Cristóvão, SE
22 de março de 2017

Iuri
Greguinho
Raul Rodrigo
Rafael Castro Nunes
Ruan
Vitor Bonfim

Grafos & Garfos

Relatório em conformidade com as normas
ABNT (pra falar a vdd n sei oq escrever aqui)

Universidade Federal De Sergipe
Faculdade de Engenharia Eletrônica
Redes e Comunicações

São Cristóvão, SE
22 de março de 2017

Agradecimentos

O agradecimento principal é direcionado a Ruan, por ter feito o trabalho todo.

Agradecimento especial ao querido professor felix, por sabotar a prova de redes.

a vida é bonita é bonita

Resumo

Colocar aqui um resumo bem legal aqui.

Palavras-chaves: Grafos. Dijkstra.

Lista de ilustrações

Figura 1 – Grafo Orientado	14
Figura 2 – Grafo Não Orientado	14
Figura 3 – Grafo Sem Peso	14
Figura 4 – Grafo Com Peso	15

Lista de tabelas

Tabela 1 – Matriz Adjacência sem Peso	15
Tabela 2 – Matriz Adjacência com Peso	15

Lista de abreviaturas e siglas

Rn Ruan

Ru Raul

Lista de símbolos

Γ	Letra grega Gama
Λ	Lambda
ζ	Letra grega minúscula zeta
\in	Pertence

Sumário

	Introdução	9
1	REVISÃO BIBLIOGRÁFICA	10
1.1	Dijkstra	10
1.2	Bellman-Ford	10
2	OBJETIVOS	12
3	FUNDAMENTAÇÃO TEÓRICA	13
3.1	Grafos	13
3.1.1	Vértices e Arestas	13
3.1.2	Antecessor, Sucessor e Vizinho	13
3.1.3	Grafos Orientados ou Não Orientados	13
3.2	Representação de um grafo	14
3.2.1	Matriz de Adjacência	14
3.3	Aplicação para o Problema do Menor Caminho	15
3.3.1	Algoritmo de Dijkstra	16
3.3.2	Algoritmo de Floyd- Warshall	16
4	FORMULAÇÃO DO PROBLEMA	17
5	RESULTADOS OBTIDOS	18
6	CONCLUSÃO	19
	REFERÊNCIAS	20

Introdução

1 Revisão Bibliográfica

1.1 Dijkstra

Em (CARVALHO, 2008) é apresentado o algoritmo de dijkstra, sendo este utilizado para obter o menor caminho de um vértice de origem até cada um dos outros vértices do grafo, onde G é um grafo simples, caso este não seja simples, é necessário torná-lo. O artigo ressalta que ao contrário do algoritmo Bellman-Ford, Bellman-Ford impõe restrições sobre o sinal do peso das arestas, criando uma solução menor, dependente dos vértices de início e final.

Ainda em (CARVALHO, 2008) é resolvido um pequeno exemplo com o algoritmo de dijkstra que envolve distância entre cidades, mostrando o menor caminho entre elas. Esse exemplo foi feito passo a passo, explicando minuciosamente como funciona esse algoritmo. Também ressalta que este só pode ser utilizado em grafos ponderados e unicamente com pesos positivos, calculando a distância entre uma cidade e todas as outras, diferentemente do Algoritmo de Floyd que calcula a distância entre todas as cidades.

Em (BARROS; PAMBOUKIAN; ZAMBONI, 2007) há a apresentação do Algoritmo de dijkstra e a explicação do algoritmo passo a passo, feita de forma diferente do (CARVALHO, 2008) pois este é feito de forma mais mecânica, com um exemplo mecânico. Apenas com uma tabela e como o algoritmo funciona e seus passos.

Em (BARROS; PAMBOUKIAN; ZAMBONI, 2007) também é dita algumas aplicações, indo de uma cadeia de produção, até o clássico problema do carteiro que não pode passar duas vezes na mesma rua. Qualquer grafo simples que possua a matriz de pesos definida pode ser submetida à proposta de dijkstra.

1.2 Bellman-Ford

Em (GARCIA, 2011) é apresentado o algoritmo de Bellman-Ford, sendo este utilizado para obter o menor caminho de um nodo de origem até cada um dos outros nodos de G , onde G é um dígrafo (grafo orientado) com arestas ponderadas. O artigo ressalta que ao contrário do algoritmo Dijkstra, Bellman-Ford não impõe restrições sobre o sinal do peso das arestas, criando uma solução mais genérica.

Ainda em (GARCIA, 2011), algumas das principais aplicações são mostradas, Protocolos de Roteamento Vetor-Distância e Problema "Triangular Arbitrage", útil para problemas da área de economia e investimento.

Mesmo com estruturas similares, algumas diferenças são vistas entre Dijkstra e Ford em (BARROS; PAMBOUKIAN; ZAMBONI, 2007) e (GARCIA, 2011), enquanto o primeiro busca exaustivamente o nodo com menor peso ainda não computado, o último usa o procedimento de relaxamento $V - 1$ vezes, onde V representa a quantidade de vértices em G .

2 Objetivos

3 Fundamentação Teórica

3.1 Grafos

O conceito de grafo é um conceito simples, porém muito amplo, tratando da relação de conexão entre elementos, nos mais diversos problemas matemáticos. Podemos então definir grafo como a união de um conjunto de vértices e um conjunto de arestas. Geralmente os grafos tem a seguinte notação:

$$G = (V, A) \quad (3.1)$$

Onde V representa o conjunto de vértices e A representa o conjunto de arestas.

3.1.1 Vértices e Arestas

Os vértices, também chamados de nós, são os elementos de um conjunto e o número de elementos deste conjunto representa a ordem da estrutura. O conjunto de arestas representa as ligações entre os elementos do conjunto de vértices.

A depender do problema a ser estudado, estes elementos assumem significados diferentes. Podendo os vértices significar pessoas, localizações geográficas, hosts, entre outros e as arestas representam então as relações entre esses elementos, desde distâncias físicas entre cidades, até relações de amizade ou afetividade entre pessoas.

3.1.2 Antecessor, Sucessor e Vizinho

Vértices vizinhos são aqueles que são ligados por uma aresta ou arco, quando este vértice está na extremidade inicial da aresta ele é chamado de antecessor, quando está na extremidade final, é chamado de vértice sucessor.

3.1.3 Grafos Orientados ou Não Orientados

Em um grafo não orientado, a ligação entre dois vértices é feita através de uma linha, chamada de aresta, já em um grafo orientado, a ligação é feita através de uma seta, chamada de arco, e esta representa o sentido correspondente desta ligação.



Figura 1 – Grafo Orientado



Figura 2 – Grafo Não Orientado

3.2 Representação de um grafo

Para melhor visualização de um grafo é usada a representação como nas figuras 1 e 2, com a identificação dos vértices entre círculos e as arestas ou arcos como linhas, porém para fins de cálculo os grafos são associados a matrizes. A matriz mais comumente utilizada é a matriz de adjacência, apesar de não ser a mais econômica computacionalmente.

3.2.1 Matriz de Adjacência

A matriz de adjacência é uma matriz booleana, quando utilizada em grafos, de tamanho $n \times n$ onde n é o número de vértices do grafo. Chamando de $A = a_{ij}$ a matriz adjacência, onde i e j representam o número de identificação do vértice, por exemplo, para a_{21} teremos nesta posição o valor correspondente a aresta que liga o vértice dois ao vértice um. A matriz é preenchida pela seguinte condição:

$a_{ij} = 1$, se existe ligação entre os vértices i e j

$a_{ij} = 0$, se não existe ligação entre os vértices i e j

Exemplo:

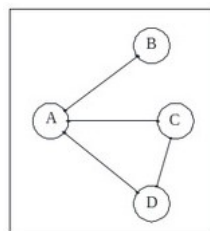


Figura 3 – Grafo Sem Peso

	A	B	C	D
A	0	1	1	1
B	1	0	0	0
C	1	0	0	1
D	1	0	1	0

Tabela 1 – Matriz Adjacência sem Peso

Caso as arestas contenham peso, a matriz de adjacência deve ser preenchida com o peso correspondente e onde não houver conexão entre os vértices, preencher com um valor que não possa ser considerado peso, como zero ou infinito.

Exemplo:

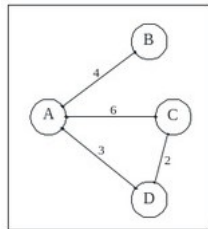


Figura 4 – Grafo Com Peso

	A	B	C	D
A	0	4	6	3
B	4	0	0	0
C	6	0	0	2
D	3	0	2	0

Tabela 2 – Matriz Adjacência com Peso

Uma observação importante é que, em grafos não direcionados, a matriz de adjacência é simétrica em relação a diagonal principal.

3.3 Aplicação para o Problema do Menor Caminho

Uma das aplicações mais comuns de grafos é para a resolução do problema de menor caminho, definindo um vértice de partida e um de chegada em um grafo, determinar qual caminho apresenta o menor custo para percorrer este caminho, considerando os pesos contidos nas arestas. Devido a importância da resolução deste tipo de problema, foram desenvolvidos algoritmos capazes de retornar como resultado o menor caminho dentro de um grafo.

3.3.1 Algoritmo de Dijkstra

Para utilizarmos o algoritmo de Dijkstra o primeiro passo é determinar o vértice de origem e o vértice de destino. Determinados a origem e o destino é feita uma análise através de interações vértice a vértice, buscando o menor, ou menores caminhos possíveis.

Partindo da origem, compara-se o custo de cada conexão ligada ao vértice do momento, para a primeira interação este vértice será a origem. Segue-se então para o próximo vértice, que apresentar o menor custo e o vértice antecessor é marcado. No novo vértice do momento é somado os custos para a chegada até ele e busca-se as conexões do vértice do momento com vértices não marcados, seguindo para o que apresente menor custo. Essa interação se repete até chegar no vértice de destino.

É importante salientar que este algoritmo não aceita custos (pesos) negativos.

3.3.2 Algoritmo de Floyd- Warshall

Este algoritmo é matricial e aceita custos negativos, mas a presença de ciclos negativos, ou seja, presença de mais de um valor negativo em uma malha de arestas, exige cuidado e conferência, já que o resultado pode não condizer com o real.

O algoritmo calcula todos os menores custos de todas as origens para todos os destinos, através de um número de interações finito e igual ao número de vértices do grafo. O algoritmo é inicializado com uma matriz onde os elementos são apenas as distâncias diretas entre os vértices, a diagonal principal, ou seja a distância entre os vértices e eles mesmos é zero e os vértices que não tem ligação direta, o elemento da matriz será infinito.

Após a montagem da matriz inicial, cada interação busca as menores distâncias entre os pontos, passando pelo vértice de número igual ao número da interação. Por exemplo, a terceira interação busca as menores distâncias entre os vértices, passando pelo vértice de número 3. Em cada interação, caso a distância seja menor do que a que já obtemos na matriz, o valor menor substitui o atual. Ao final das interações temos todas as menores distâncias entre os vértices.

4 Formulação do problema

5 Resultados Obtidos

6 Conclusão

(CARVALHO, 2008) (CARDOSO, 2005) (COELHO, 2013) (COSTA, 2011) (COSTALONGA, 2012) (OLIVEIRA; RANGEL; ARAUJO,) (FEOFILOFF; KOHAYAKAWA; WAKABAYASHI, 2011)

Referências

BARROS, E. A.; PAMBOUKIAN, S. V.; ZAMBONI, L. C. Algoritmo de dijkstra: apoio didático e multidisciplinar na implementação, simulação e utilização computacional. In: *INTERNATIONAL CONFERENCE ON ENGINEERING AND COMPUTER EDUCATION*, São Paulo. [S.l.: s.n.], 2007. Citado 2 vezes nas páginas 10 e 11.

CARDOSO, D. M. Teoria dos grafos e aplicações. *Dep. Matemática, U. Aveiro. Disponível em <http://arquivoscolar.org/bitstream/arquivo/78/1/TGA2004.pdf> (acessado em 01/12/2013)*, 2005. Citado na página 19.

CARVALHO, B. M. P. S. de. Algoritmo de dijkstra. *Universidade de Coimbra, Coimbra, Portugal*, 2008. Citado 2 vezes nas páginas 10 e 19.

COELHO, A. M. Teoria dos grafos. 2013. Citado na página 19.

COSTA, P. P. d. Teoria dos grafos e suas aplicações. Universidade Estadual Paulista (UNESP), 2011. Citado na página 19.

COSTALONGA, J. P. Grafos e aplicações. *Notas do Minicurso da 23ª Semana da Matemática do DMA-UEM, Maringá*, 2012. Citado na página 19.

FEOFILOFF, P.; KOHAYAKAWA, Y.; WAKABAYASHI, Y. Uma introdução sucinta à teoria dos grafos. *Disponível em <http://www.ime.usp.br/~pf/teoriadosgrafos>*, 2011. Citado na página 19.

GARCIA, C. Programação dinâmica: Algoritmo de bellman-ford. Universidade Federal do Rio Grande do Sul, 2011. Citado 2 vezes nas páginas 10 e 11.

OLIVEIRA, V. A. de; RANGEL, S.; ARAUJO, S. A. de. Teoria dos grafos. Citado na página 19.