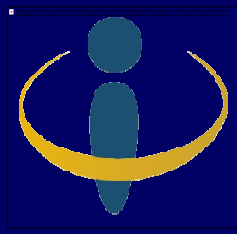
Grafos



Disciplina

Estrutura de Dados

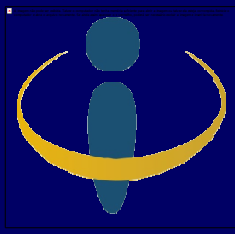
Professor Dr.

Paulo Roberto Gomes Luzzardi

Aluno

Luciano Vargas Gonçalves

Sumário



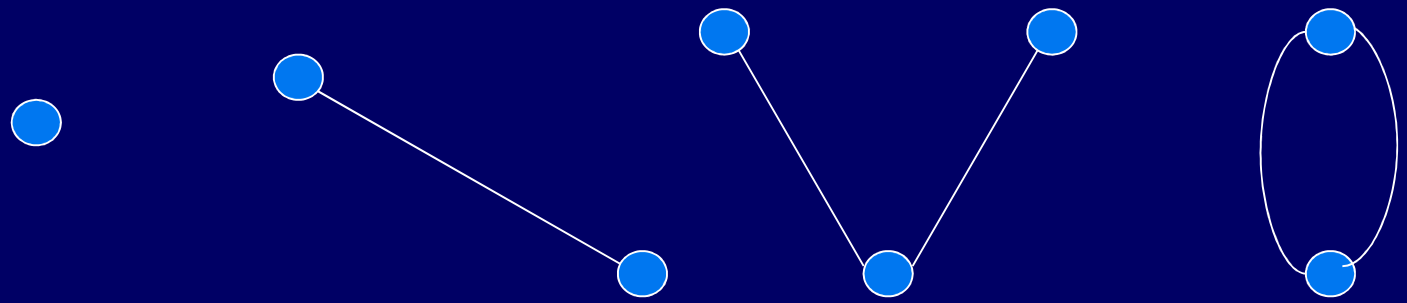
- Introdução
- Conceitos
- Histórico
- Aplicações
- Definições
  - Conceitos
  - Representação
- Percurso
- Algoritmo Dijkstra
  - Applet
- Referencias

Introdução

- **Porque estudar Grafos**
  - **Importante ferramenta matemática com aplicação em diversas áreas do conhecimento:**
    - **Genética, química, pesquisa operacional, telecomunicações, redes de computadores, conexão de vôos aéreos, restrições de precedência, fluxo de programas, dentre outros**
  - **Os estudos teóricos em grafos buscam o desenvolvimento de algoritmos mais eficientes.**

Conceitos

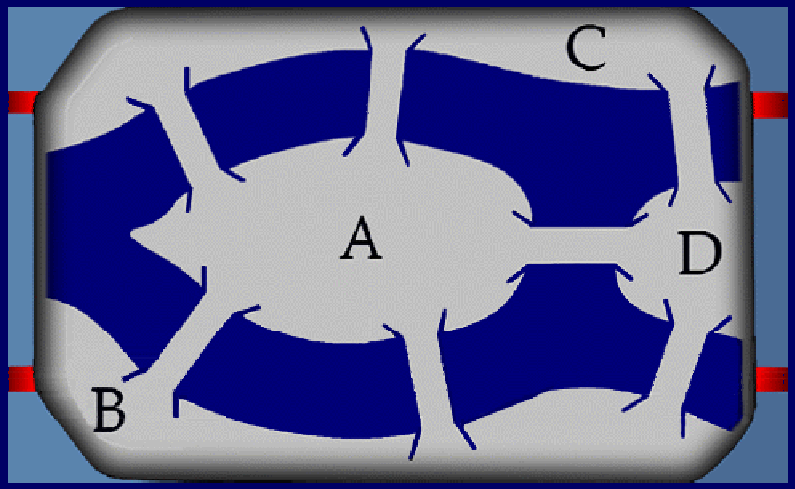
- O que são Grafos



- Tipicamente um grafo é representado como um conjunto não vazio de pontos ou vértices ligados por retas, que são chamadas de arestas.
- Abstração matemática que representa situações reais através de um diagrama.

Histórico

- **As pontes de Königsberg**



O rio Pregel divide o centro da cidade de Königsberg (Prússia no século XVII, atual Kaliningrado, Rússia) em quatro regiões. Essas regiões são ligadas por um complexo de sete (7) pontes, conforme mostra a figura.

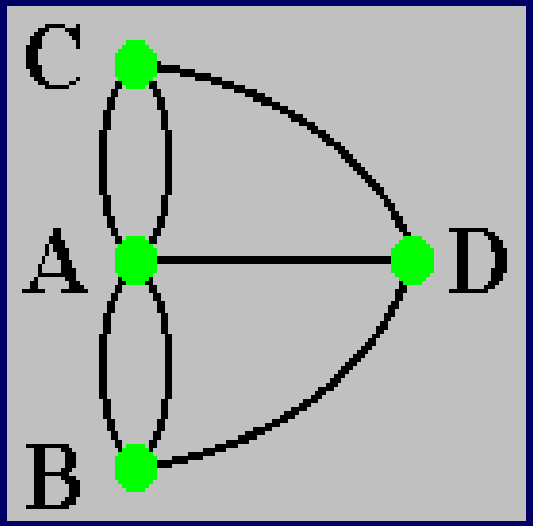
Discutia-se nas ruas da cidade a possibilidade de atravessar todas as pontes, voltando ao lugar de onde se saiu, sem repetir alguma. Havia-se tornado uma lenda popular a possibilidade da façanha quando Euler, em 1736, provou que não existia caminho que possibilitasse tais restrições.

Histórico

- As pontes de Königsberg
  - Resolvido em 1736 por Leonhard Euler
  - Necessário um modelo para representar o problema
  - Abstração de detalhes irrelevantes:
    - Área de cada ilha
    - Formato de cada ilha
    - Tipo da ponte, etc.

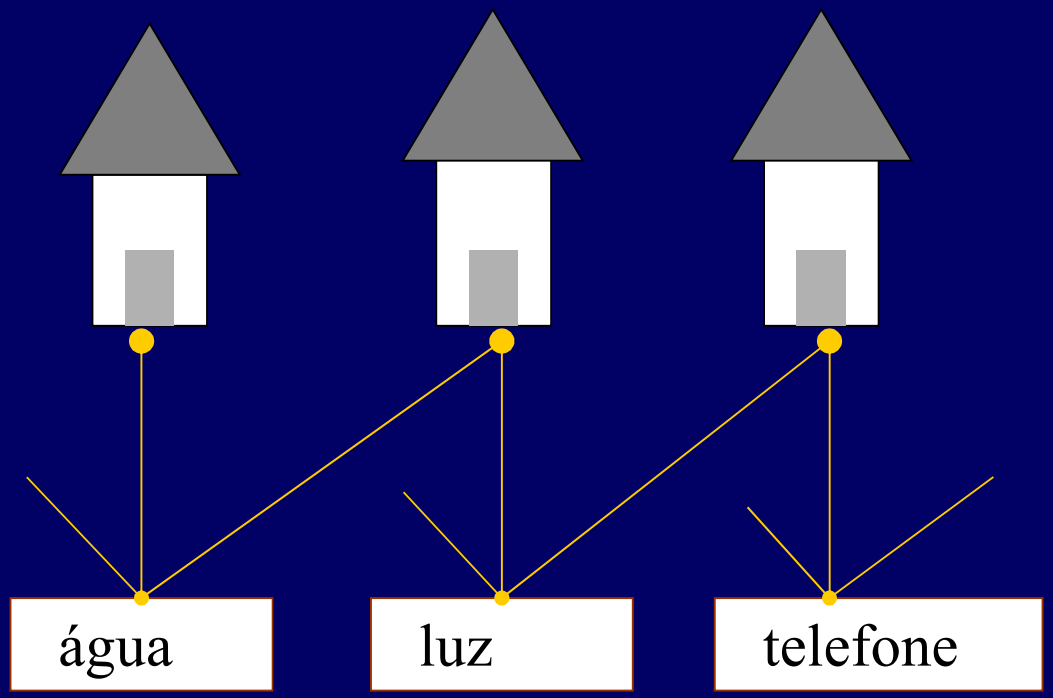
Histórico

- As pontes de Königsberg
  - Euler generalizou o problema através de um modelo de grafos



Aplicações

- O problema das três casas e três recursos



Aplicações

- Questões sobre o caminho mínimo
  - De forma a reduzir seus custos operacionais, uma empresa de transporte de cargas deseja oferecer aos motoristas de sua frota, um mecanismo que os auxilie a selecionar o melhor caminho (o de menor distância) entre quaisquer duas cidades por ela servidas, de forma a que sejam minimizados os custos de transporte.

Aplicações

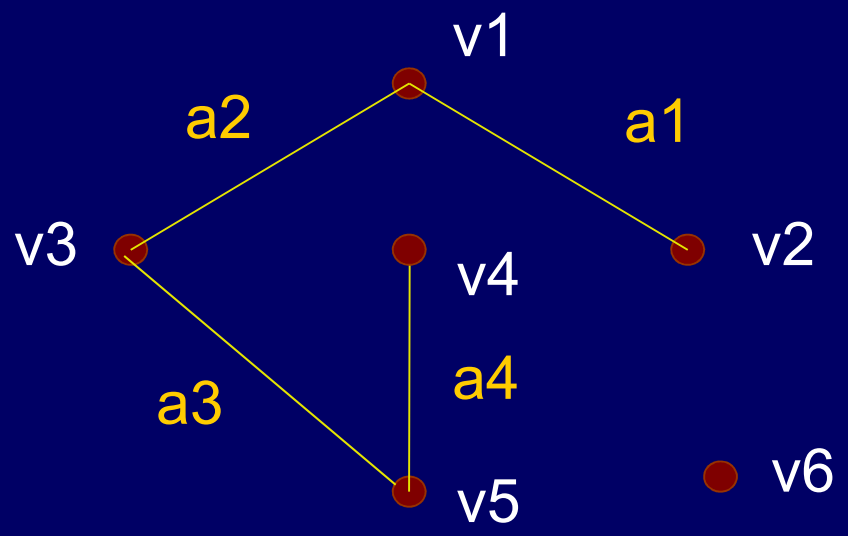

VENEZ.
GUYANA
FRENCH GUIANA
SUR.
COLOMBIA
0 400 Miles
0 400 Kilometers
Atlantic Ocean
Rio Branco
Macapá
Amazon R.
Belém
Manaus
São Luís
Fortaleza
Equator 0°
Teresina
Juruá R.
Natal
João Pessoa
Xingu R.
Tocantins R.
Campina Grande
Pôrto Velho
Madeira R.
Recife
Rio Branco
Miracema do Norte
N
PERU
Guaporé R.
Aracaju
Maceió
10° S
Mamoré R.
Cuiabá
Brasília
Salvador
BOLIVIA
Goiânia
Belo Horizonte
Corumbá
Paranaíba R.
Campo Grande
Vitória
Campinas
Nova Iguaçu
Pacific Ocean
PARAGUAY
São Paulo
Rio de Janeiro
20° S
Santos
CHILE
Curitiba
Tropic of Capricorn
Santa Maria
Florianópolis
ARGENTINA
Pôrto Alegre
Uruguaiana
30° S
Brasil
70° W
60° W
URUGUAY
40° W
30° W

Aplicações

- Problema do ciclo Hamiltoniano (Hamilton 1859) .
  - Existem n cidades. Cada par de cidades pode ser adjacente ou não arbitrariamente. Partindo de uma cidade qualquer, o problema consiste em determinar um trajeto que passe exata-mente uma vez em cada cidade e retorne ao ponto de partida. (Caixeiro Viajante)

Definições

- Grafos – Elementos básicos
  - Vértices ou nós (v1,v2,v3,v4,v5,v6)
  - Arestas (a1,a2,a3,a4)



Definições

- G = (V,E)
  - *V* é um conjunto finito não-vazio de vértices (ou nós).
  - *E* é um conjunto de pares não ordenados de elementos distintos de *V*, chamados de arestas.
  - Cada aresta *e* pertencente ao conjunto *E* será denotada pelo par de vértices *{x,y}* que a forma.
  - Dizemos que os vértices *x* e *y* são extremos (ou extremidades) da aresta *e*.

Exemplos

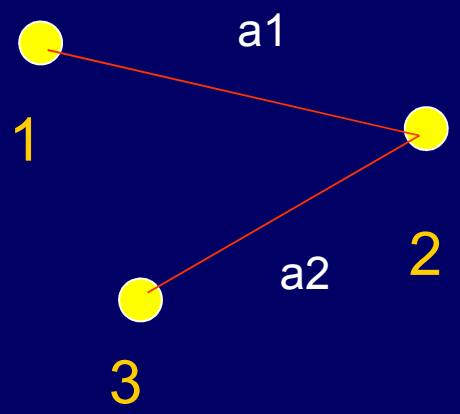
- G = (V, E)

G = ({ 1 } , Ø)

1

Gráfico Trivial

G = ({1,2,3} , {(1,2), (2,3)})



Conceitos

- Dois vértices *x* e *y* são ditos adjacentes ou vizinhos se existe uma aresta unindo-os.

- Os vértices *u* e v são ditos incidentes na aresta *e*, se eles são extremos de *e*.

  Duas arestas são adjacentes se elas têm ao menos um vértice em comum. A aresta *e={x,y}* é incidente a ambos os vértices *x* e *y*.

- Quando uma aresta possui indicação de sentido (uma seta), ela é chamada de arco.

Conceitos

- **Orientação** é a direção para a qual uma seta aponta, um grafo deste tipo é chamado grafo dirigido ou orientado.
- **Cardinalidade** (ordem) de um conjunto de vértices é igual a quantidade de seus elementos (grafo denso e pouco povoado). A **Dimensão** (A) é o número de arestas do grafo.
- **Laço** é uma aresta que retorna ao mesmo vértice.

Conceitos

- **Passeio** é uma seqüência de vértices e arestas
- **Caminho** é um passeio sem vértices repetidos.
  - **Circuito** é um caminho de comprimento não nulo fechado, ou seja, tem os vértices extremos iguais (é um passeio onde v0 = vk.).
- **Trajeto** é um passeio sem arestas repetidas.
- Dois **vértices são adjacentes** se estão ligados por uma aresta. Um **vértice é dito isolado** se não existe aresta incidente sobre ele.

Conceitos

- **Grau** de um vértice, em um grafo não-dirigido, é o número de arestas incidentes ao vértice.
- Em um grafo dirigido, pode-se dividir o grau em dois: **grau de emissão** (número de arestas que saem do vértice) e **grau de recepção** (número de arestas que chegam no vértice).

Grau (A) = 3
Grau (B) = 1

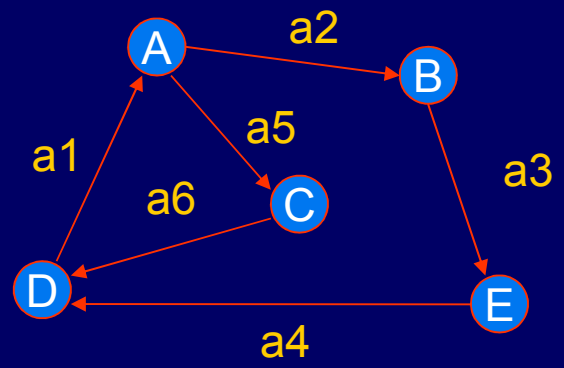
Conceitos

- ***Toda árvore é um grafo, mas nem todo grafo é uma árvore.*** **o grafo é uma árvore se, e somente se, não existirem arestas de retorno.**

- **Um grafo onde existe um número associado a cada arco (peso) é chamado de rede ou grafo ponderado.**

Conceitos

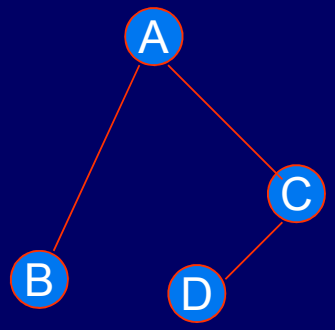
G = (V,E)
V = {A,B,C,D,E}
E = {a1,a2,a3,a4,a5,a6}



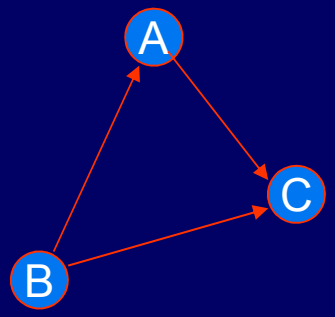
Cardinalidade = 5
Grau (A) = 3
Dimensão = 6
Passeio = { (D,a1), (A,a2), (B,a3), (E,a4) }
Caminho = (D, A, B, E)
Circuito = (D, A, B, E, D)

Tipos

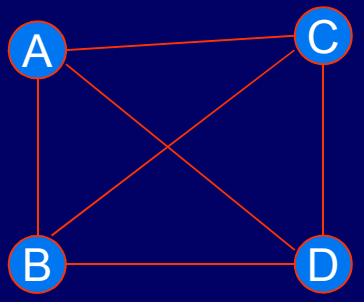
- **Simples**



- **Dirigido**



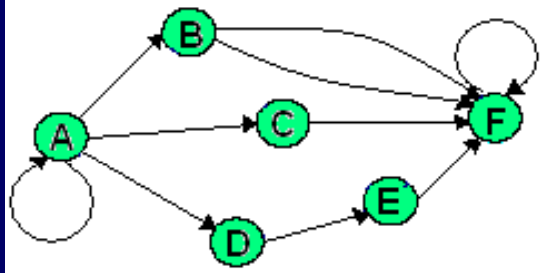
- **Completo**



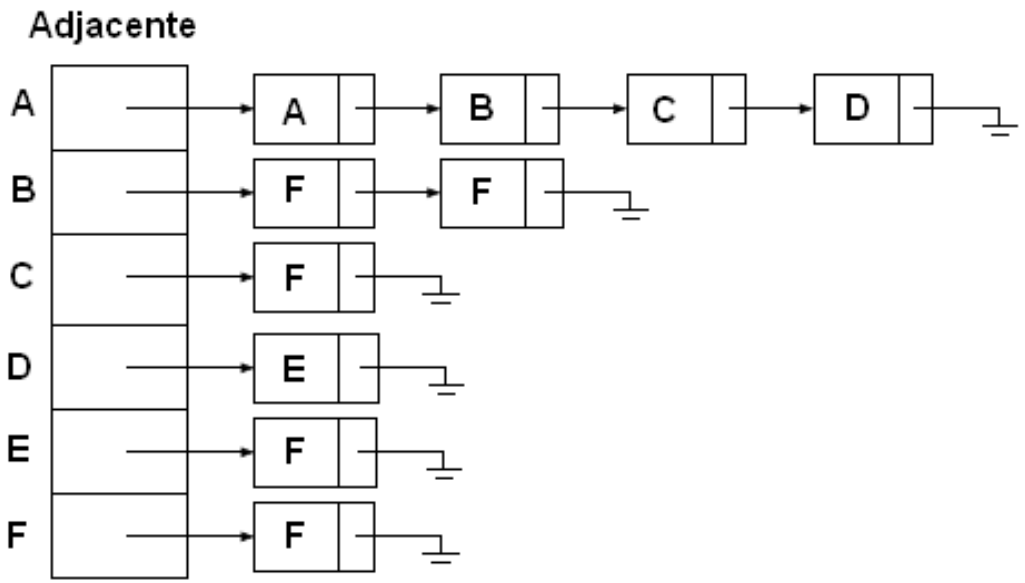
- **Multigrafo -** Loop
- **Valorado -** Peso
- **Planar -** Sem cruzamento
- **Regular –** Mesmo Grau

Representação

- Lista de Adjacência – Grafo Dirigido

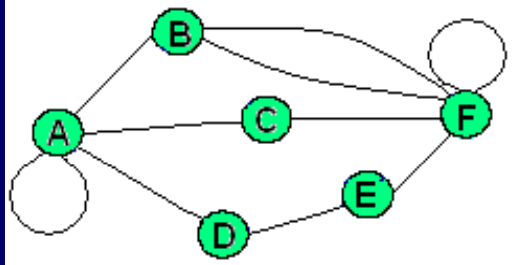


Adjacente[A] = [A, B, C, D]
Adjacente[B] = [F, F]
Adjacente[C] = [F]
Adjacente[D] = [E]
Adjacente[E] = [F]
Adjacente[F] = [F]

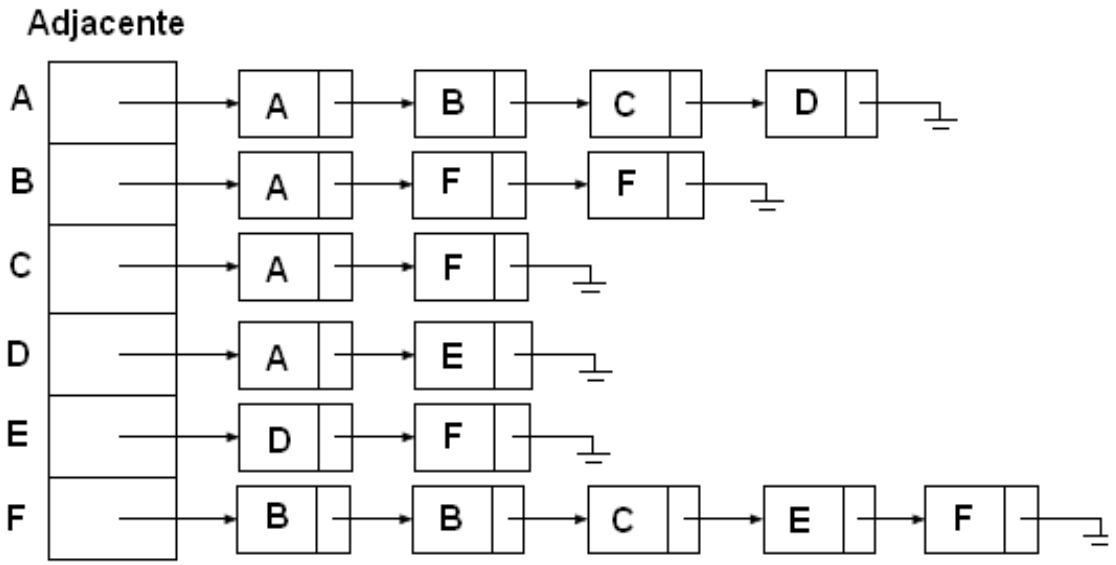


Representação

- Lista de Adjacência – Grafo não Dirigido

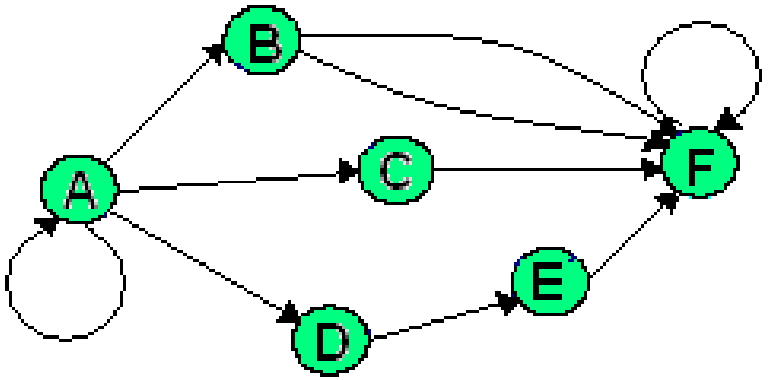


Adjacente[A] = [A, B, C, D]
Adjacente[B] = [A, F, F]
Adjacente[C] = [A, F]
Adjacente[D] = [A, E]
Adjacente[E] = [D, F]
Adjacente[F] = [B, B, C, E, F]



Representação

- Matriz de Adjacência – Grafo Dirigido

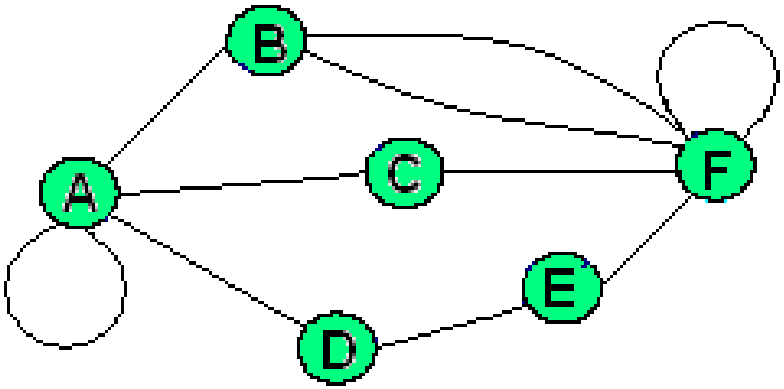


A

	A	B	C	D	E	F
A	1	1	1	1	0	0
B	0	0	0	0	0	2
C	0	0	0	0	0	1
D	0	0	0	0	1	0
E	0	0	0	0	0	1
F	0	0	0	0	0	1

Representação

- Matriz de Adjacência – Grafo não Dirigido



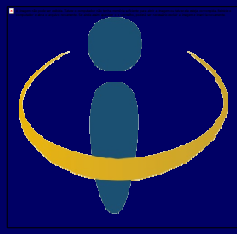
A

	A	B	C	D	E	F
A	1	1	1	1	0	0
B	1	0	0	0	0	2
C	1	0	0	0	0	1
D	1	0	0	0	1	0
E	0	0	0	1	0	1
F	0	2	1	0	1	1

Percurso

- Percurso em Amplitude
- Percurso em Profundidade
- Em ambos os percursos parte-se de um nodo qualquer escolhido arbitrariamente e visita-se este nodo. A seguir, considera-se cada um dos nodos adjacentes ao nodo escolhido.

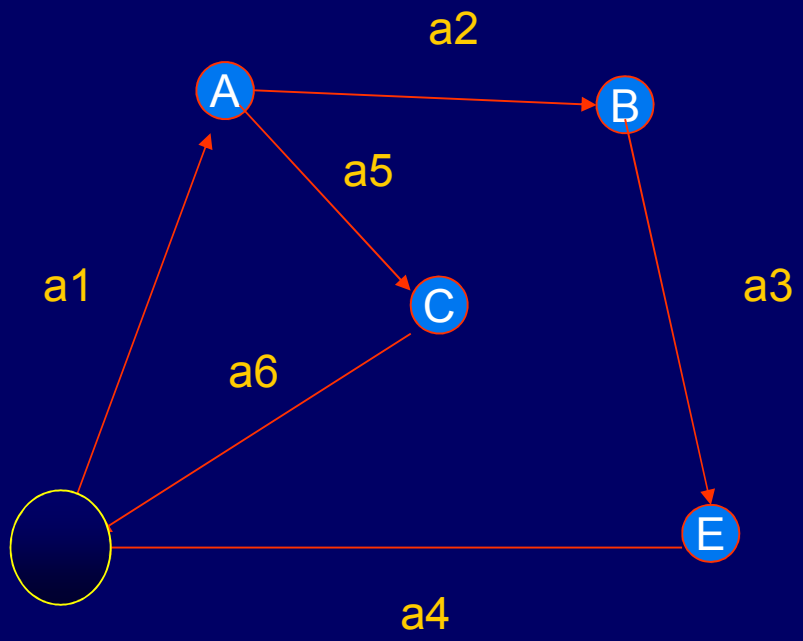
Percurso em Amplitude



1. Seleciona-se um vértice para iniciar o caminhamento.
2. Visitam-se os vértices adjacentes, marcando-os como visitados.
3. Coloca-se cada vértice adjacente numa fila.
4. Após visitar os vértices adjacentes, o primeiro da fila torna-se o novo vértice inicial. Reinicia-se o processo.
5. O caminhamento termina quanto todos os vértices tiverem sido visitados ou o vértice procurado for encontrado

Percurso em Amplitude

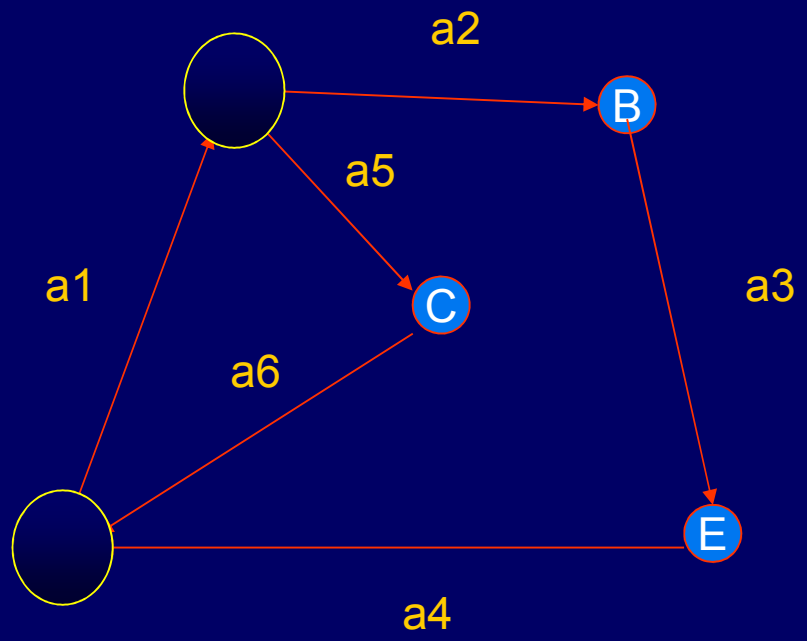
- Exemplo



Fila

Percurso em Amplitude

- Exemplo

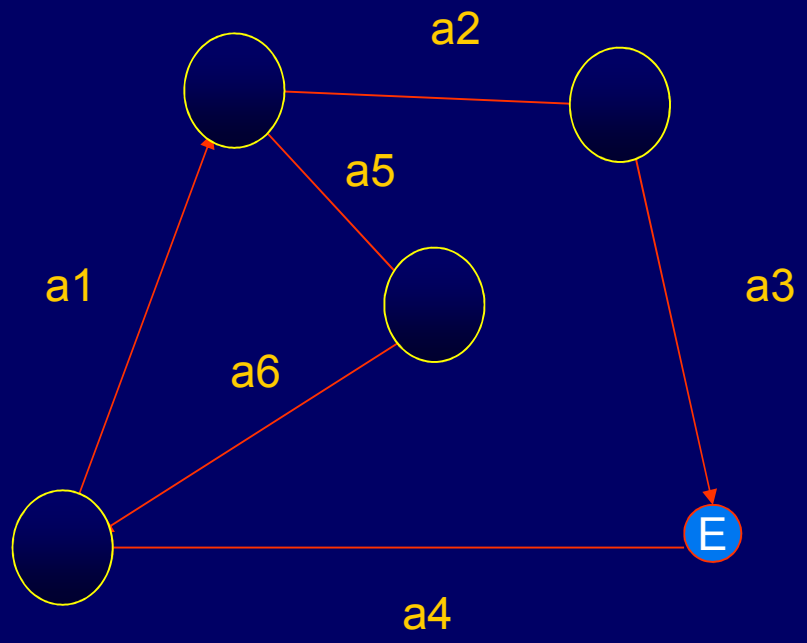


Fila

A

Percurso em Amplitude

- Exemplo

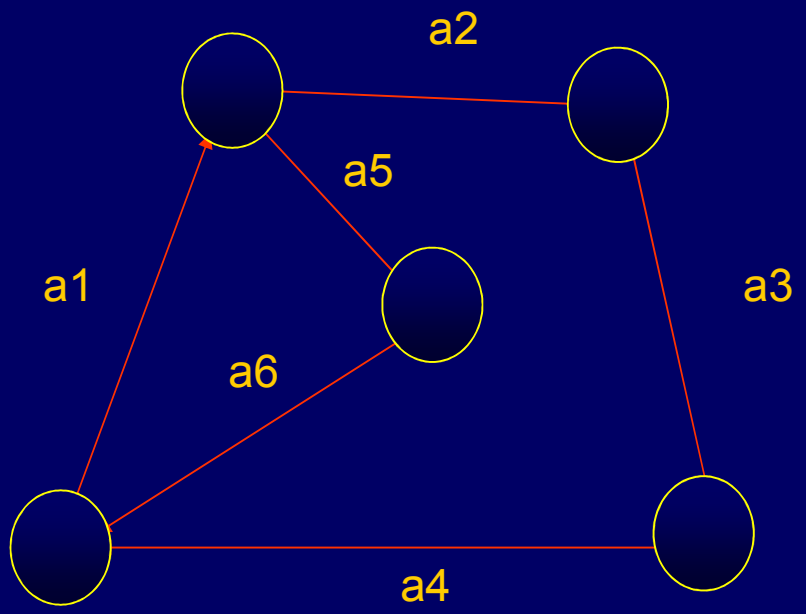


Fila

C B

Percurso em Amplitude

- Exemplo

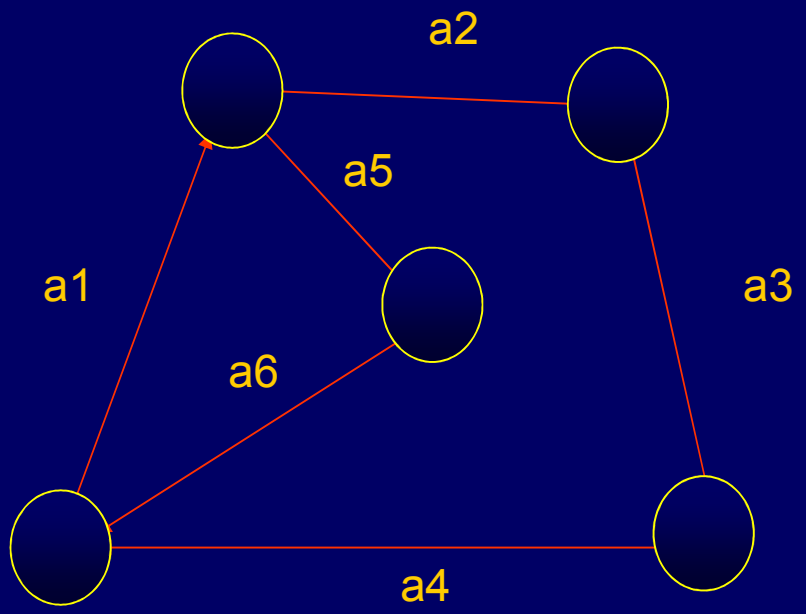


Fila

E C

Percurso em Amplitude

- Exemplo

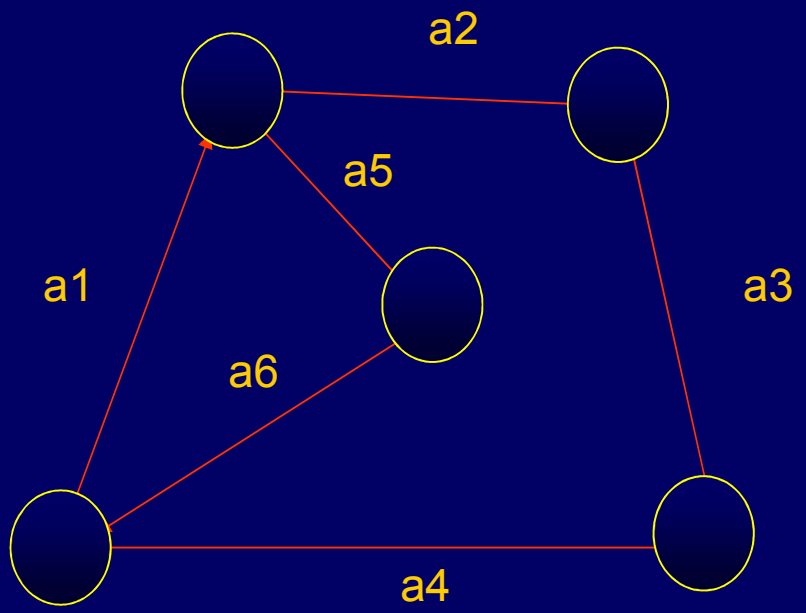


Fila

E

Percurso em Amplitude

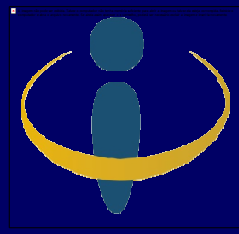
- Exemplo



Fila

Seqüência: D, A, B, C, E

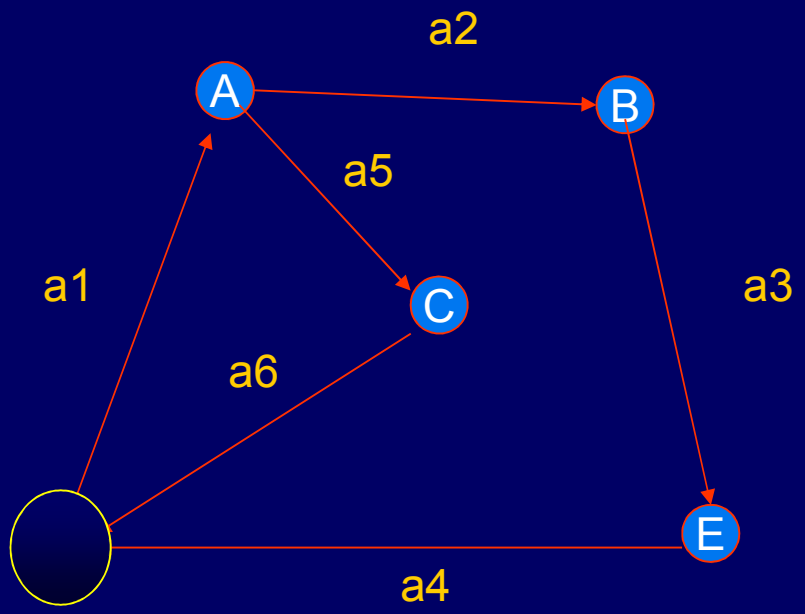
Percurso em Profundidade



1. Seleciona-se um vértice para iniciar o caminhamento.
2. Visita-se um primeiro vértice adjacente, marcando-o como visitado.
3. Coloca-se o vértice adjacente visitado numa pilha.
4. O vértice visitado torna-se o novo vértice inicial.
5. Repete-se o processo até que o vértice procurado seja encontrado ou não haja mais vértices adjacentes. Se verdadeiro, desempilha-se o topo e procura-se o próximo adjacente, repetindo o algoritmo.
6. O processo termina quando o vértice procurado for encontrado ou quando a pilha estiver vazia e todos os vértices tiverem sido visitados.

Percurso em Profundidade

- Exemplo

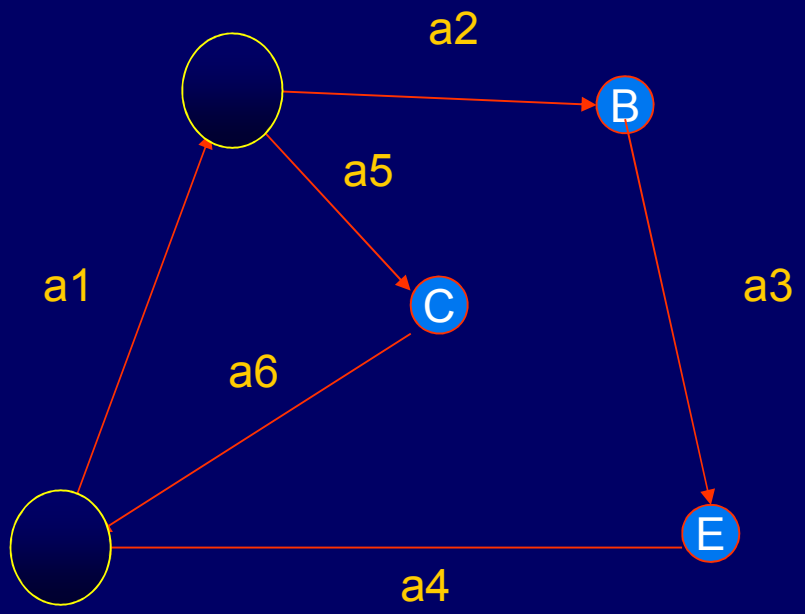


Pilha

D

Percurso em Profundidade

- Exemplo

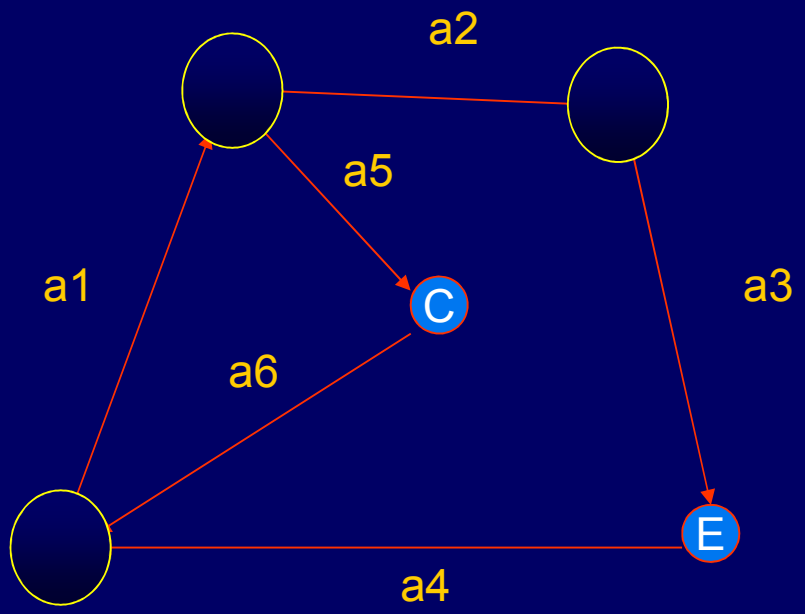


Pilha

A
D

Percurso em Profundidade

- Exemplo

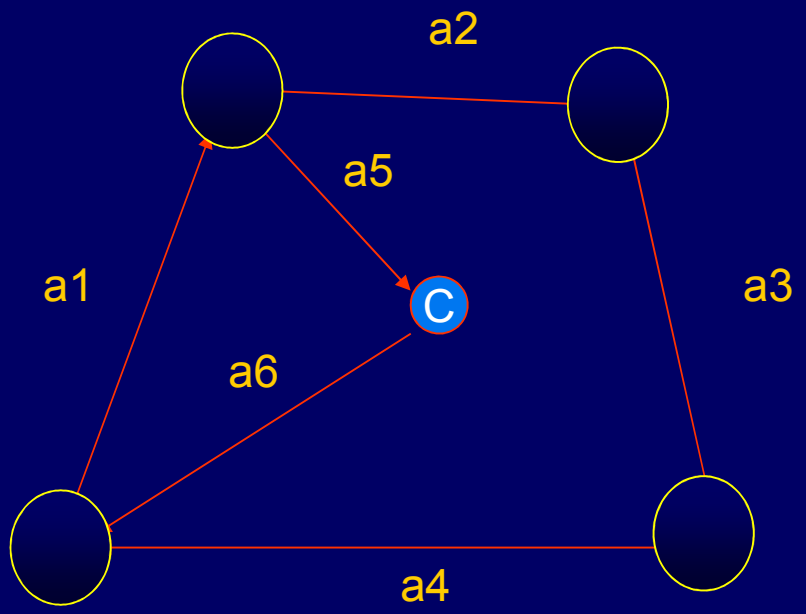


Pilha

B
A
D

Percurso em Profundidade

- Exemplo

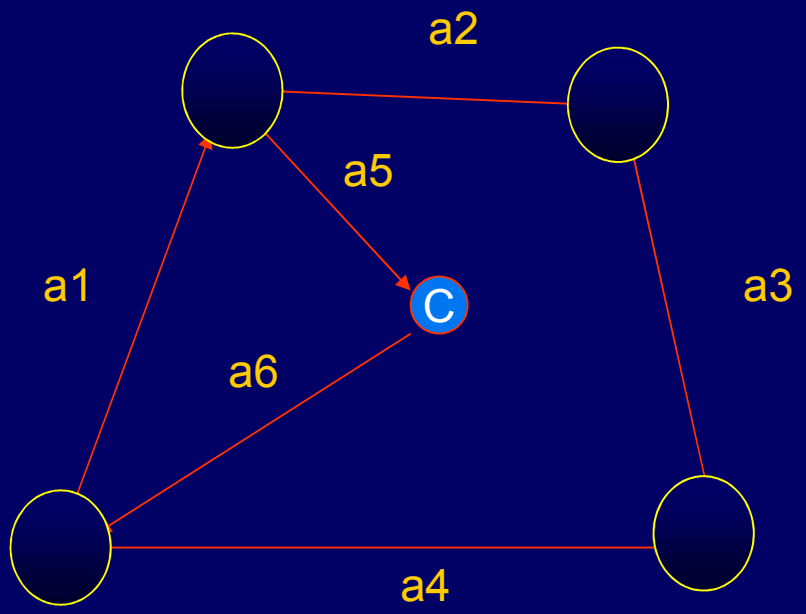


Pilha

E
B
A
D

Percurso em Profundidade

- Exemplo

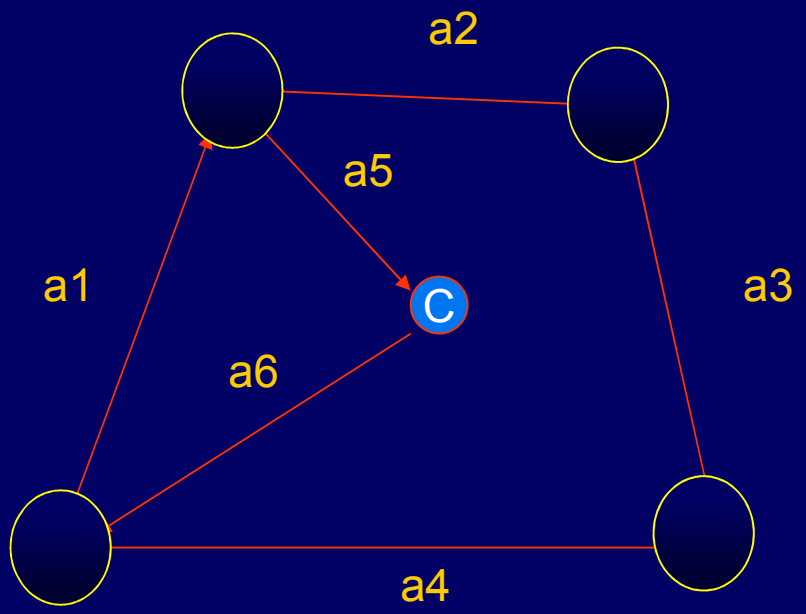


Pilha

B
A
D

Percurso em Profundidade

- Exemplo

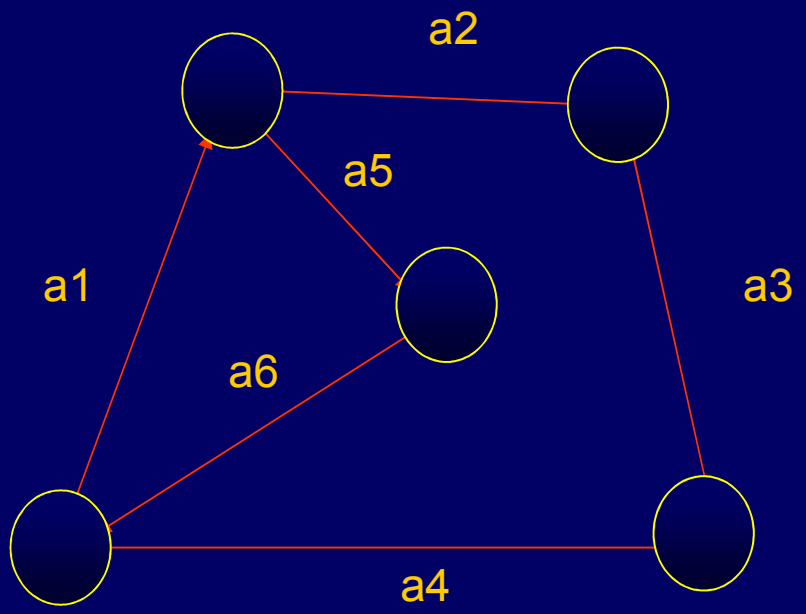


Pilha

A
D

Percurso em Profundidade

- Exemplo

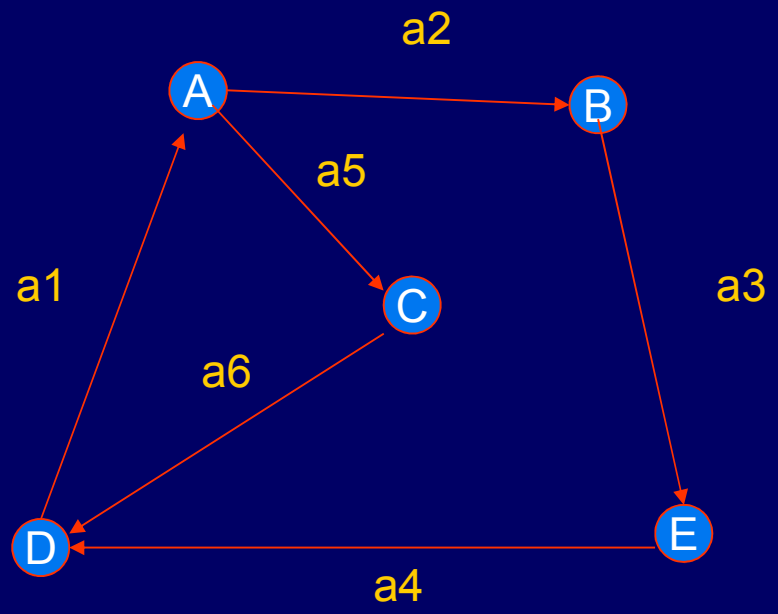


Pilha

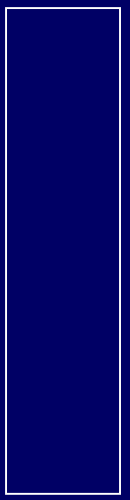
C
A
D

Percurso em Profundidade

- Exemplo



Pilha



Seqüência: D, A, B, E, C

Caminho Mínimo

- Clássicos da Ciência da Computação.
  - Este problema consiste, genericamente, em encontrar o caminho de menor custo entre dois nós da rede, considerando a soma dos custos associados aos arcos percorridos.

Algoritmo Dijkstra (1959).

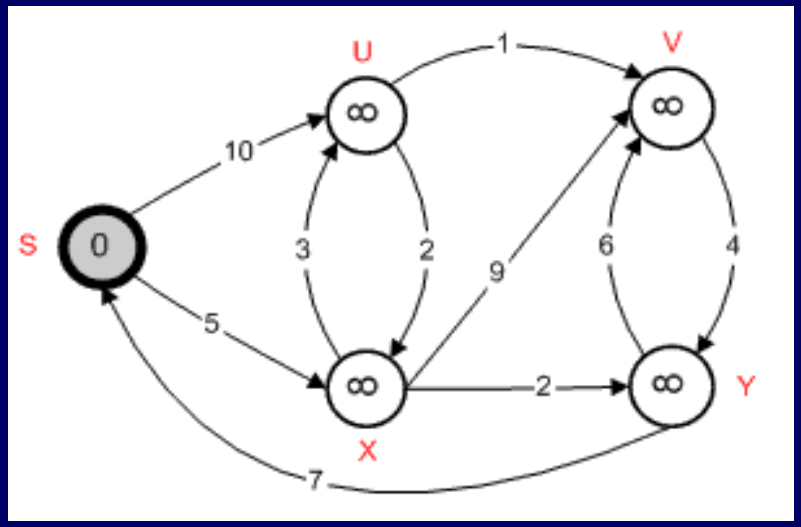
- O nome se origina de seu inventor, o cientista da computação Edsger Dijkstra, que solucionou o problema do caminho mais curto em grafo dirigido com arestas de peso não negativo, em tempo computacional O([m+n]log n) onde m é o número de arestas e n é o número de vértices. O algoritmo que serve para resolver o mesmo problema em um grafo com pesos negativos é o algoritmo de Bellman-Ford.
- Um exemplo prático de problema que pode ser resolvido pelo algoritmo de Dijkstra é: alguém precisa se deslocar de uma cidade para outra. Para isso, ela dispõe de várias estradas, que passam por diversas cidades. Qual delas oferece uma trajetória de menor caminho?

Algoritmo Dijkstra (1959).

- Atribua valor zero à estimativa do custo mínimo do vértice s (a raiz da busca) e infinito às demais estimativas;
- Atribua um valor qualquer aos precedentes (o precedente de um vértice t é o vértice que precede t no caminho de custo mínimo de s para t);
- Enquanto houver vértice aberto:
  - seja k um vértice ainda aberto cuja estimativa seja a menor dentre todos os vértices abertos;
  - feche o vértice k
  - Para todo vértice j ainda aberto que seja sucessor de k faça:
    - some 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, substitua-a e anote k como precedente de j.

Algoritmo Dijkstra (1959).

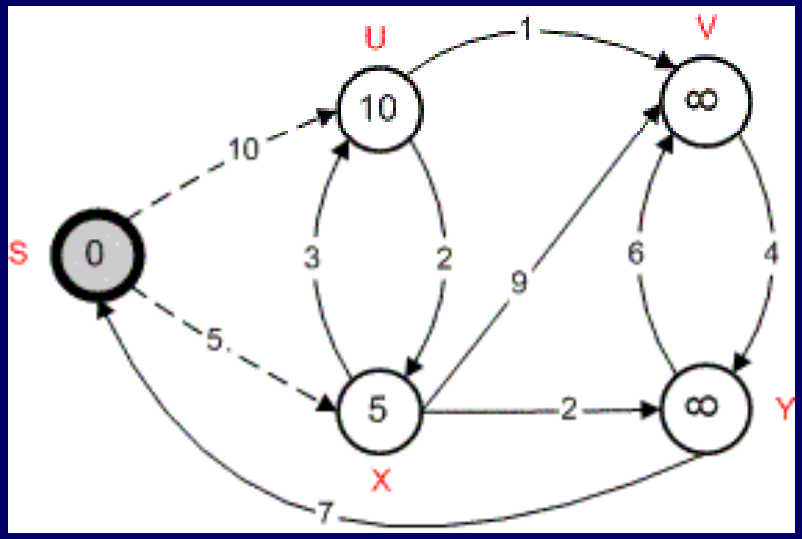
- Exemplo – Estado inicial



vértice	Perm	Dist	Path
s	sim	0	-
u	não	∞	-
x	não	∞	-
v	não	∞	-
y	não	∞	-

Algoritmo Dijkstra (1959).

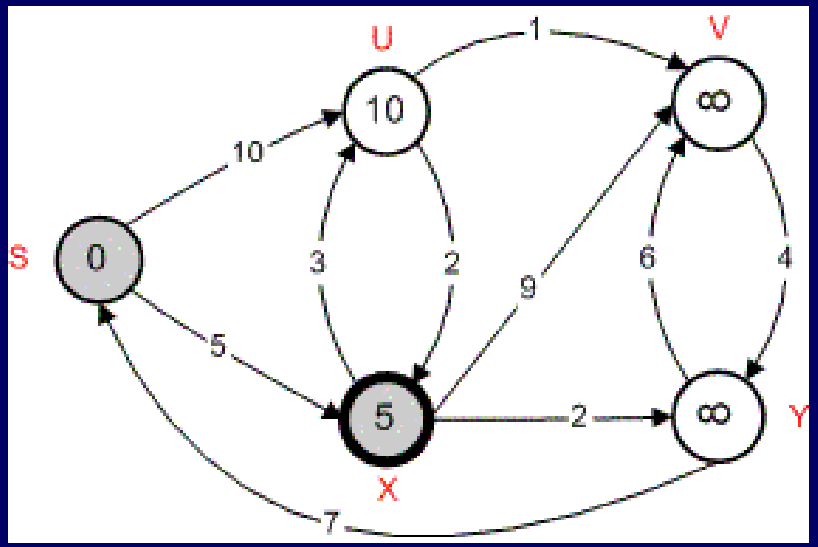
- Exemplo



vértice	Perm	Dist	Path
s	sim	0	-
u	não	10	s
x	não	5	s
v	não	∞	-
y	não	∞	-

Algoritmo Dijkstra (1959).

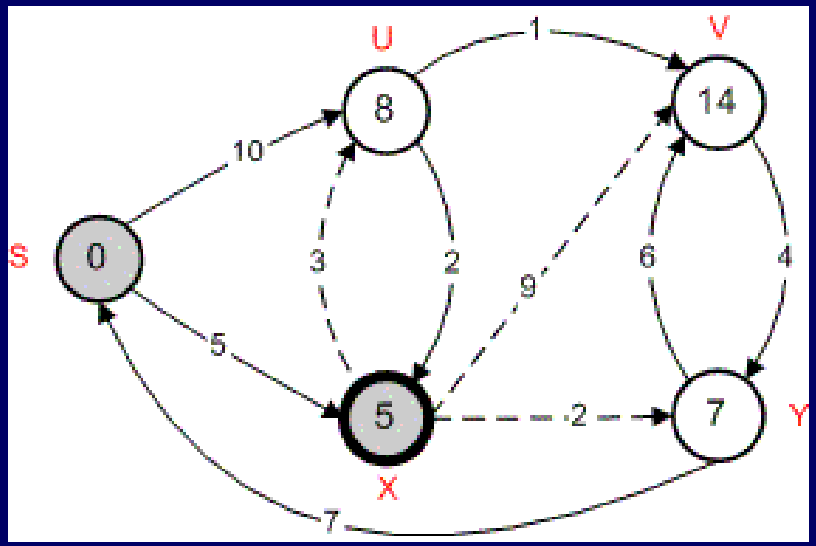
- Exemplo



vértice	Perm	Dist	Path
s	sim	0	-
u	não	10	s
x	não	5	s
v	não	∞	-
y	não	∞	-

Algoritmo Dijkstra (1959).

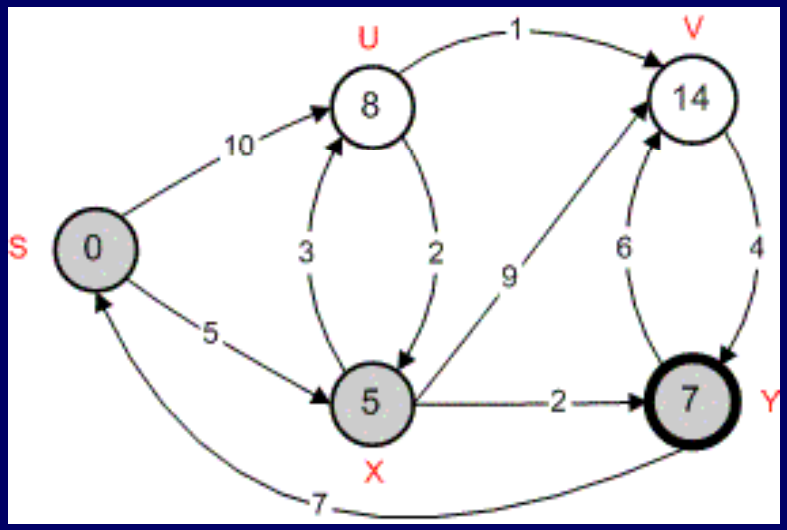
- Exemplo



vértice	Perm	Dist	Path
s	sim	0	-
u	não	8	x
x	sim	5	s
v	não	14	x
y	não	7	x

Algoritmo Dijkstra (1959).

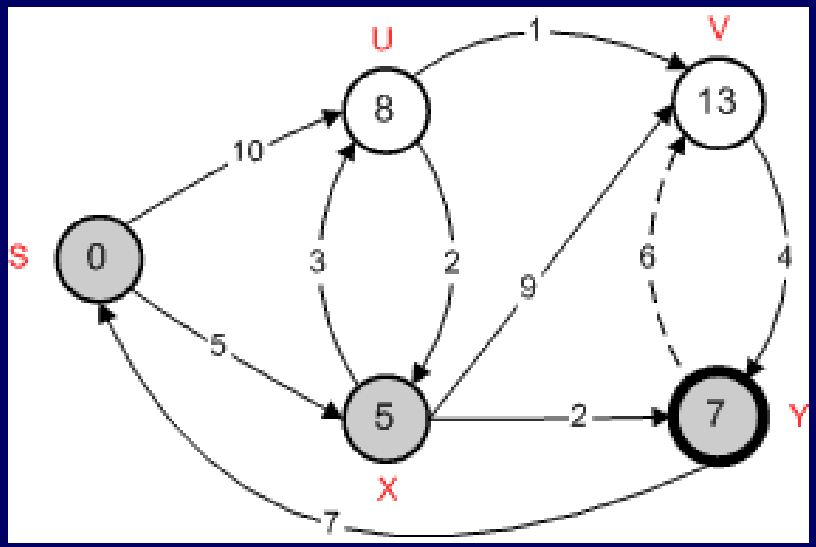
- Exemplo



vértice	Perm	Dist	Path
s	sim	0	-
u	não	8	x
x	sim	5	s
v	não	14	x
y	não	7	x

Algoritmo Dijkstra (1959).

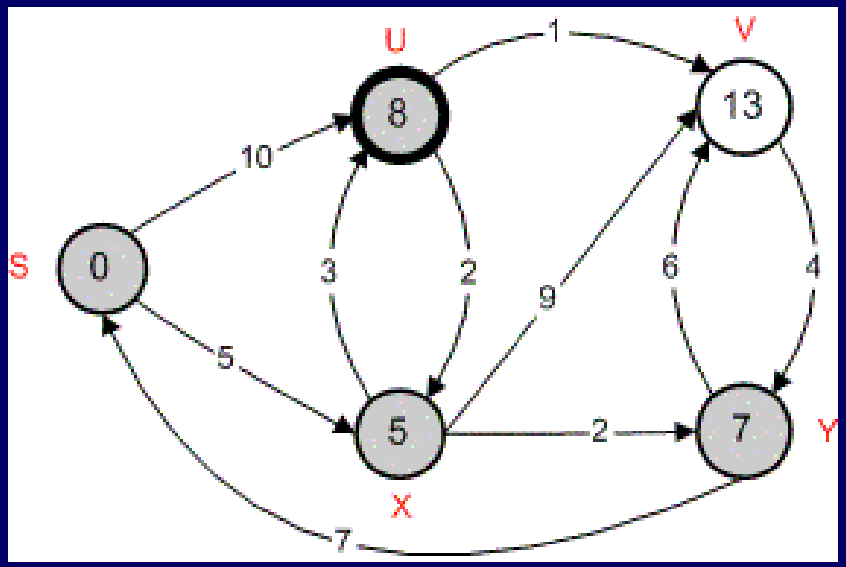
- Exemplo



vértice	Perm	Dist	Path
s	sim	0	-
u	não	8	x
x	sim	5	s
v	não	13	y
y	sim	7	x

Algoritmo Dijkstra (1959).

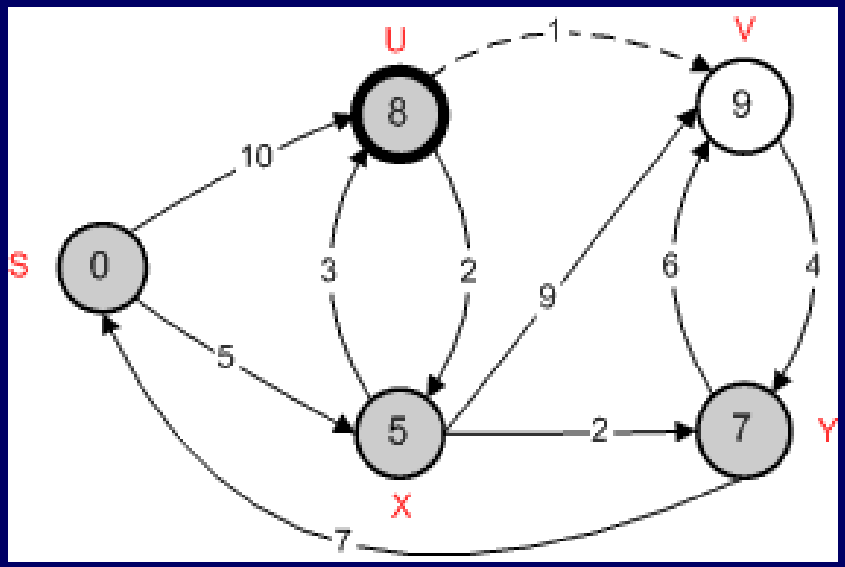
- Exemplo



vértice	Perm	Dist	Path
s	sim	0	-
u	não	8	x
x	sim	5	s
v	não	13	y
y	sim	7	x

Algoritmo Dijkstra (1959).

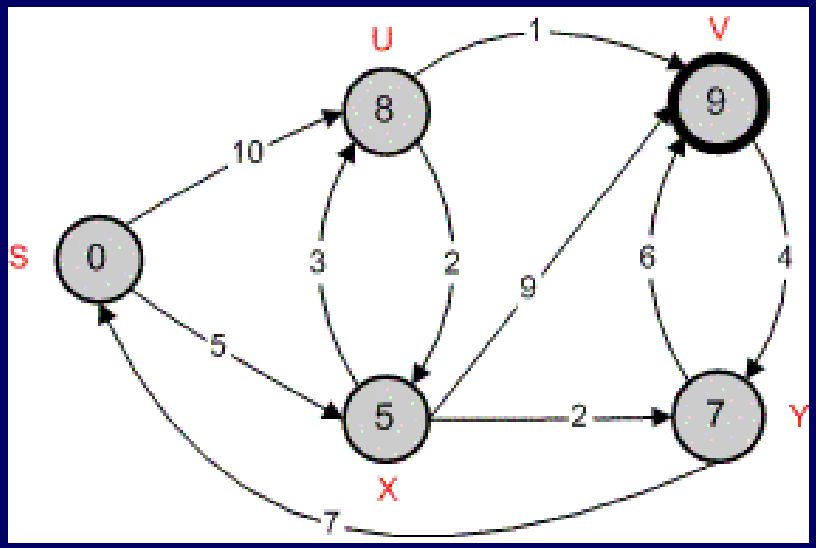
- Exemplo



vértice	Perm	Dist	Path
s	sim	0	-
u	sim	8	x
x	sim	5	s
v	não	9	u
y	sim	7	x

Algoritmo Dijkstra (1959).

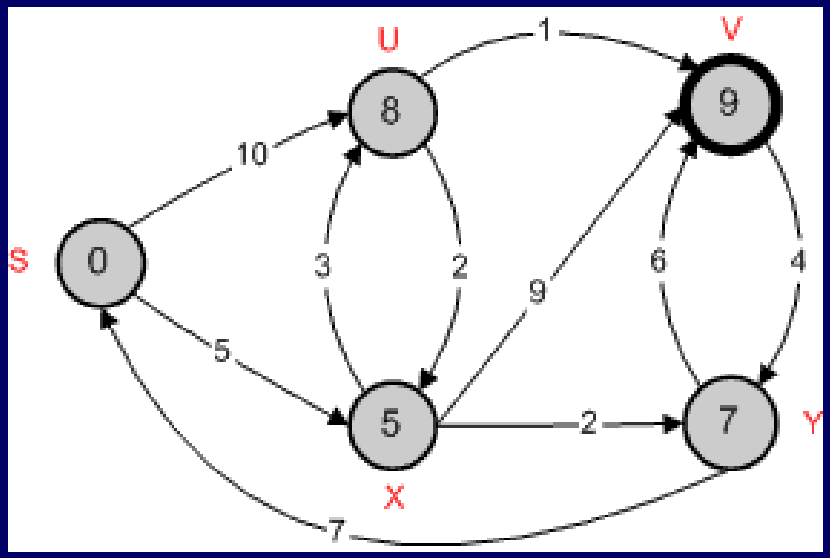
- Exemplo



vértice	Perm	Dist	Path
s	sim	0	-
u	sim	8	x
x	sim	5	s
v	não	9	u
y	sim	7	x

Algoritmo Dijkstra (1959).

- Exemplo



vértice	Perm	Dist	Path
s	sim	0	-
u	sim	8	x
x	sim	5	s
v	sim	9	u
y	sim	7	x

Applet

- Applet Java Dijkstra
  - http://www-b2.is.tokushima-u.ac.jp/~ikeda/suuri/dijkstra/Dijkstra.shtml
- Implementação C
  - http://www.mis-algoritmos.com/ejemplos/source-154.html

Referencias

- Livros
  - Estrutura de Dados – Luzzardi, Paulo Roberto – Universidade Católica de Pelotas.
- Sites

http://www-b2.is.tokushima-u.ac.jp/~ikeda/suuri/dijkstra/Dijkstra.shtml

http://www.lcad.icmc.usp.br/~nonato/ED/Dijkstra/node84.html

http://pt.wikipedia.org/wiki/Teoria_dos_grafos

http://www.cin.ufpe.br/~if670/2-2005/Aula1grafos.ppt