Fundamentos de Teoria de Grafos

Motivação

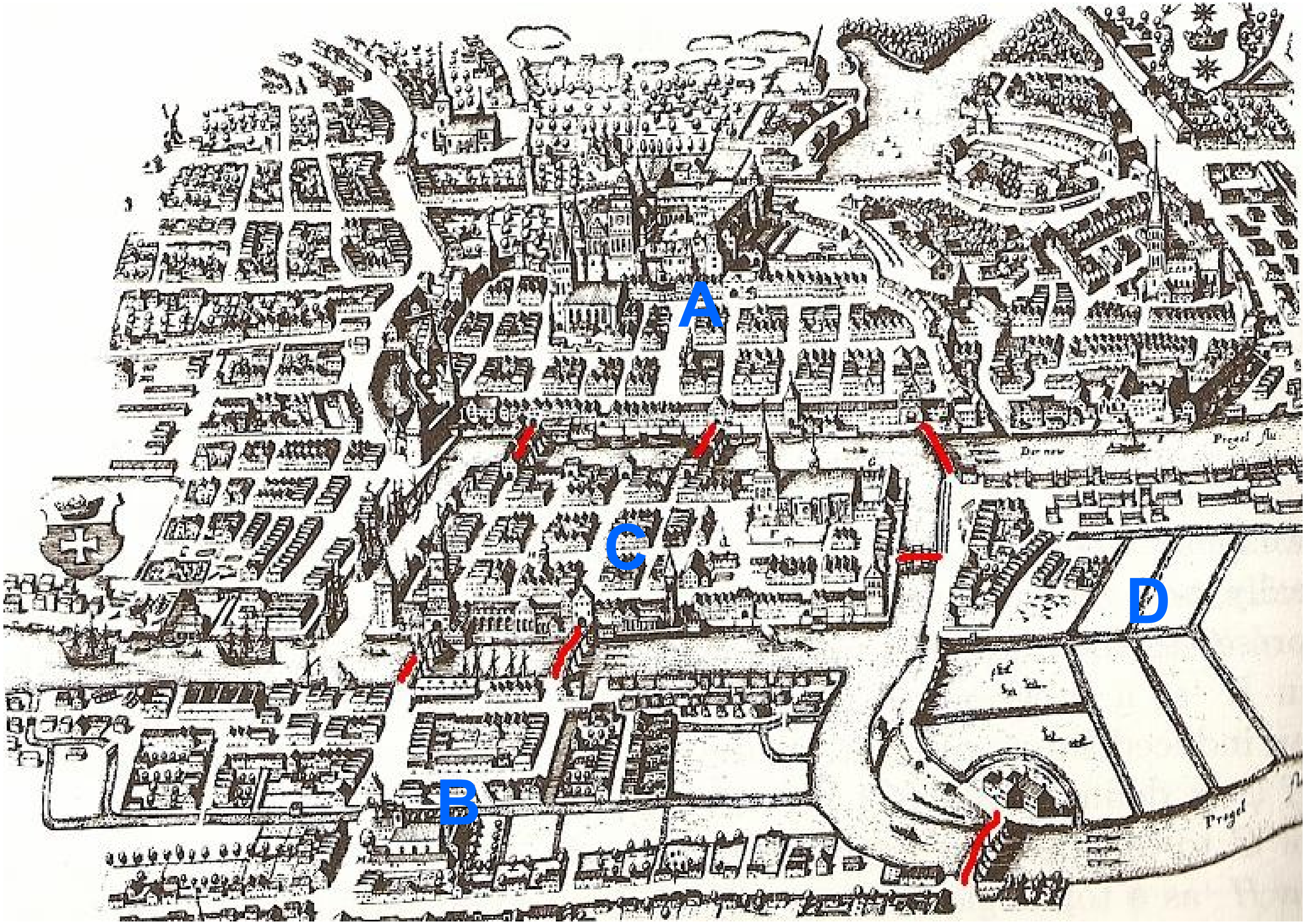
- Por que estudar grafos?
  - Importante ferramenta matemática com aplicação em diversas áreas do conhecimento
  - Utilizados na definição e/ou resolução de problemas
  - Existem centenas de problemas computacionais que empregam grafos com sucesso.

Primeiras motivações da área ...

- Königsberg Bridge Problem

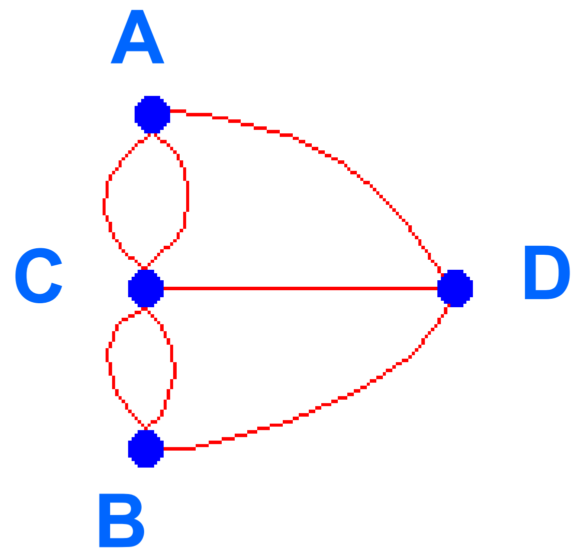
Duas ilhas C e D, existentes no rio Pregel em Königsberg (Rússia), foram ligadas às margens do rio (A e B) através de 7 pontes. É possível iniciar uma caminhada a partir de um dos blocos de terra (A, B, C ou D), passar por cada uma das pontes e voltar ao ponto de partida sem nadar pelo rio?

As pontes de Königsberg



O problema das 7 pontes

- 1736: Euler foi o primeiro a representar esse problema usando grafos e provou que uma solução para o mesmo não existe!



Introdução

- Um grafo G=(V,E) consiste em um conjunto V de vértices e um conjunto E de pares de vértices ou arestas.
- Os grafos são uma forma de modelar os problemas.
- Muitos problemas algorítmicos são simplificados ao pensarmos neles em termos de grafos.

Introdução

- *A teoria dos Grafos fornece uma linguagem para tratarmos com as propriedades dos grafos.*

- *Conhecer diferentes problemas algorítmicos em grafos é melhor do que entender os detalhes de algoritmos particulares em grafos.*

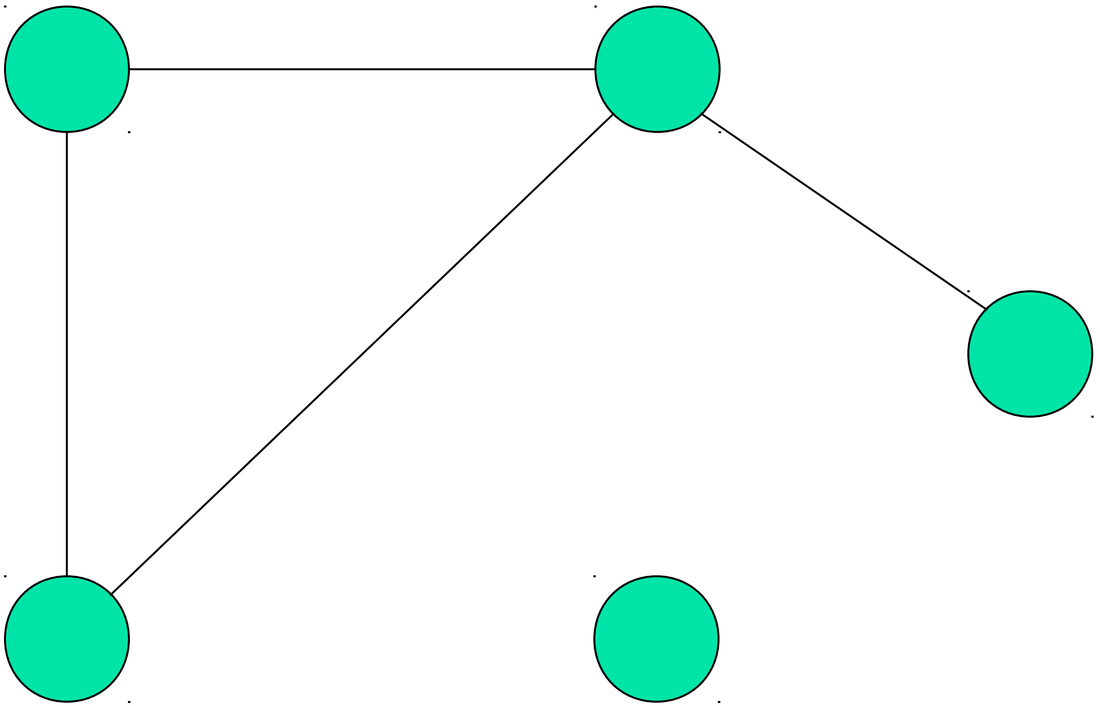
Conteúdo

- Estruturas de dados básicas e operações de atravessamento.
- Algoritmos mais sofisticados para: caminhos mais curtos e árvore geradora.

Observações

- Grafos podem ser utilizados para modelar uma variedade de estruturas e relações.
- Muitas aplicações de grafos podem ser reduzidas a propriedades padrão de grafos e usando algoritmos bem conhecidos.
- Busca em profundidade e busca em largura fornecem mecanismos para visitar cada aresta e cada vértice do grafo.

O grafo de relacionamentos



O grafo de relacionamentos

- *Se sou seu amigo, você é meu amigo?*
- Grafos orientados e não-orientados
- Laços, multi-arestas, grafos simples
- Grafos com pesos
- Caminho em um grafo

Caminho mínimo

O grafo de relacionamentos

- *Existe um caminho de relacionamento entre quaisquer duas pessoas no mundo?*
-
-
-

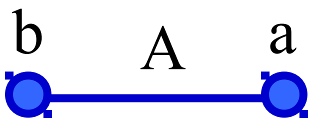
Componentes conexas

Clique

Circuitos

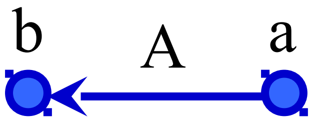
Vértices Adjacentes

- *Grafo:*



  - o vértice a é adjacente ao vértice b;
  - o vértice b é adjacente ao vértice a.

- Dígrafo:



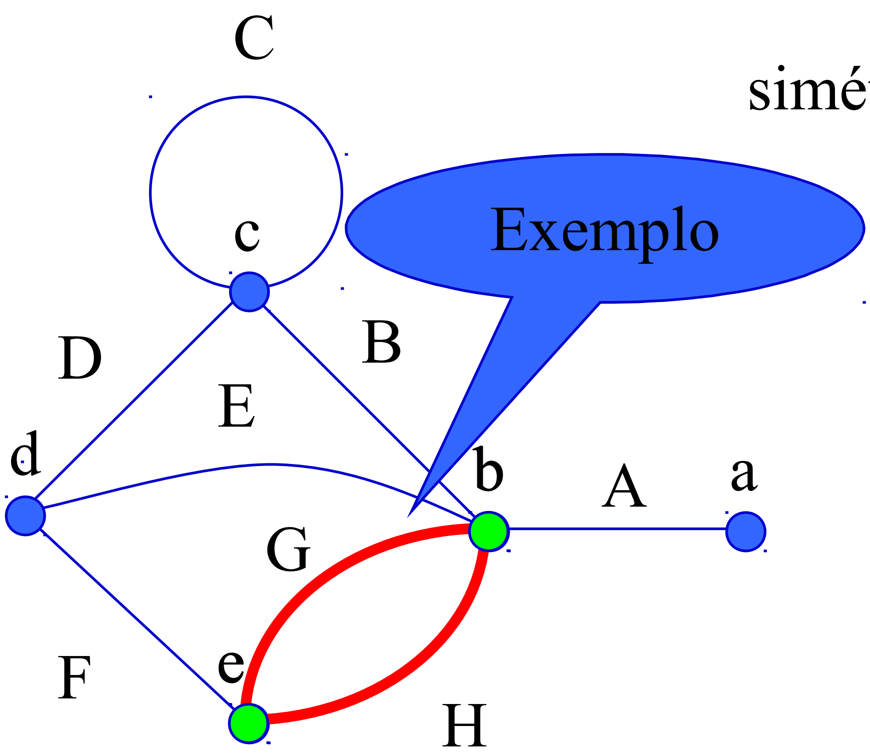
  - **o vértice a <u>não</u> é adjacente ao vértice b;**
  - **o vértice b é adjacente ao vértice a.**

Matriz de adjacências

- *Espaço: $O(n^2)$*
- *Espaço pode ser reduzido agrupando-se as informações em bits ou armazenando apenas a matriz triangular.*
- *Tempo constante para testar a pertinência de uma aresta ao grafo.*

Matriz de adjacências

- cada elemento da matriz é a quantidade de arestas que vão do vértice i ao vértice j e vice-versa (são adjacentes).



simétrica

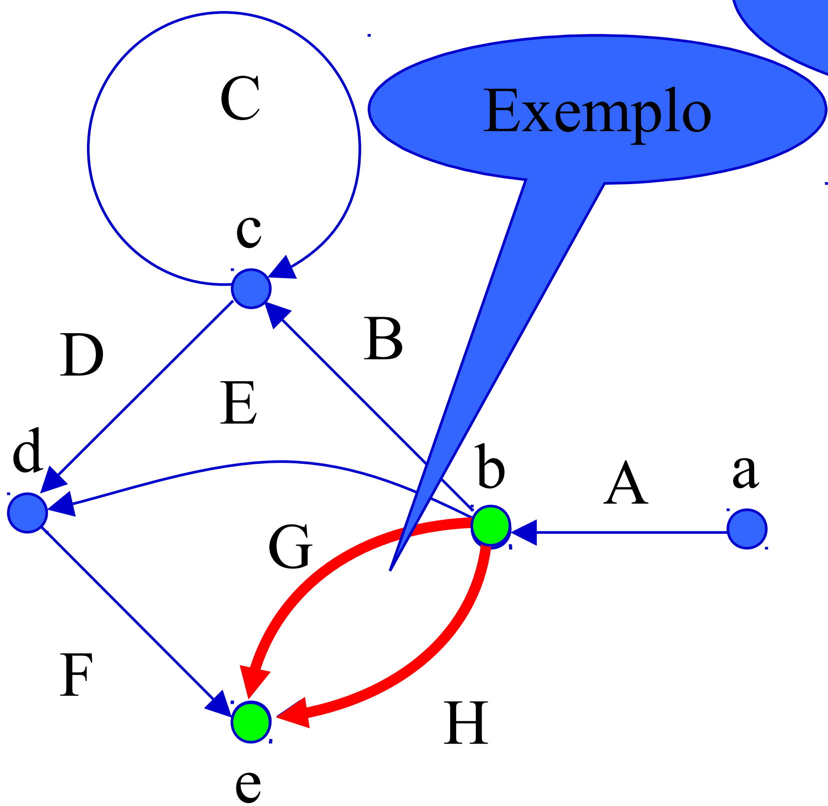
j

i →

	a	b	c	d	e
a	0	1	0	0	0
b	1	0	1	1	2
c	0	1	1	1	0
d	0	1	1	0	1
e	0	2	0	1	0

Matriz de adjacências

- cada elemento da matriz é a quantidade de arestas que vão do vértice i ao vértice j (o j é adjacente ao i).



18/25 = 72% de nulos

ao j ↓

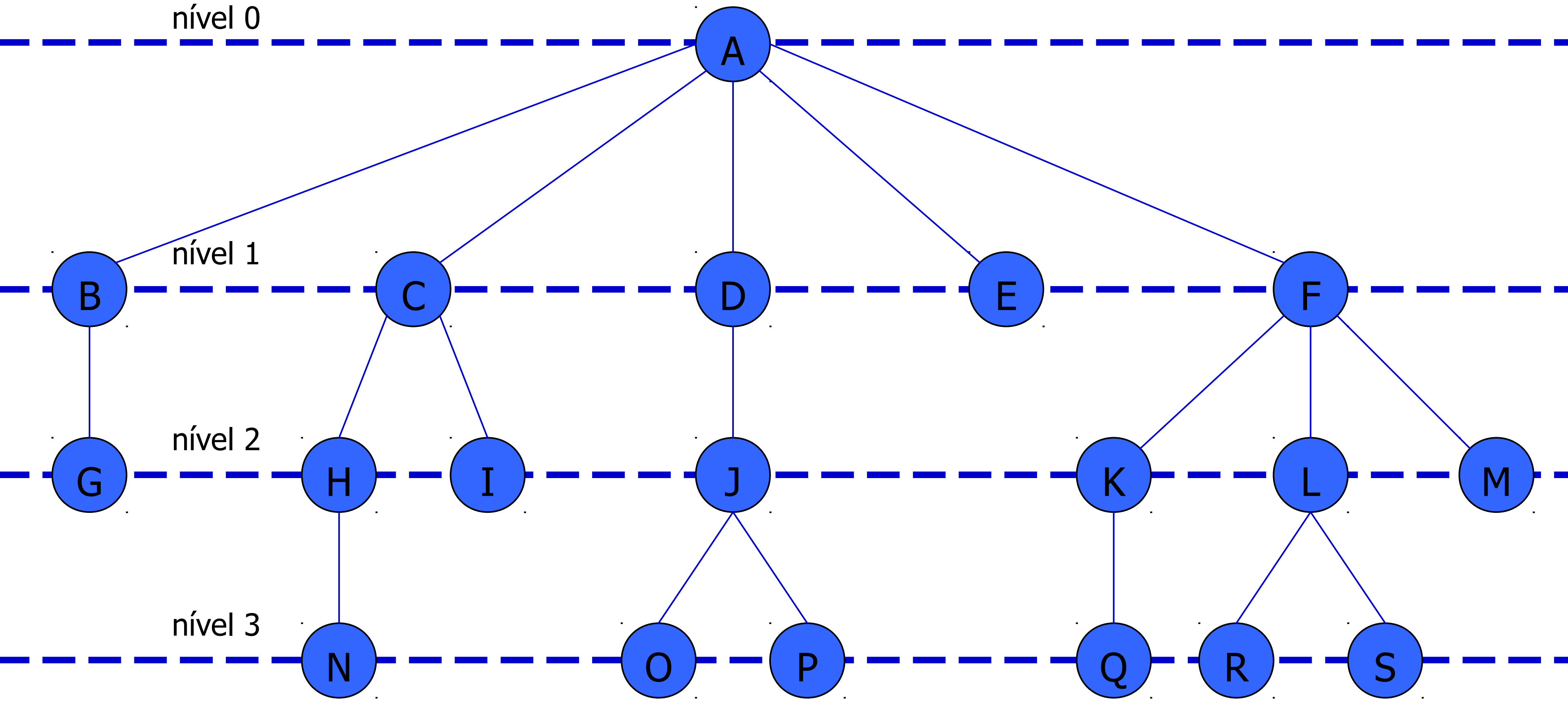
do i →

	a	b	c	d	e
a	0	1	0	0	0
b	0	0	1	1	2
c	0	0	1	1	0
d	0	0	0	0	1
e	0	0	0	0	0

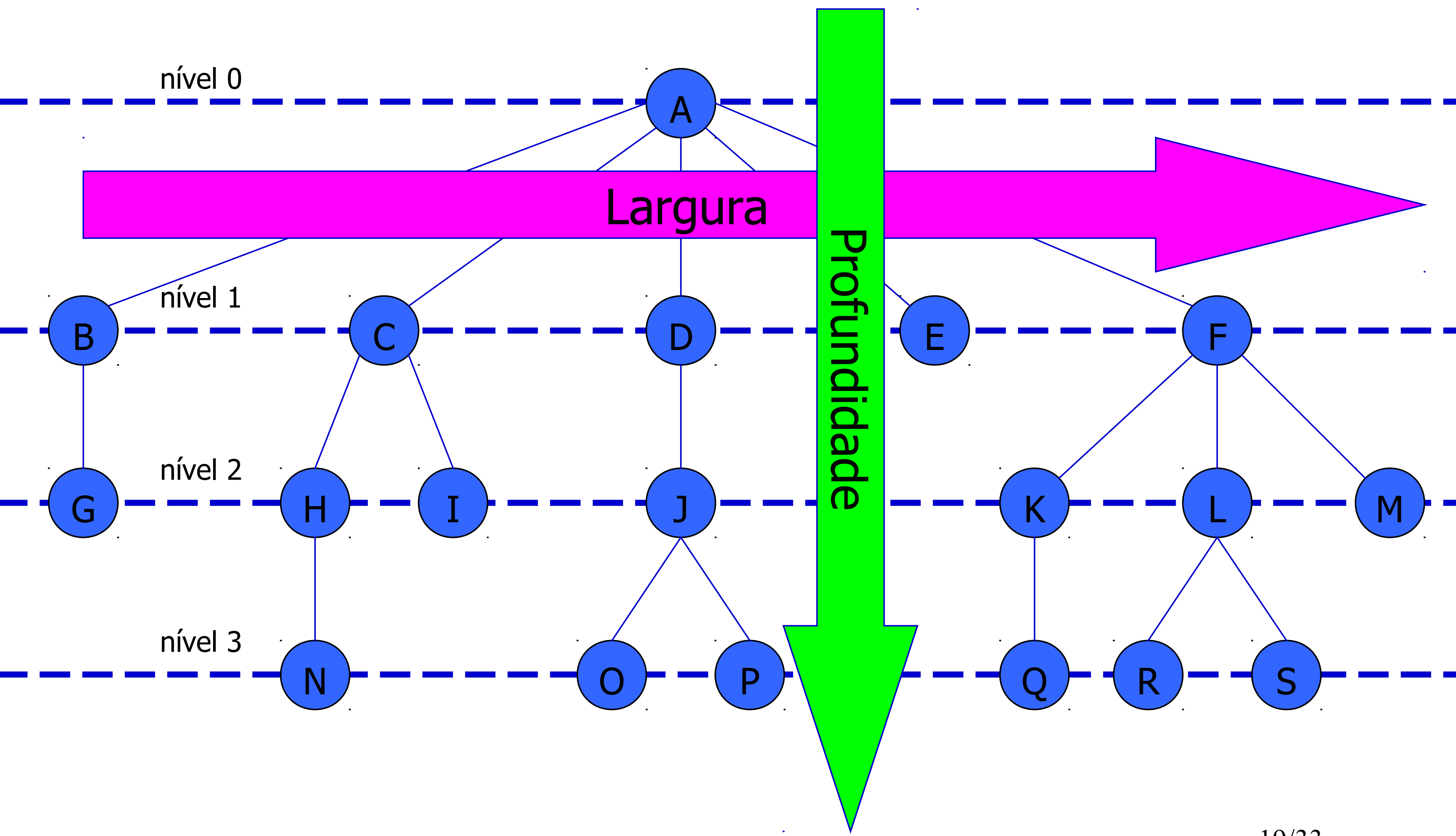
Matriz de adjacências X Lista de adjacências

Testar se aresta está no grafo.	**Matriz de Adjacência**
Determinar o grau de um vértice.	**Listas de Adjacências**
Menos memória em grafos pequen	**Listas de Adjacências**
Menos memória em grafos grande	**Matriz de Adjacência**
Inserção ou remoção de aresta.	**Matriz de Adjacência**
Atravessar o grafo.	**Listas de Adjacências**
Melhor na maioria dos problemas.	**Listas de Adjacências**

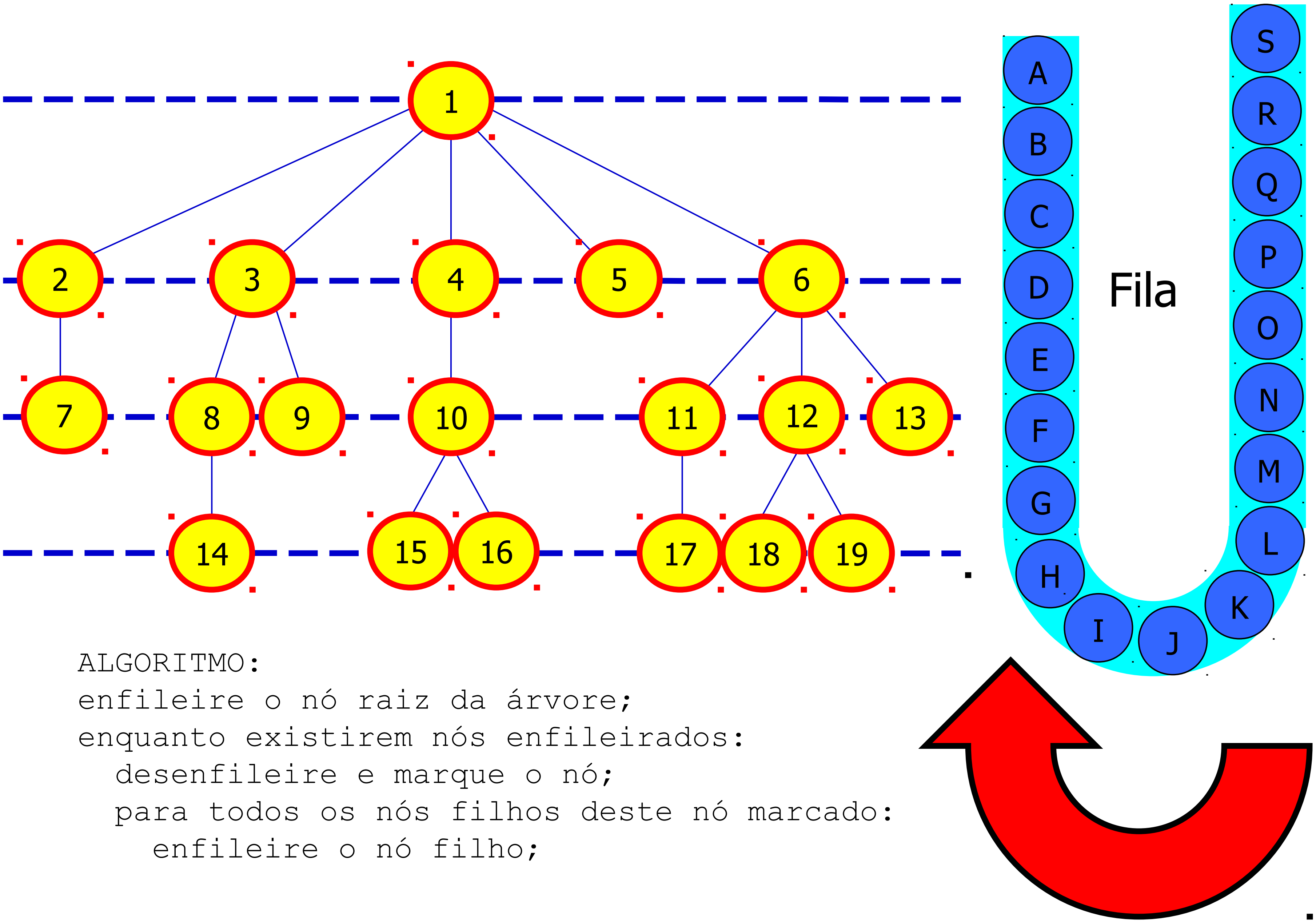
Árvore de busca



Árvore: busca em largura e busca em profundidade

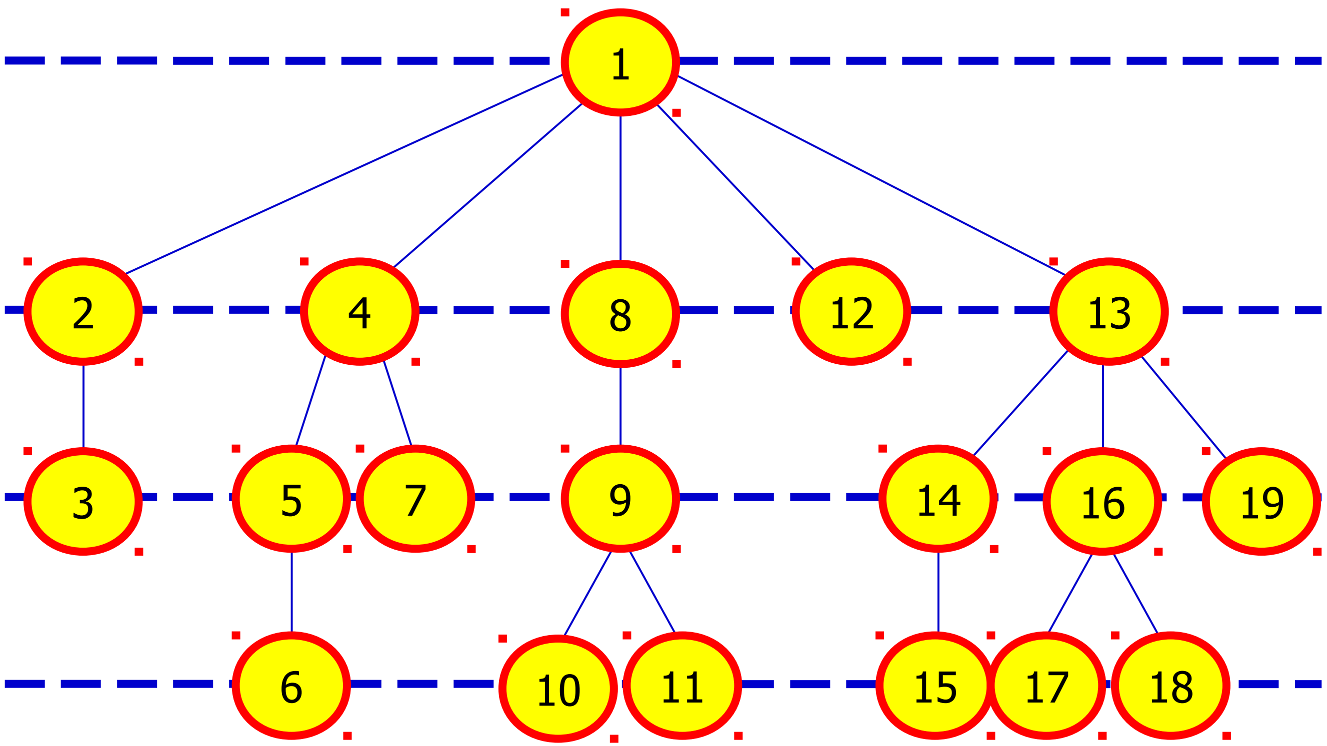


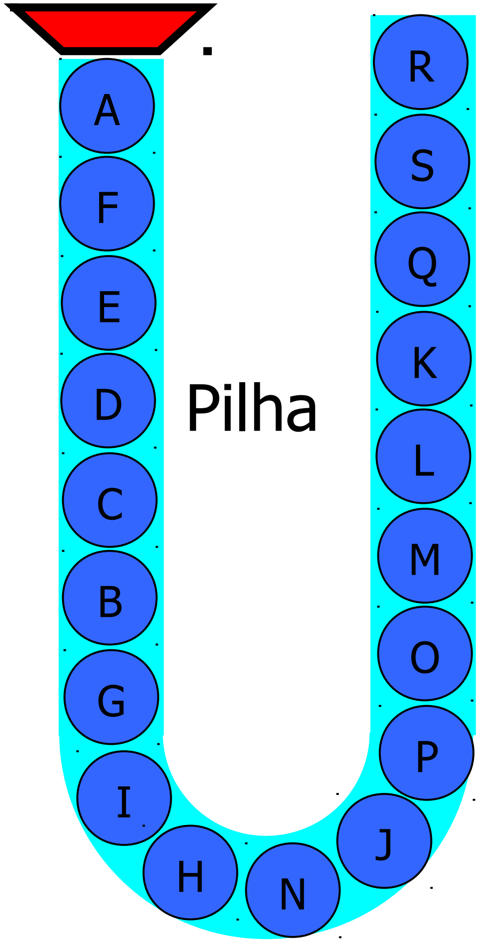
Busca em largura




```
ALGORITMO:
enfileire o nó raiz da árvore;
enquanto existirem nós enfileirados:
  desenfileire e marque o nó;
  para todos os nós filhos deste nó marcado:
    enfileire o nó filho;
```

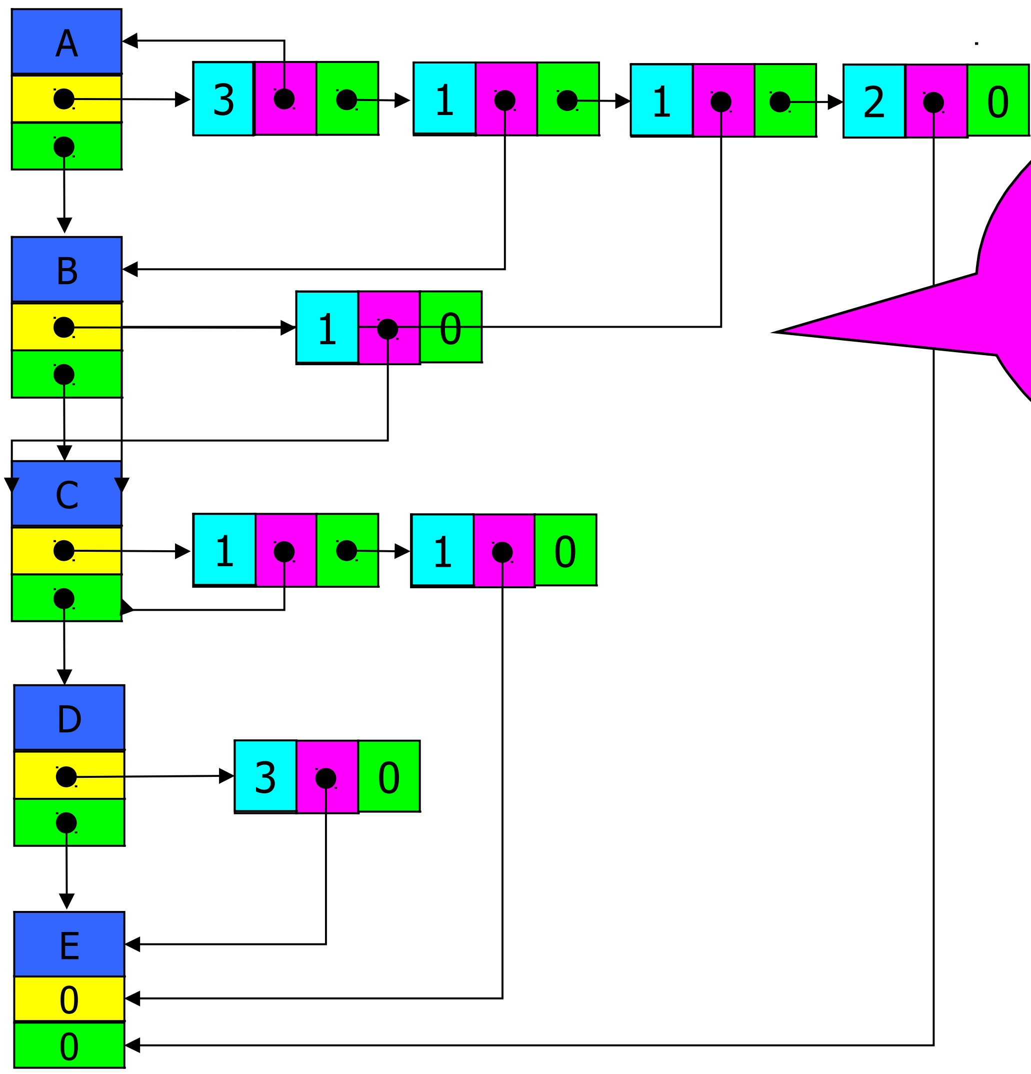
Busca em profundidade



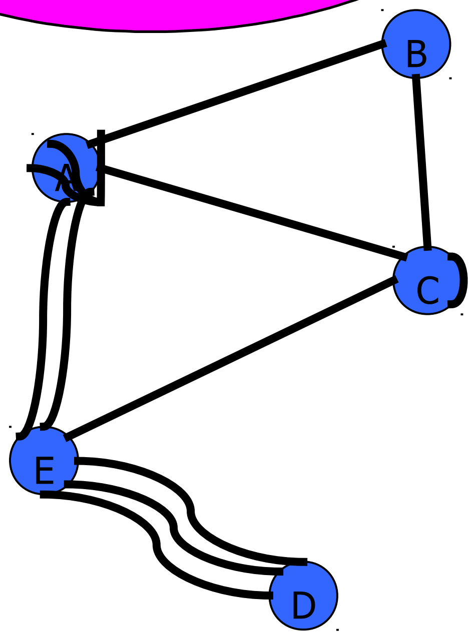



```
ALGORITMO:
empilhe o nó raiz da árvore;
enquanto existirem nós empilhados:
  desempilhe e marque o nó;
  para todos os nós filhos deste nó marcado:
    empilhe o nó filho;
```

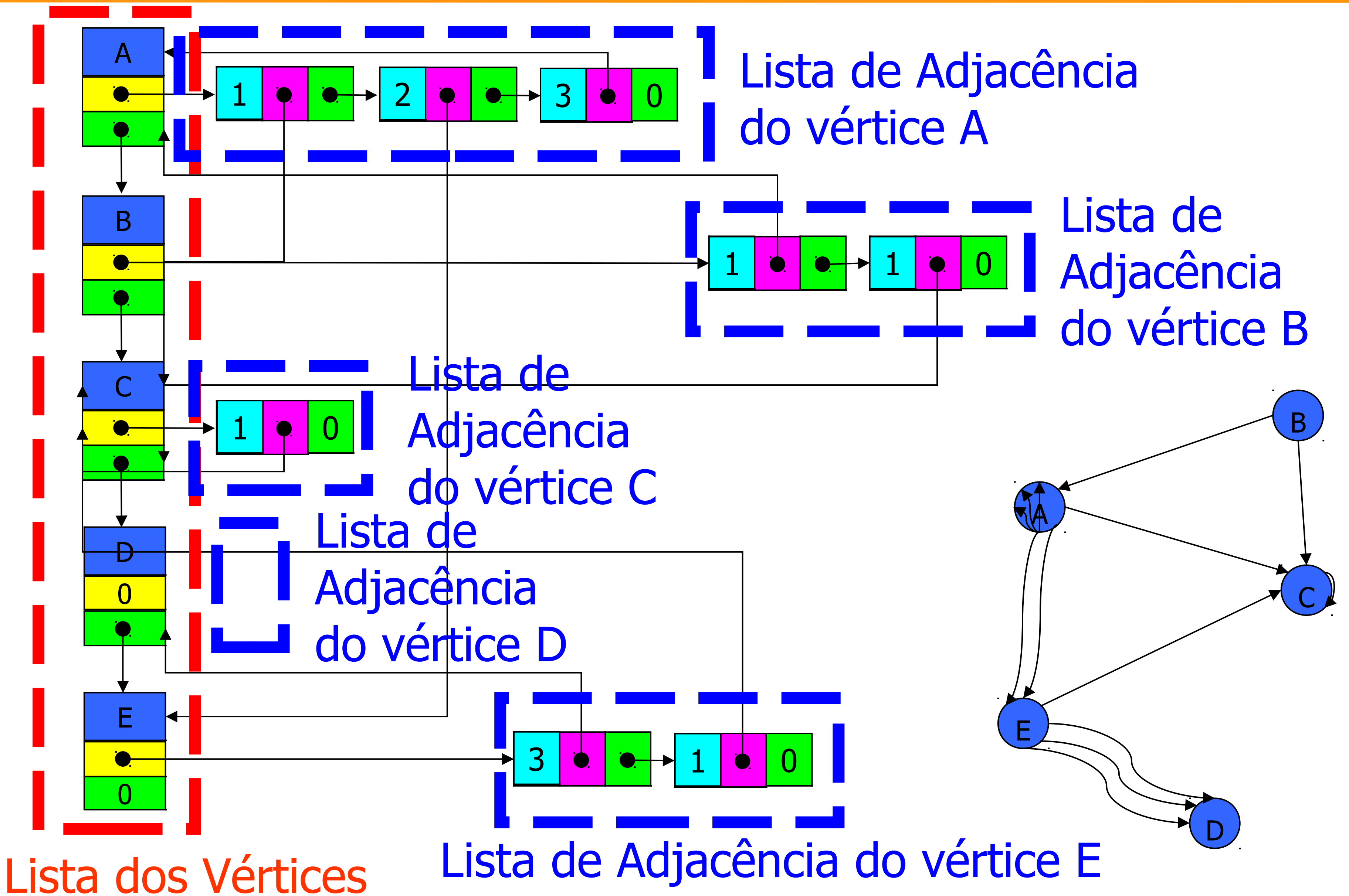
Grafos e Listas de Adjacência



A lista de adjacência está em ordem crescente de vértices (do A até o E) de forma a facilitar a implementação.



Grafos e Listas de Adjacência



Aplicações da Travessia

- *Componentes Conexas*
- *Detecção de Árvores e Ciclos*
- *2-coloração de Grafos*
- *Ordenação Topológica*
- *Vértices de Articulação*

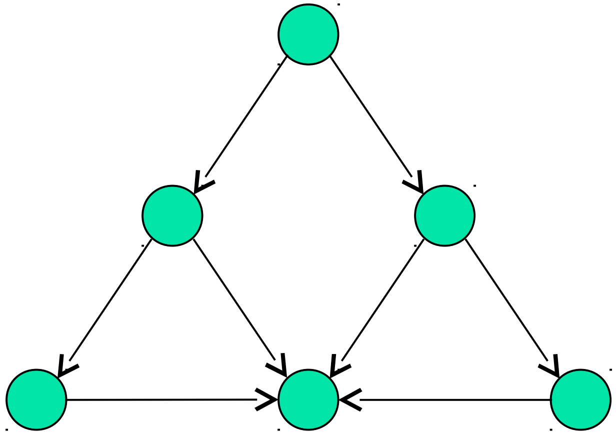
Componentes Conexas

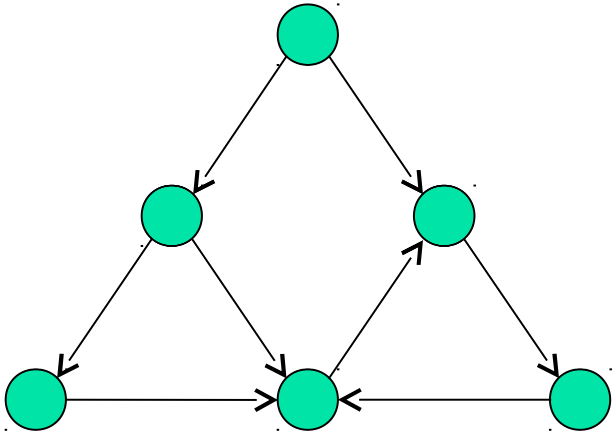
- *Pode-se usar Busca em Largura ou Busca em Profundidade.*
- *Em grafos orientados temos os conceitos de fracamente conexo e fortemente conexo.*
- *Tempo: O(m+n)*

Detecção de Árvores e Ciclos

- *Detecção de árvores é feita usando busca em profundidade $\Rightarrow$ o grafo é uma árvore se, e somente se, não existirem arestas de retorno.*
- *A detecção de ciclo é feita quando a primeira aresta de retorno é identificada.*
- *Tempo: O(n)*

- *Consideremos grafos dirigidos acíclicos.*





Ordenação Topológica

- *É uma ordenação nos vértices de forma que todas as arestas vão da esquerda para a direita.*
- *Busca em profundidade é utilizada para determinar se o grafo orientado é acíclico e então determinar uma ordenação topológica.*

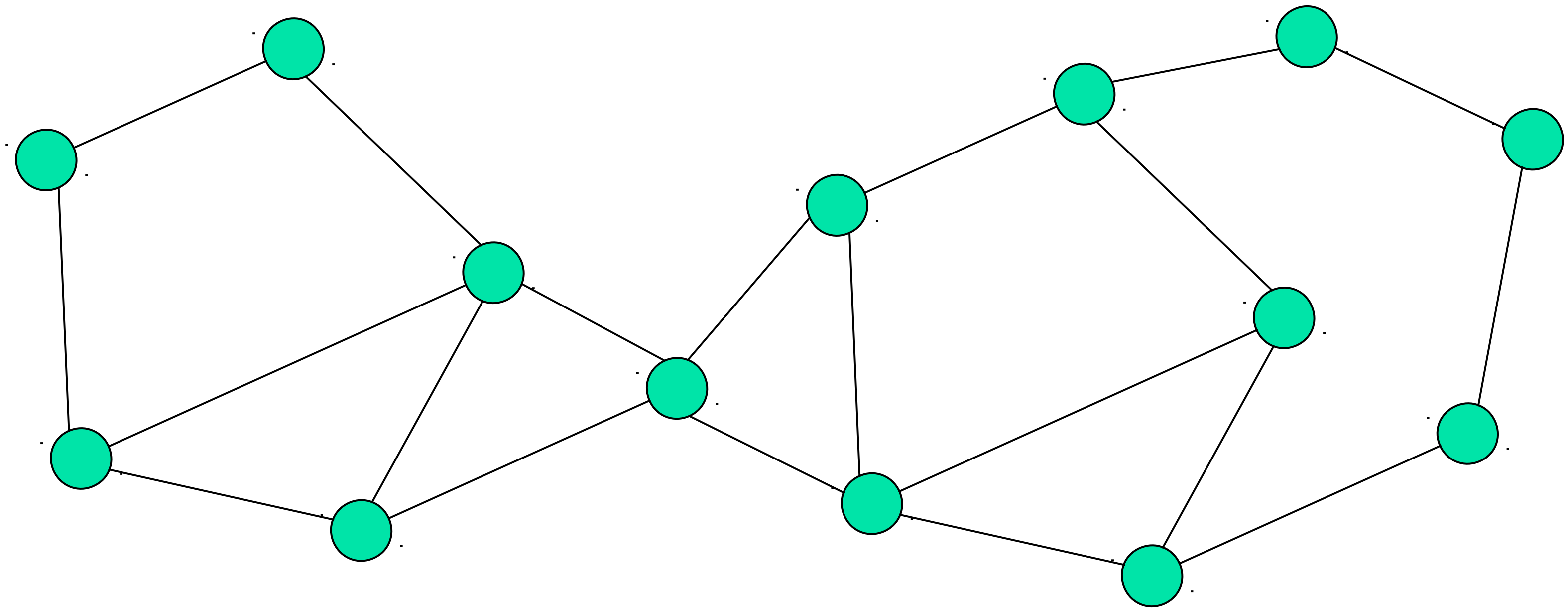
Ordenação Topológica

- *Um grafo orientado é acíclico se, e somente se, não são encontradas arestas de retorno durante uma busca em profundidade.*
- *Cada vértice é rotulado em ordem inversa à ordem em que eles são marcados completamente explorados.*

Vértices de Articulação

- *É um vértice de um grafo conexo cuja remoção torna o grafo desconexo.*
- *Qualquer grafo que contenha um vértice de articulação é frágil.*
- *A conectividade de um grafo é a quantidade de vértices necessários para desconectar o grafo.*
- *Conectividade é uma importante medida de robustez no projeto de rede.*

Vértices de Articulação



Modelagem de Problemas

- *Procura-se um algoritmo para o projeto de rotas naturais para personagens de jogos de vídeo game através de uma sala cheia de objetos. Como fazer?*

Como proceder ?

- *Uma grade onde cada vértice é um lugar válido e uma aresta ligando vértices vizinhos, com pesos dependendo da distância entre os vértices.*
- *Determinar o caminho mais curto.*

Modelagem de Problemas

- *No sequenciamento de DNA, os dados experimentais consistem de pequenos fragmentos.*
- *Para cada fragmento f, existem fragmentos que são forçados a estar à esquerda de f, outros à direita e outros que podem ir para qualquer um dos lados.*
- *Como podemos determinar uma ordenação consistente dos fragmentos da esquerda para a direita que satisfaça todas as restrições?*

Como proceder ?

- *Criar um grafo dirigido no qual a cada vértice é atribuído um fragmento.*
- *Uma aresta (l,f) liga um fragmento l que é obrigado a estar à esquerda.*
- *Uma aresta (f,r) liga um fragmento r que é obrigado a estar à direita.*
- *Determinar uma ordenação topológica neste grafo.*

Modelagem de Problemas

- *Dado um conjunto qualquer de retângulos no plano, como distribuí-los em um número mínimo de cestos tal que o subconjunto de retângulos em um mesmo cesto não intersectam-se entre si?*

Como proceder ?

- *Cada vértice representa um retângulo.*
- *Uma aresta liga vértices cujos retângulos se intersectam.*
- *Cada cesto corresponde a um conjunto independente.*
- *A coloração de vértices particiona um grafo em conjuntos independentes, logo nosso problema é determinar uma coloração mínima de vértices.*

- *Ao portar códigos de UNIX para DOS, tem-se que reduzir o tamanho dos nomes de várias centenas de arquivos para no máximo 8 caracteres cada. Como diminuir o nome dos arquivos e garantir que não haja conflito?*

Como proceder ?

- *Construir um grafo com vértices correspondendo ao nome original ligados a vértices que possam corresponder a reduções do nome.*
- *Determinar um conjunto de n arestas que não tenham vértices em comum.*
- *Determinar um conjunto independente de arestas em um grafo bipartido.*

Árvore Geradora Mínima

- *Uma árvore é uma grafo sem ciclos.*
- *Um árvore geradora é um subgrafo de G com os mesmo vértices que é uma árvore.*
- *Um árvore geradora mínima de um grafo com pesos é a árvore geradora de peso mínimo.*

Observações

- *Úteis em determinar a menor quantidade de fios necessárias para interligar um grupo de casas ou cidades.*
- *Pode existir mais de uma tal árvore.*
- *Para o caso em que todas as arestas têm o mesmo peso, todas as árvores geradoras são mínimas.*
- *Para o caso geral, o problema não é tão simples.*

Algoritmo de Prim

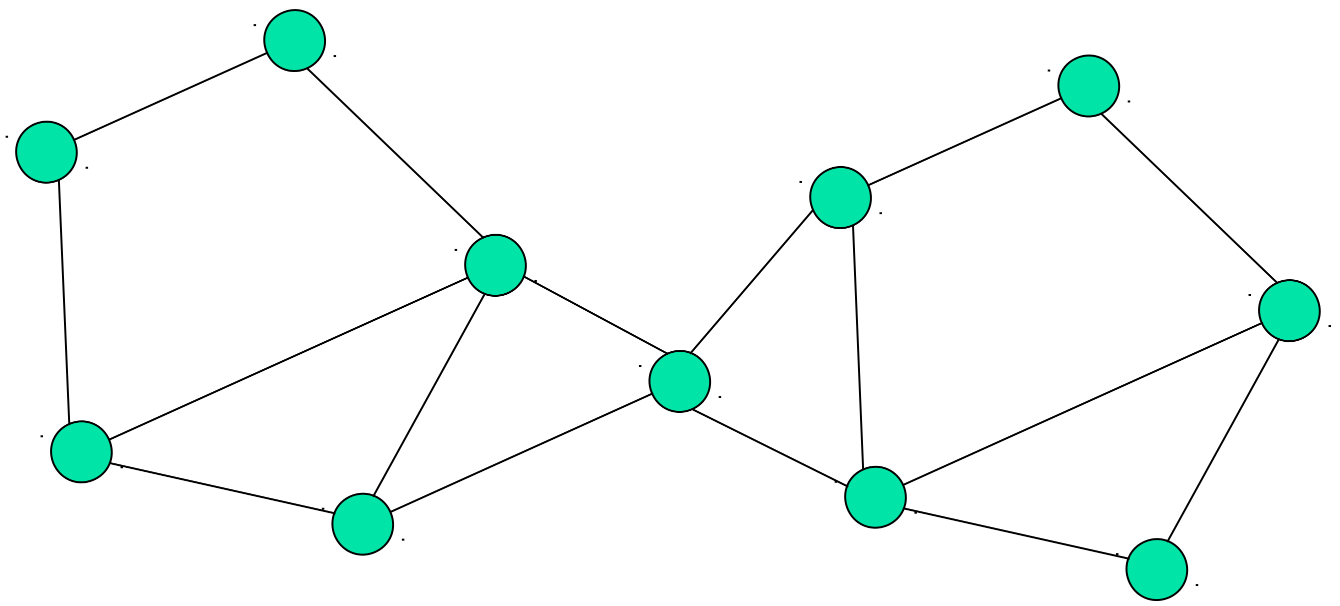
Escolha um vértice arbitrário s

enquanto existe algum vértice fora da árvore faça

tome a aresta de menor peso entre a árvore e um vértice não pertencen- te à árvore

adicione a aresta selecionada e o vértice à árvore T

Exemplo



Tempo

- *$O(nm)$*
- *Pode ser reduzido a $O(n^2)$.*

Algoritmo de Kruskal

Mantenha as arestas em uma fila de prioridade ordenada pelo peso

Conta <- 0

enquanto (Conta < n-1) faça

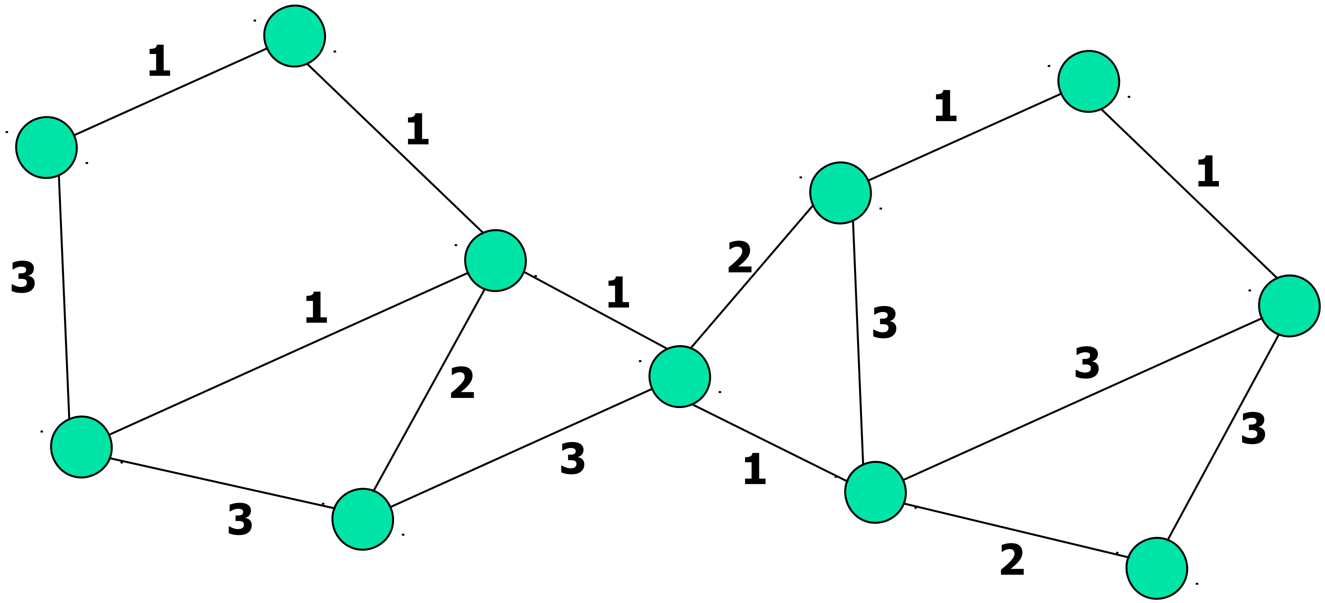
tome o próxima aresta (v,w)

se componente(v) != componente(w)

insira (v,w) em T

junte as componentes

Exemplo



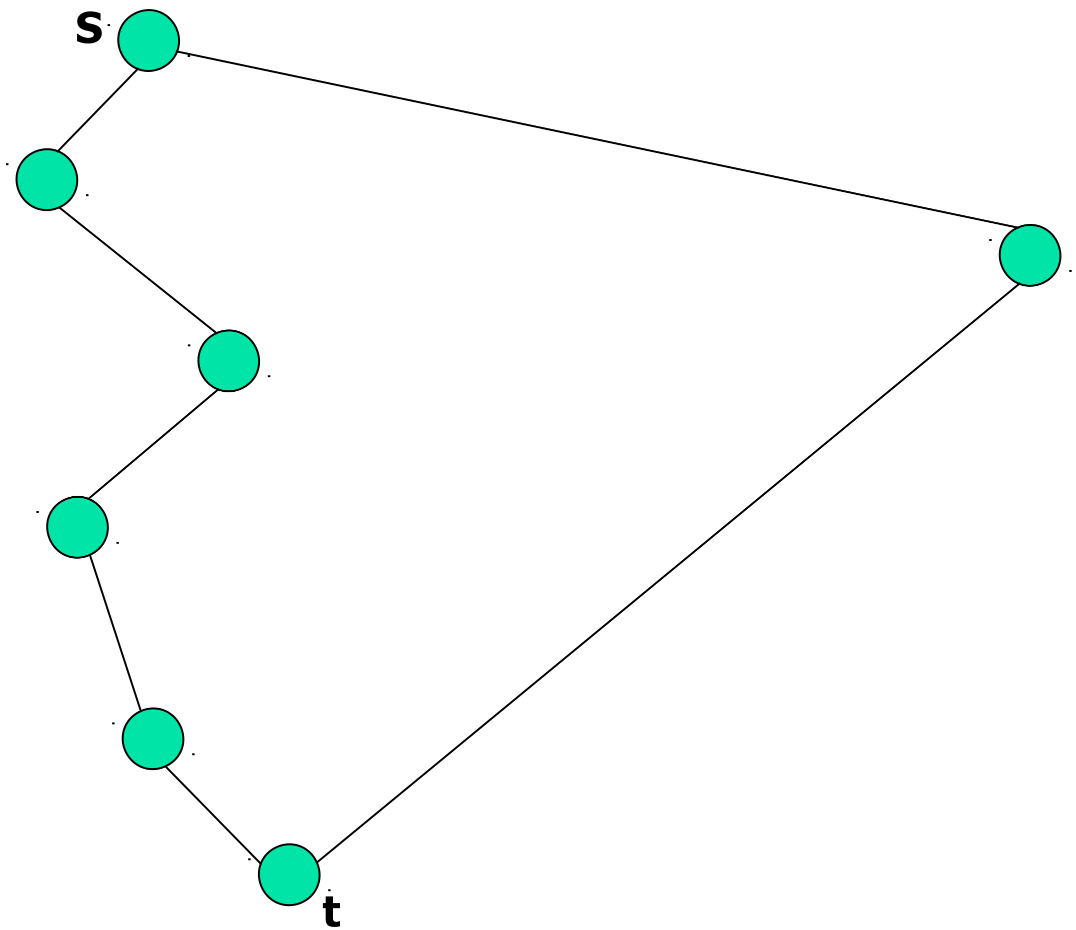
Tempo

- *O(nm)*
- *Pode ser melhorado para O(m log m).*

Caminho mais curto

- *Quando o grafo é sem pesos, a determinação de um caminho mais curto pode ser feita através de uma busca em largura.*
- *Quando o grafo tem pesos associados às arestas, o caminho mais curto pode não ser o que usa menos arestas.*

Caminho mais curto



Algoritmo de Dijkstra

- *O grafo de entrada tem pesos não-negativos associados às suas arestas.*
- *Dado o caminho mais curto entre s e cada vértice $v_1, v_2, \ldots v_k$ de um conjunto de vértices, deve existir algum outro vértice x tal que o caminho mais curto de s a v_i a x, para algum $1 \leq i \leq k$.*