Árvores

Emílio Bergamim Júnior

Instituto de Geociências e Ciências Exatas - UNESP

2024

Introdução

- Como já discutidos anteriormente, a conexidade de um grafo é uma propriedade extremamente interessante.
- Obviamente, quanto maior o número de arestas em um grafo, maior a probabilidade deste ser conexo. Obviamente, um grafo completo é conexo.
- No entanto, armazenar grafos completos pode ser extremamente custoso computacionalmente.
- Qual o menor número de arestas (ou arcos, no caso direcionado) possível para tornar um grafo de n vértices conexo?

Problemas de interligação

- Suponha o seguinte: deseja-se interligar um conjunto de casas com uma rede elétrica, de forma a fornecer energia para todas.
- Há um problema de custo intrínseco: pode-se conectar todas casas, mas isso gera um custo adicional de realizar essas conexões.
- Portanto, existe um problema de como construir um grafo conexo de forma a minimizar o número de ligações.

Árvore

Uma árvore é um grafo não-orientado conexo sem ciclos.
Uma aresta é dita uma ponte se sua remoção torna um grafo desconexo.
São equivalentes as três afirmações para um grafo não-orientado T de ordem n:

- T é uma árvore.
- T é conexo e possui $n - 1$ arestas.
- Cada aresta de T é uma ponte

Isso nos dá uma noção do menor tamanho possível para que um grafo seja conexo.

Ciclos como redundâncias

- Além do tamanho do grafo, há uma outra propriedade interessante em uma árvore: esta é um grafo **acíclico**.
- Repare que, se seu grafo é conexo e possui um ciclo, remover uma única aresta do ciclo pode desmanchar o mesmo, mas mantém a propriedade de conexidade válida.
- Dessa forma, podemos ver que uma árvore é realmente minimal quanto à conexidade.
- Em termos de atingibilidade, pode-se identificar um ciclo como a existência de uma redundância: ainda que perca-se uma aresta, esta propriedade será mantida.

Ciclos como redundâncias

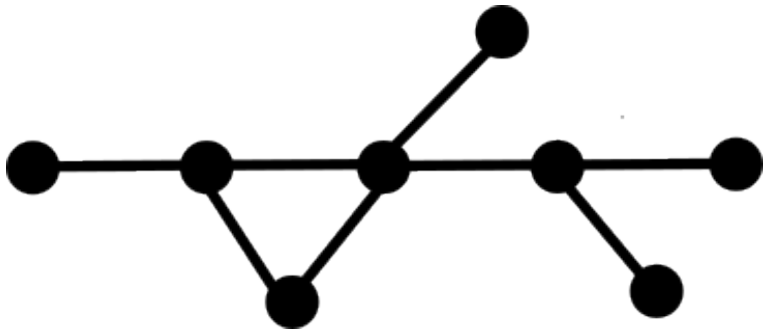

Figura: Exemplo de um grafo conexo com um ciclo.

Ciclos como redundâncias

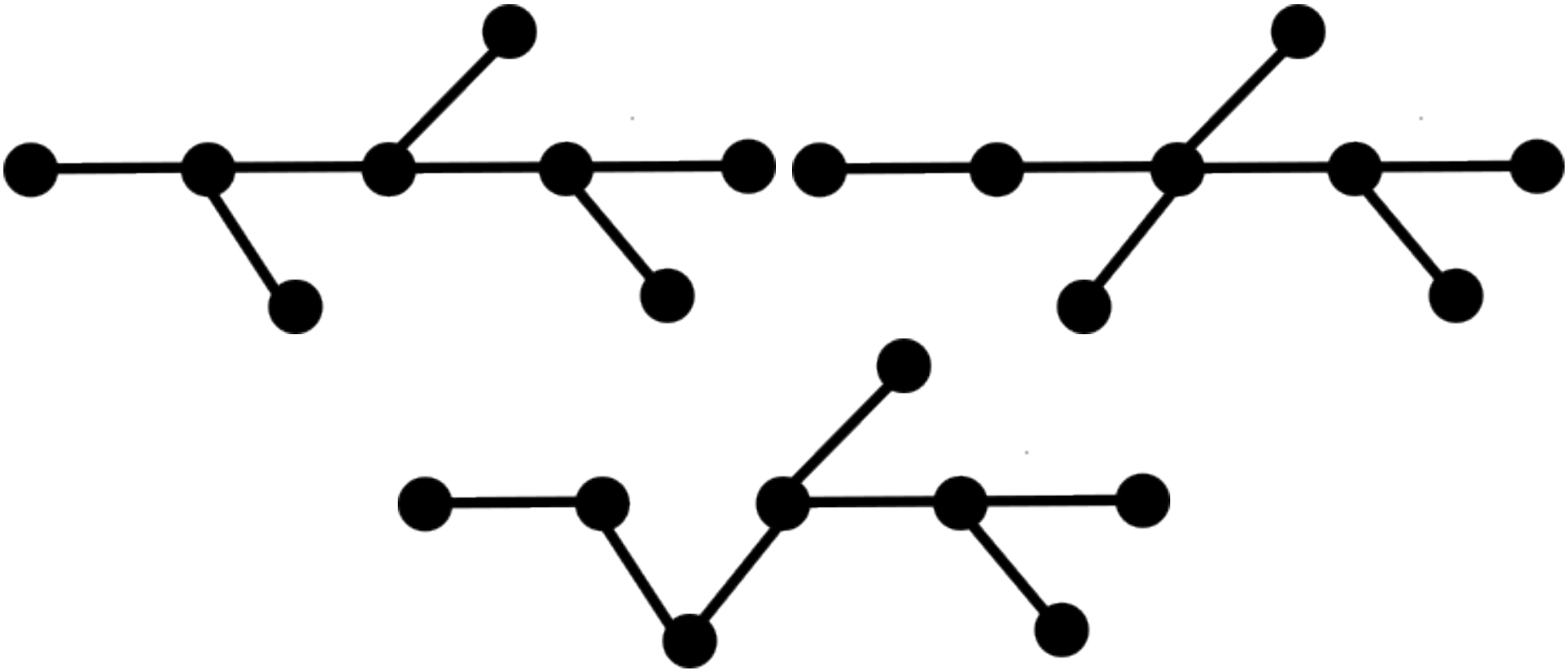

Figura: Exemplos de como quebrar um ciclo pode manter a propriedade de conexidade.

Conectividade, tamanho e distâncias

- Quando aumenta-se o número de arestas em um grafo conexo, esta propriedade obviamente não é alterada.
- No entanto, distâncias podem ser encurtadas.
- Veja que a escassez de arestas em árvores permite um armazenamento $O(n)$ da estrutura, porém as distâncias entre vértices podem se tornar bastante elevadas.

Conectividade, tamanho e distâncias

- A famosa teoria dos seis graus de separação estipula que qualquer ser humano está separado de outro por, no máximo, seis relações.
- Veja que, dado o tamanho da população humana, se as relações entre pessoas formassem uma árvore, essa distância podia ser na casa de bilhões.
- No entanto, o fato de que a densidade de relações reais é significativamente maior do que em uma árvore permite uma forma de comunicação mais eficiente.

Árvores direcionadas?

- O conceito de árvore discutido até aqui é aplicável somente a grafos não-orientados.
- Como pode-se estender esse conceito para grafos orientados?

Arborescências

- Como cada aresta em uma árvore é uma ponte, pode-se escolher um nó raiz.
- As arestas que conectam uma raiz a seus vizinhos são então orientadas, partindo da raiz para os vizinhos. Isto é, os vizinhos da raiz serão seus sucessores.
- As arestas dos vizinhos de vizinhos são então transformados em sucessores dos primeiros. E o procedimento é repetido até chegar a nós que não possuem vizinhos.
- Nesta definição, note que a raiz é o único nó que não possui nenhum sucessor. Veja também que qualquer nó pode ser tomado como raiz.

Arborescências

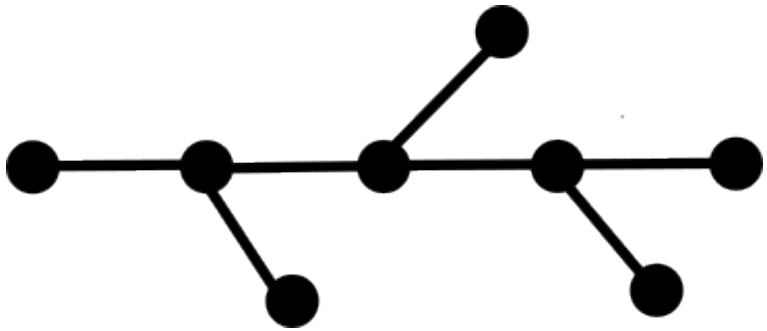

Figura: Considere uma árvore como esta. Escolhendo diferentes nós como raízes, diferentes arborescências são possíveis.

Arborescências

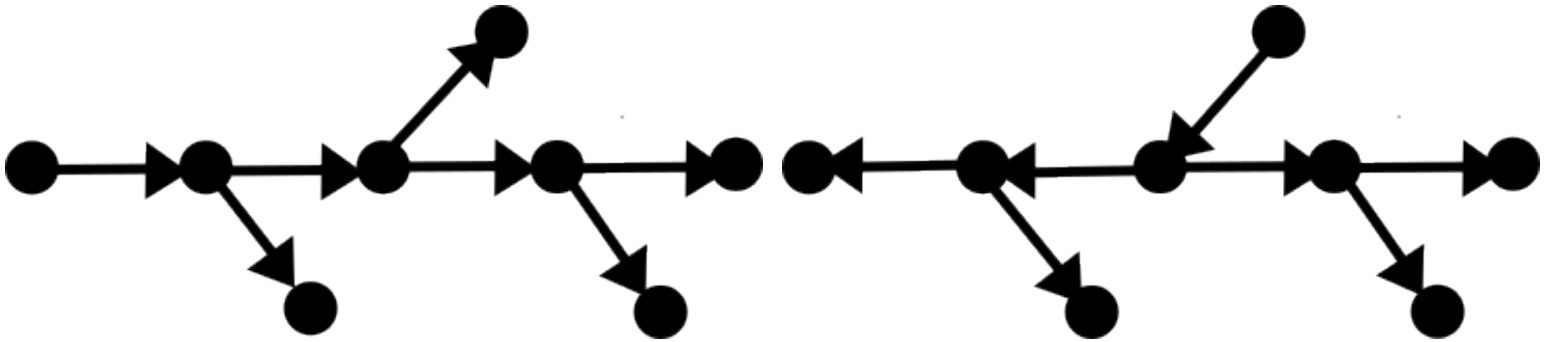

Figura: Exemplos de arborescências que podem ser feitas a partir da árvore anterior.

Arborescências e ordenação

- Ao escolher uma raiz, cria-se um ordem intrínseca para a arborescência.
- Cada nó pode então ser ordenado de acordo com sua distância da origem: a raiz é a primeira, os sucessores desta vem em sequências, depois os sucessores de sucessores e assim por diante.
- Exemplos clássicos de arborescências são as árvores binárias, que possuem diversas aplicações.
    - Uma aplicação clássica é o algoritmo Heapsort, que utiliza desse tipo de estrutura para ordenar uma lista.

Caminhos mínimos em arborescências

- Note que, ciente dos números de vizinhos de cada nó e da ordenação imposta pela arborescência, o cálculo de caminhos mínimos é consideravelmente facilitado.
- Com o adicional da orientação das arestas, sabe-se que só existem distâncias finitas entre nós situados em um mesmo ramo da árvore. Isso significa que um dos nós deve ser atingível a partir do outro.
- Assim, se o nó u está a uma distância $h_{0,u}$ da raiz e v está a uma distância $h_{0,v}$ da mesma e v é atingível a partir de u, temos que

$$h_{u,v} = h_{0,v} - h_{0,u}. \tag{1}$$

Ordenação topológica

- Qualquer nó pode ser tomado como raiz
- Uma vez escolhida uma raiz, está posta uma ordenação de acordo com as distâncias da raiz
- Assim, partindo do nó raiz e então prosseguindo para seus vizinhos, os vizinhos destes e assim por diante, chega-se a uma ordenação na qual o sucessor de um nó em uma arborescência está situado sempre depois deste nesta ordenação.
- Uma ordenação deste tipo é uma ordenação topológica
  - Diferentes ordenações topológicas existem para uma mesma raiz. A obtida pela busca em profundidade é apenas uma delas.

Busca em profundidade

BuscaP(Árvore)

1. Visitado = $\emptyset$
2. Para cada nó em Grafo:
   - Se nó não está em Visitado:
     - Visita(nó,NULL,Visitado)
3. Retorna Visitado

Visita(nó,pai,Visitado)

1. Para cada vizinho de nó diferente de pai:
   - Visita(vizinho,nó,Visitado)
2. Insere nó como primeiro elemento de Visitado.

Ao final, os elementos de Visitado estarão em uma ordenação topológica.

Busca em profundidade

- Em essência, Visitado é uma pilha: o elemento inserido por último está sempre na primeira posição.
- Alterar a ordem de inserção também produz um algoritmo de ordenação topológica: basta fazer a inserção antes de cada chamada de Visita
- A necessidade de identificar o pai de um nó na arborescência é uma necessidade em grafos não-orientados. No caso orientado, isso é dispensado.
- Note, no entanto, que o algoritmo entra numa recursão infinita caso o grafo contenha algum ciclo.

Busca em profundidade

- Dizemos que um nível da árvore é um conjunto de nós situados a uma mesma distância da raiz. O número de níveis de uma árvore é dita sua profundidade.
- Sendo b o número médio de sucessores para os nós de uma árvore e d sua profundidade, a complexidade da busca em profundidade é $O(bd)$.

Busca em largura

BuscaL(Grafo,raiz)

1. Inicializa duas filas: F (fronteira) e E (explorados) e adicionada raiz a ambas
2. Enquanto F não estiver fazia,
    1. p recebe o primeiro elemento de F, que é removido de F.
    2. Para cada vizinho de p que não está em E,
        - Insere vizinho em F e em E

Ao final, os elementos de E estarão em uma ordenação topológica. (Elementos inseridos na fila são colocados na última posição)

Busca em largura

- A busca em largura ordena a árvore por níveis, de acordo com sua distância da raiz.
- Primeiro são inseridos os vizinhos da raiz, depois seus vizinhos e depois os vizinhos destes e assim por diante.
- No caso orientado, não há necessidade de checar se um vizinho está em E, pois essa noção é substituída pela de sucessor
- A complexidade desse algoritmo também é $O(bd)$.

Exercício

Implemente a busca em largura para ordenação topológica.

Árvores de Bienaymé

Árvores podem ser usadas para modelar relações familiares. Cada nó corresponde a um indivíduo e a existência de uma aresta denota uma relação de parentesco. Assim, em uma ordenação topológica adequada, os sucessores serão filhos de seus antecessores.
Bienaymé e, posteriormente, Galton e Watson debruçaram-se sobre o problema de árvores genealógicas nos quais o número de filhos de um nó segue uma distribuição de probabilidade.

- Um exemplo razoavelmente simples é o de uma distribuição binomial com parâmetros $n_{max} \in \mathbb{N}$ e $p \in (0, 1)$. De forma que, para cada nó são gerados um número de vizinhos com a dita distribuição até que chegue-se em um nível no qual não são gerados filhos ou atinja-se uma profundidade limite d_{max}.

Trabalho

- Aplique a busca em largura para todos os nós de uma árvore binomial e calcule a média das profundidades das ordenações geradas.
- Faça experimentos variando n_{max}, p e d_{max}. Para cada tripla desses valores, gere pelo menos 10 árvores aleatórias de forma a tomar a média entre estas. Apresente seus resultados na forma de tabelas ou gráficos.
- Esse modelo tem uma conhecida transição de fase na qual, para $n_{max}p < 1$ existe algum d_{max} para o qual um nível sem filhos existe. Caso $n_{max}p > 1$ o algoritmo será parado somente por atingir a profundidade limite.
- Você pode ler mais sobre esse assunto aqui.