Árvore geradora mínima

Algoritmos de Prim, Kruskal e Delete-Reverse

Prof. Marcelo de Souza

marcelo.desouza@udesc.br

Material de consulta

Leitura obrigatória:

- Capítulo 3 de Goldbarg and Goldbarg (2012) – Árvores.
- Capítulo 4 de Kleinberg and Tardos (2006) – Algoritmos gulosos.

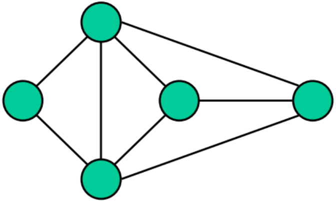
Leitura complementar:

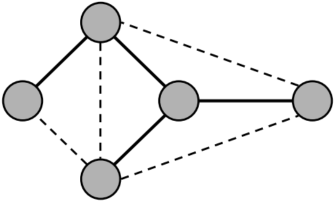
- Capítulo 14 de Goodrich et al. (2014) – Algoritmos em grafos.
- Capítulo 15 de Preiss (2001) – Grafos e algoritmos em grafos.

Árvore geradora

Conceitos

- Uma árvore geradora de um grafo $G = (V, E)$ é um subgrafo de G que é acíclico (árvore) e conexo.
  - Também chamada de *árvore de cobertura* ou *árvore de extensão*.

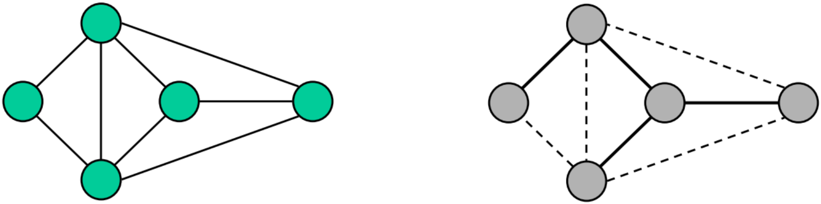




Árvore geradora

Conceitos

- Uma árvore geradora de um grafo $G = (V, E)$ é um subgrafo de G que é acíclico (árvore) e conexo.
  - Também chamada de *árvore de cobertura* ou *árvore de extensão*.

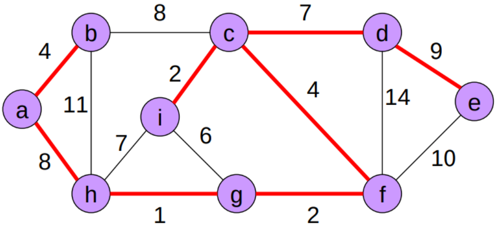


- Se T é uma árvore geradora de G, então:
  - T é acíclico e conectado.
  - T possui $n-1$ arestas.
  - Removendo uma aresta de T torna o grafo desconexo.
  - Inserindo uma aresta a T torna o grafo cíclico.

Árvore geradora mínima

Conceitos

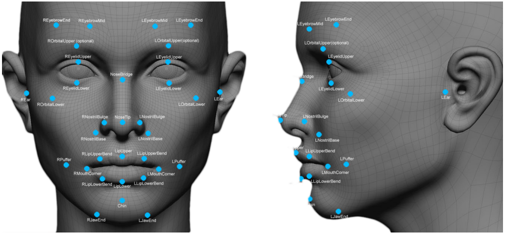
- Dado um grafo $G = (V, E)$ ponderado, uma árvore geradora mínima é uma árvore geradora cujo somatório dos pesos das arestas é mínimo.



Árvore geradora mínima

Aplicações

- Cenário 1: os pinos de uma placa de circuito impresso devem ser conectados com a menor quantidade de fio.
  - Pinos são os vértices e os fios são as arestas.
- Cenário 2: a universidade deseja construir passarelas para ligar os prédios do campus. Quais passarelas devem ser construídas para termos o menor gasto possível de recusros financeiros?
  - Prédios são os vértices e suas possíveis ligações são as arestas.
- Cenário 3: reconhecimento de faces em tempo real.



Árvore geradora mínima

Algoritmos

- **Problema:** dado um grafo conexo não-direcionado ponderado $G = (V, E)$, encontra uma árvore geradora mínima $H = (G, T)$.

Árvore geradora mínima

Algoritmos

- **Problema:** dado um grafo conexo não-direcionado ponderado $G = (V, E)$, encontra uma árvore geradora mínima $H = (G, T)$.

- Algoritmo de Prim: análogo ao algoritmo de Dijkstra, inicia de um vértice raiz s e cresce uma árvore a partir dele. A cada iteração insere o vértice que pode ser alcançado com menor custo a partir da árvore parcialmente construída.

Árvore geradora mínima

Algoritmos

- **Problema:** dado um grafo conexo não-direcionado ponderado $G = (V, E)$, encontra uma árvore geradora mínima $H = (G, T)$.
- Algoritmo de Prim: análogo ao algoritmo de Dijkstra, inicia de um vértice raiz s e cresce uma árvore a partir dele. A cada iteração insere o vértice que pode ser alcançado com menor custo a partir da árvore parcialmente construída.
- Algoritmo de Kruskal: inicia a árvore sem nenhuma aresta (somente os vértices). Iterativamente insere arestas de E, do menor ao maior custo, desde que sua inserção não forme um ciclo em H.

Árvore geradora mínima

Algoritmos

- **Problema:** dado um grafo conexo não-direcionado ponderado $G = (V, E)$, encontra uma árvore geradora mínima $H = (G, T)$.
- Algoritmo de Prim: análogo ao algoritmo de Dijkstra, inicia de um vértice raiz s e cresce uma árvore a partir dele. A cada iteração insere o vértice que pode ser alcançado com menor custo a partir da árvore parcialmente construída.
- Algoritmo de Kruskal: inicia a árvore sem nenhuma aresta (somente os vértices). Iterativamente insere arestas de E, do menor ao maior custo, desde que sua inserção não forme um ciclo em H.
- Algoritmo Reverse-Delete: é a versão reversa do algoritmo de Kruskal. Inicia com $H = V$ e, iterativamente, remove arestas do maior para o menor custo, desde que a remoção não desconecte o grafo.

Algoritmo de Prim

Pseudocódigo

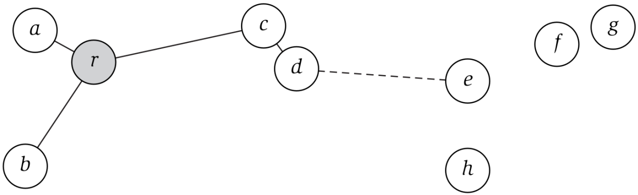
Algorithm: `prim(Vertex s)`

$Q \leftarrow V$
Set $d(v) = \infty$ and $p(v) = -1$ for each $v \in V$
$d(s) = 0$
while $Q \neq \emptyset$ **do**
    $u \leftarrow$ extract-min(Q)
    for *each* v *adjacent to* u **do**
        if $v \in Q$ *and* $d(v) > w(u, v)$ **then**
            $d(v) \leftarrow w(u, v)$
            $p(v) \leftarrow u$
            update(Q, v)

Algoritmo de Prim

Funcionamento

- Uma execução intermediária do algoritmo de Prim:



- r é o vértice raiz da busca.
- Arestas adicionadas: $\{r, a\}; \{r, b\}; \{r, c\}; \{c, d\}$.
- Algoritmo só adiciona arestas que farão parte da *AGM*.
  - Sempre escolhe a aresta de menor custo que liga a árvore já construída com algum vértice ainda desconectado.

Algoritmo de Prim

Complexidade

- A operação `extract-min` é executada n vezes.
- O laço interno totaliza m passos quando utilizada uma lista de adj.
  - `update` é executado m vezes no pior caso.
- O laço interno totaliza n^2 passos quando utilizada uma matriz de adj.
  - `update` é executado n^2 vezes no pior caso.

	matriz de adj.	**lista de adj.**
UnorderedPQ	$O(n^2)$	$O(n^2)$
OrderedPQ	$O(n^3)$	$O(n^2)$
BinaryHeap	$O(n^2 \log n)$	$O(m \log n)$

Exercício

Árvore geradora mínima

1. Crie um programa que leia a especificação de um grafo e calcule a árvore geradora mínima, utilizando um dos algoritmos discutidos em aula. Aplique seu programa para determinar a árvore geradora mínima entre as cidades da região do Alto Vale do Itajaí.

Referências

Goldbarg, M. and Goldbarg, E. (2012). *Grafos: Conceitos, algoritmos e aplicações*. Elsevier.

Goodrich, M. T., Tamassia, R., and Goldwasser, M. H. (2014). *Data structures and algorithms in Java*. John Wiley & Sons, 6th edition.

Kleinberg, J. and Tardos, É. (2006). *Algorithm Design*. Pearson Education India.

Preiss, B. R. (2001). *Estruturas de dados e algoritmos: padrões de projetos orientados a objetos com Java*. Campus.