Teoria dos Grafos
Unidade 6: Árvores

Prof. Dr. Paulo César Rodacki Gomes
paulo.rodacki@blumenau.ifc.edu.br

INSTITUTO FEDERAL
CATARINENSE
Câmpus Blumenau

Blibliografia

- Márcia A. Rabuske. **Introdução à Teoria dos Grafos**. Editora da UFSC. 1992.
- Joan M. Aldous, Robin J. Wilson. **Graphs and Applications**: as introductory approach. Springer. 2001
- Thomas Cormen et al. **Algoritmos**: teoria e prática. Ed. Campus. 2004.

Tópicos

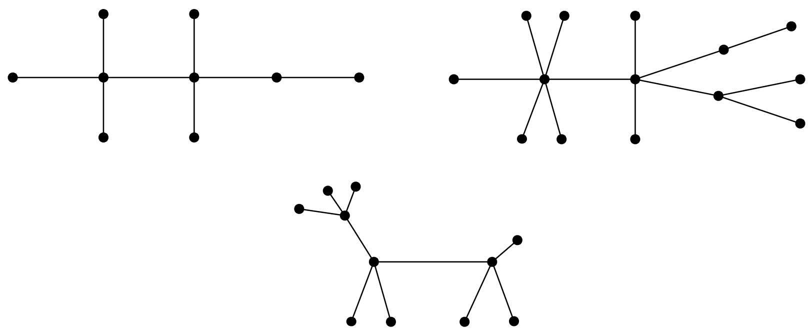
- Introdução/definições
- Problema da árvore geradora de custo mínimo
- Algoritmo de Kruskal
- Algoritmo de Prim

Definições

Uma árvore é:

1. um grafo conexo com n vértices e n-1 arestas;
2. um grafo conexo sem ciclos;
3. um grafo no qual cada par de vértices é ligado por um e somente um caminho simples;
4. um grafo conexo onde todas as arestas são "pontes";
5. um grafo acíclico conexo, porém se dois vértices não adjacentes forem ligados por uma aresta, então o grafo passará a ter exatamente um ciclo.

Árvores - exemplos



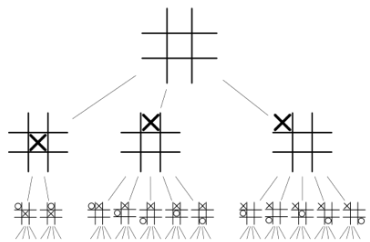
Exercícios

1. Existem oito árvores não rotuladas com cinco ou menos vértices. Desenhe todas elas.
2. Explique porque toda árvore também é um grafo bipartido. Faça isso colorindo os vértices alternadamente de preto e branco.
3. Explique porque uma árvore com n vértices possui n-1 arestas.

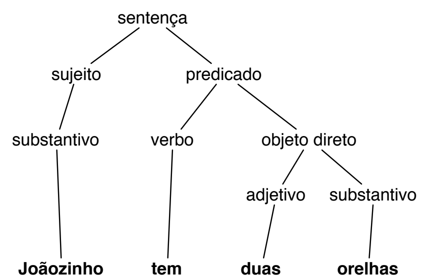
Definições

- Uma **floresta** é um conjunto de árvores;
- Uma **árvore dirigida** é um digrafo acíclico onde o grau de entrada de cada vértice é 1, exceto o da raiz, que possui grau de entrada zero;
- a **raiz** de uma árvore dirigida *T* é um vértice ***r*** tal que qualquer outro vértice de *T* pode ser alcançado a partir de ***r***.

Exemplos



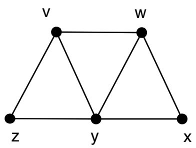
Árvore de decisão



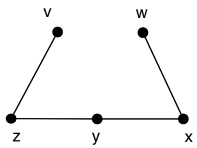
Parser

Árvore geradora (*spanning tree*)

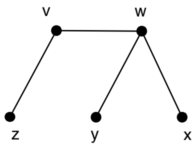
- Definição:
  Seja G um grafo conexo. Então uma árvore geradora em G é um subgrafo de G que inclui todos os seus vértices e também é uma árvore.



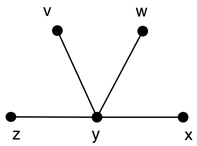
grafo G



árvore geradora



árvore geradora



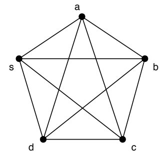
árvore geradora

Árvore geradora

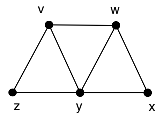
- Método construtivo:
  - selecione uma aresta de cada vez, de forma que não sejam criados ciclos;
  - continue este preocedimento até que todos os vértices sejam incluídos.
- Método de redução:
  - escolha qualquer ciclo e remova uma de suas arestas;
  - repita o procedimento até que não reste mais nenhum ciclo.

Exercícios

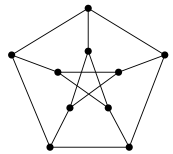
1. Use cada um dos métodos anteriores para construir uma árvore geradora num grafo completo K_5



2. O grafo ao lado possui 21 árvores geradoras. Encontre tantas quantas você conseguir.

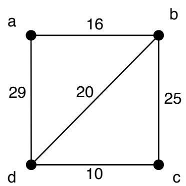


3. Encontre três árvores geradoras no Grafo de Petersen.

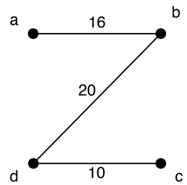


Problema do conector mínimo (*minimum spanning tree*)

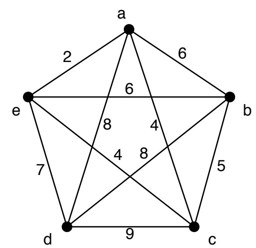
Suponha que precisamos projetar um sistema de canais de irrigação interconectando determinado número de localidades. O custo de construção de cada canal é conhecido. Por alguns motivos, alguns pares de localidades não podem ser conectados diretamente. Como podemos projetar o sistema que interconecte todas as localidades com custo total mínimo?



grafo valorado



árvore geradora



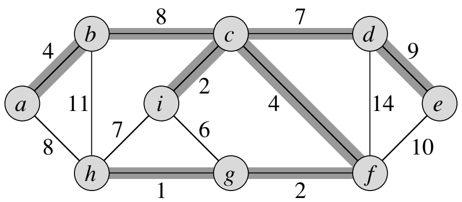
exercício

Problema do conector mínimo (*minimum spanning tree*)

- Definição:
  Seja T uma árvore geradora do grafo valorado G com custo total mínimo. Então T é uma árvore geradora de custo mínimo, ou um conector mínimo em G.
- Métodos de solução:
  - Algoritmo de Kruskal
  - Algoritmo de Prim

Algoritmo de Kruskal

- Para construir uma árvore geradora de custo mínimo em um grafo valorado conexo G, escolha sucessivamente arestas de G com custo mínimo de tal forma que não se formem ciclos, até que uma árvore geradora seja encontrada.



Algoritmo de Kruskal

O algoritmo de Kruskal utiliza estrutura auxiliar de conjuntos (SET) com as seguintes operações:

- $MAKE-SET(v)$: cria um conjunto contendo o vértice v, e atribui um identificador único a este conjunto;
- $FIND-SET(v)$: retorna o identificador do conjunto que contém o vértice v;
- $UNION(u, v)$: une os conjuntos que contém os vértices u e v, e atribui um identificador ao conjunto resultante.

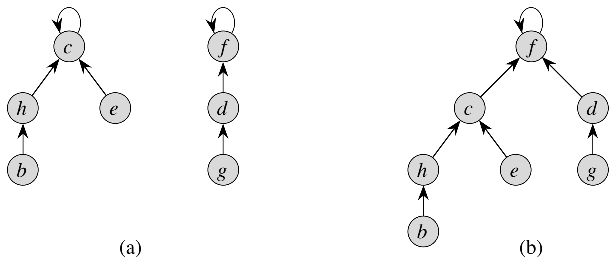
Algoritmo de Kruskal

Algoritmo: Kruskal(G, W)

$A \leftarrow \emptyset$;
para cada *vértice* $v \in V[G]$ **faça**
  $MAKE-SET(v)$;
Ordene as arestas de E em ordem não crescente de custo w;
para cada *aresta* $(u, v) \in E$, *em ordem não crescente de custo* **faça**
  se $FIND-SET(u) \neq FIND-SET(v)$ **então**
    $A \leftarrow A \cup (u, v)$;
    $UNION(u, v)$;
retorna A

Estrutura de dados para conjuntos disjuntos

Operações de criação, consulta e união devem ser eficientes (MAKE-SET, FIND-SET, UNION)



2 conjuntos UNION(h,g)

MAKE-SET(x)
```
p[x] ← x
rank[x] ← 0
```

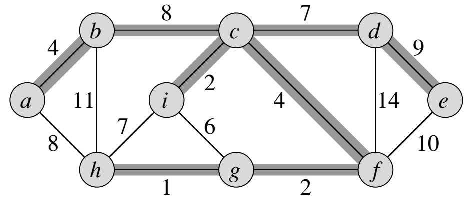
UNION(x, y)
```
LINK(FIND-SET(x), FIND-SET(y))
```

LINK(x, y)
```
if rank[x] > rank[y]
   then p[y] ← x
   else p[x] ← y
        if rank[x] = rank[y]
           then rank[y] ← rank[y] + 1
```

FIND-SET(x)
```
if x ≠ p[x]
   then p[x] ← FIND-SET(p[x])
return p[x]
```


Algoritmo de Prim

- Para construir uma árvore geradora de custo mínimo *T* em um grafo valorado conexo G, proceda passo a passo da seguinte forma:
  - coloque um vértice arbitrário em *T;*
  - sucessivamente adicione arestas com custo mínimo juntando um vértice em *T* com outro vértice fora de *T, até obter a árvore geradora.*



Algoritmo de Prim

Elementos do algoritmo de Prim:

- r: vértice inicial (*root, raiz*);
- Q: fila de prioridades;
- $chave(v)$: custo mínimo (atual) para conectar o vértice v à árvore;
- $\pi(v)$: vértice predecessor de v na árvore. Indica em qual vértice ele deve se conectar à árvore cm custo atual $chave(v)$;

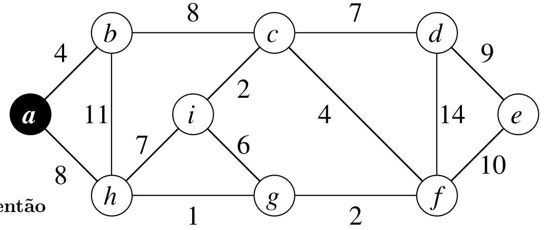
Algoritmo: Prim(G, W)

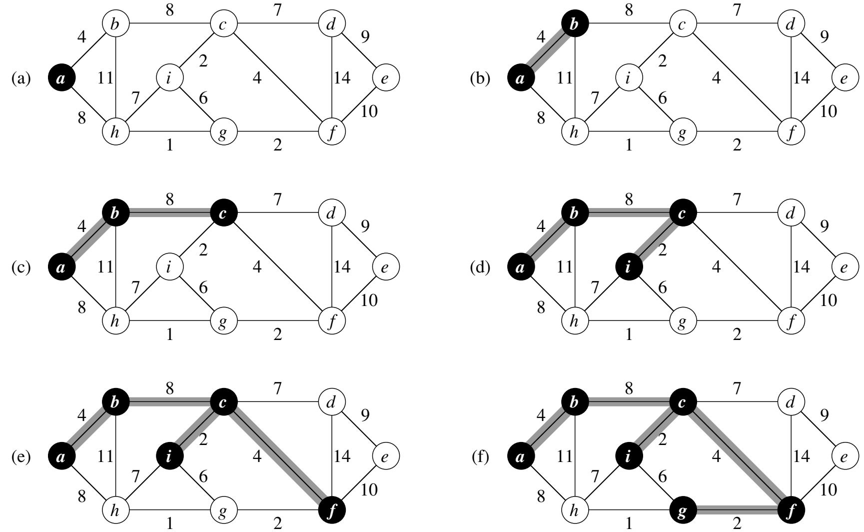
$Q \leftarrow V[G]$;
para cada *vértice* $u \in Q$ **faça**
  $chave[u] \leftarrow \infty$;
$chave[r] \leftarrow 0$;
$\pi[r] \leftarrow NIL$;
enquanto $Q \neq \emptyset$ **faça**
  $u \leftarrow REMOVE-MINIMO(Q)$;
  para cada $v \in Adj[u]$ **faça**
    se $v \in Q$ E $w(u,v) < chave[v]$ **então**
      $\pi[v] \leftarrow u$;
      $chave[v] \leftarrow w(u,v)$;

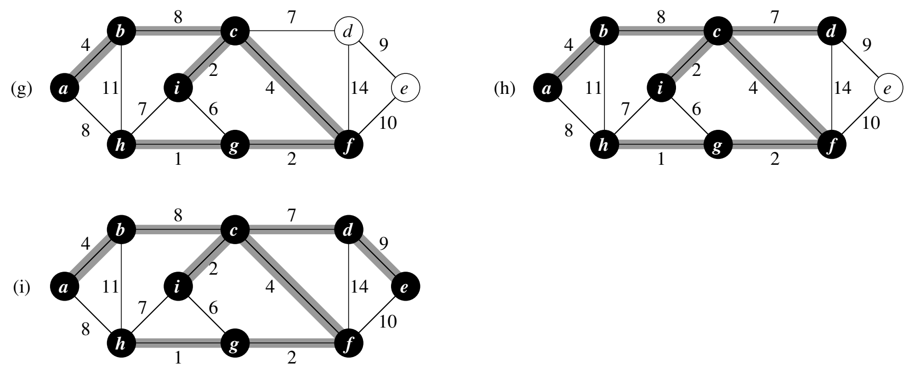
Iter	u	a		b		c		d		e		f		g		h		i	
		ch	π	ch	π	ch	π	ch	π	ch	π	ch	π	ch	π	ch	π	ch	π
inicio		0	nil	∞	nil	∞	nil	∞	nil	∞	nil	∞	nil	∞	nil	∞	nil	∞	nil
1	a	*	*	4	a	∞	nil	∞	nil	∞	nil	∞	nil	∞	nil	8	a	∞	nil
2	b	*	*	*	*	8	b	∞	nil	∞	nil	∞	nil	∞	nil	8	a	∞	nil
3	c	*	*	*	*	*	*	7	c	∞	nil	4	c	∞	nil	8	a	2	c
4	i	*	*	*	*	*	*	7	c	∞	nil	4	c	6	i	7	i	*	*
5	f	*	*	*	*	*	*	7	c	10	f	*	*	2	f	7	i	*	*
6	g	*	*	*	*	*	*	7	c	10	f	*	*	*	*	1	g	*	*
7	h	*	*	*	*	*	*	7	c	10	f	*	*	*	*	*	*	*	*
8	d	*	*	*	*	*	*	*	*	9	d	*	*	*	*	*	*	*	*
9	e	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

Algoritmo: Prim(G, W)

$Q \leftarrow V[G]$;
para cada *vértice* $u \in Q$ **faça**
  $chave[u] \leftarrow \infty$;
$chave[r] \leftarrow 0$;
$\pi[r] \leftarrow NIL$;
enquanto $Q \neq \emptyset$ **faça**
  $u \leftarrow REMOVE-MINIMO(Q)$;
  para cada $v \in Adj[u]$ **faça**
    se $v \in Q$ E $w(u,v) < chave[v]$ **então**
      $\pi[v] \leftarrow u$;
      $chave[v] \leftarrow w(u,v)$;







Exercícios

1. Sabendo que uma árvore é um tipo particular de grafo, escreva pelo menos cinco definições diferentes para árvores.

2. Desenhe um exemplo de uma árvore com sete vértices e...

   a) exatamente dois vértices de grau 1;

   b) exatamente quatro vértices de grau 1;

   c) exatamente seis vértices de grau 1.

3. Utilize o "lema do aperto de mão" (*handshaking lemma*) para provar que qualquer árvore com n vértices, onde $n \geq 2$, possui pelo menos dois vértices de grau 1.

4. Uma **floresta** é um grafo (não necessariamente conexo), onde cada uma de suas componentes conexas é uma árvore.

   a) Seja G uma floresta com n vértices e k componentes conexas. Quantas arestas existem em G?

   b) Construa (e desenhe) uma floresta com 12 vértices e 9 arestas.

   c) É verdade que qualquer floresta com k componentes possui pelo menos $2k$ vértices de grau 1?

5. Para cada um dos ítens abaixo, tente desenhar um grafo que atenda a descrição dada, ou então explique porque tal grafo não existe.

   a) Um grafo simples com 6 vértices, 2 componentes conexas e 11 arestas.

   b) Um grafo simples com 7 vértices, 3 componentes conexas e 10 arestas.

   c) Um grafo simples com 8 vértices, 2 componentes conexas, 9 arestas e exatamente 3 ciclos.

   d) Um grafo simples com 8 vértices, 2 componentes conexas, 10 arestas e exatamente 3 ciclos.

   e) Um grafo simples com 9 vértices, 2 componentes conexas, 10 arestas e exatamente 2 ciclos.

   f) Um grafo simples conexo com 9 vértices, 12 arestas e exatamente 2 ciclos.

   g) Um grafo simples conexo com 9 vértices, 12 arestas e exatamente 3 ciclos.

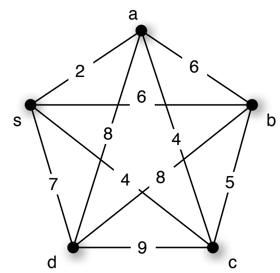
   h) Uma floresta com 10 vértices, 2 componentes conexas e 9 arestas.

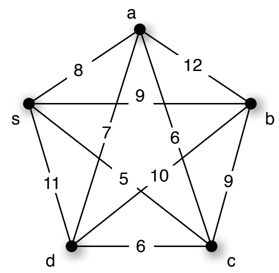
   i) Uma floresta com 10 vértices, 3 componentes conexas e 9 arestas.

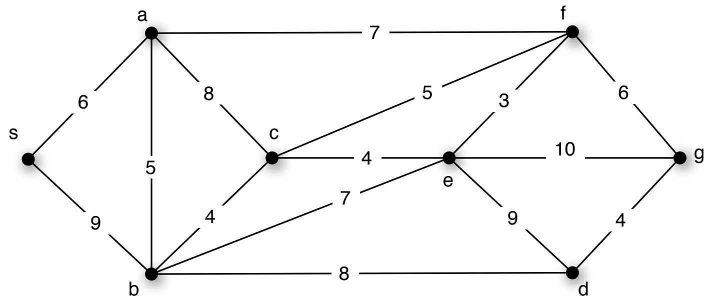
   j) Um grafo simples conexo com 11 vértices, 14 arestas e 5 ciclos disjuntos (ciclos que não compartilham arestas).

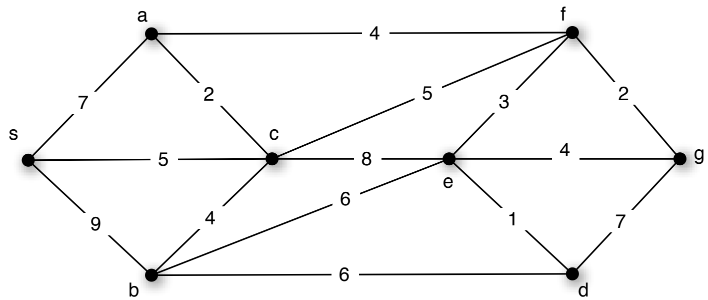
6. Prove que a seguinte afirmação é verdadeira ou falsa: "Qualquer grafo simples conexo com n vértices e n arestas deve conter exatamente 1 ciclo".

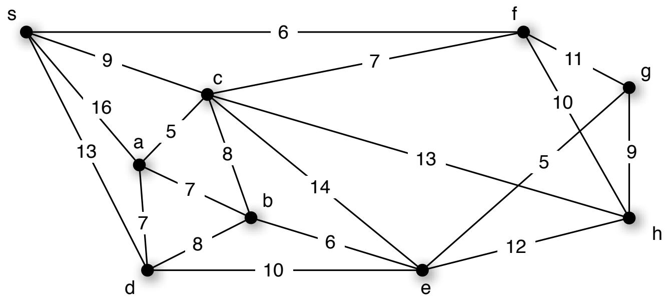
7. Dados os grafos abaixo, encontre suas respectivas árvores geradoras de custo mínimo utilizando o algoritmo de Prim. Mostre passo a passo a solução de acordo com os passos de execução do algoritmo, mostrando os valores de $chave(v)$ e $\pi(v)$ para todos os vértices. Utilize o vértice s como vértice inicial e resolva empates dando preferência para os vértices em ordem alfabética. Desenhe as árvores resultantes e calcule seus custos.

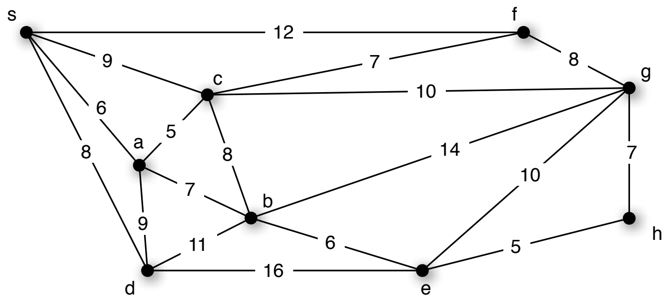












8. Encontre as respectivas árvores geradoras de custo mínimo dos grafos da questão anterior utilizando o algoritmo de Kruskal. Escreva a lista de arestas em ordem crescente de custo. Escreva a lista de conjuntos de vértices em cada passo do algoritmo, indicando a ocorrência de operações $UNION(u, v)$. Desenhe as árvores resultantes e calcule seus custos.

9. Adapte os algoritmos de Prim e Kruskal para encontrar árvores geradoras de custo **máximo** de grafos valorados.

10. Implemente o algoritmo de prim.

11. Implemente o algoritmo de Kruskal.

Fim da Unidade 6: Àrvores

Teoria dos Grafos
Unidade 6: Árvores

Prof. Dr. Paulo César Rodacki Gomes
paulo.rodacki@blumenau.ifc.edu.br

INSTITUTO FEDERAL
CATARINENSE
Câmpus Blumenau