Departamento de Sistemas e Computação – FURB
Curso de Ciência da Computação
Disciplina de Teoria dos Grafos

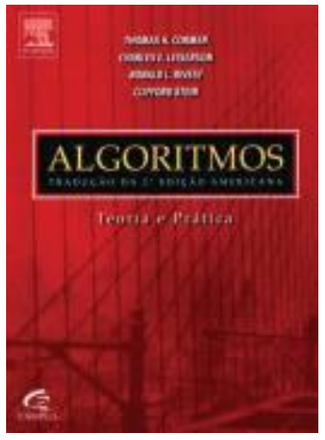
Árvore geradora

Prof. Aurélio Hoppe
aureliof@furb.br
http://www.inf.furb.br/~aurelio/

Grupo de Processamento de Imagens,
Análise de dados, Robótica e
Simulação computacional

Bibliografia

Márcia A. Rabuske. **Introdução à Teoria dos Grafos**. Editora da UFSC. 1992



Thomas Cormen et al. **Algoritmos**: teoria e prática. Ed. Campus. 2004.

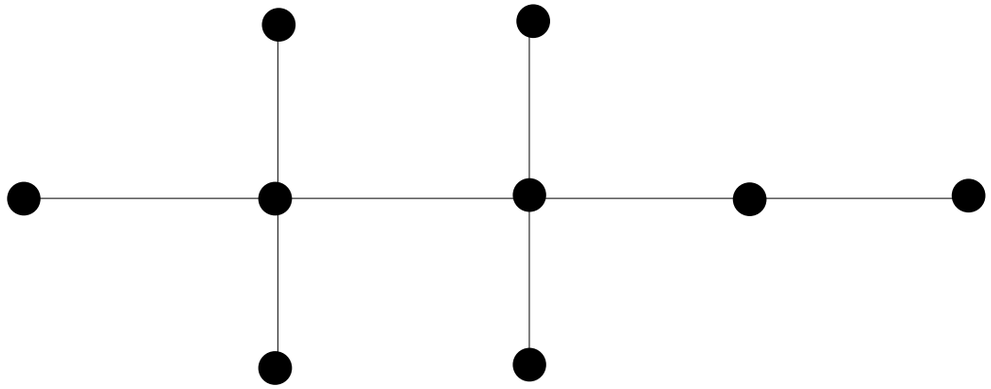
Joan M. Aldous, Robin J. Wilson. **Graphs and Applications**: as introductory approach. Springer. 2001

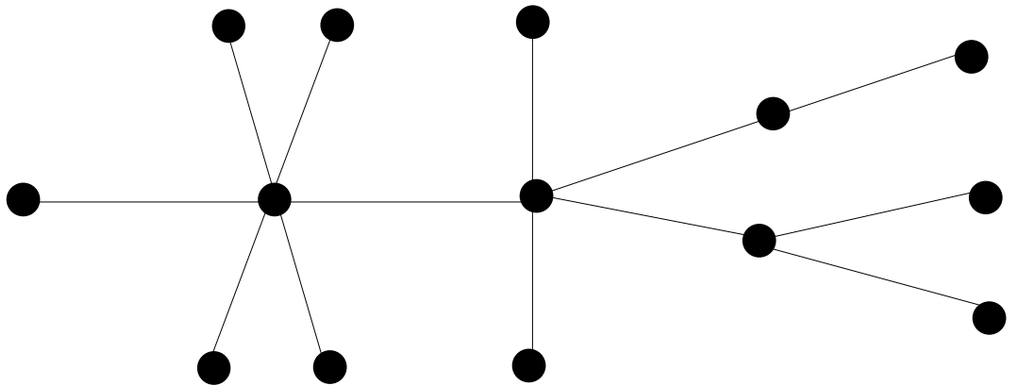
Definições

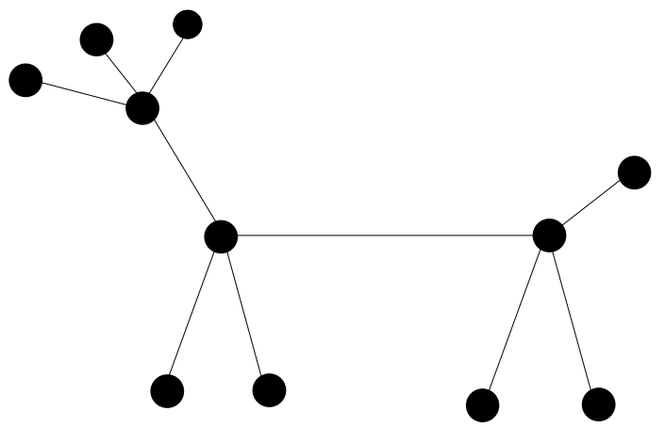
Uma árvore é:

- um grafo conexo com n vértices e n-1 arestas
- um grafo conexo sem ciclos
- um grafo no qual cada par de vértices é ligado por um e somente um caminho simples
- um grafo conexo onde todas as arestas são “pontes”

Árvores - exemplos



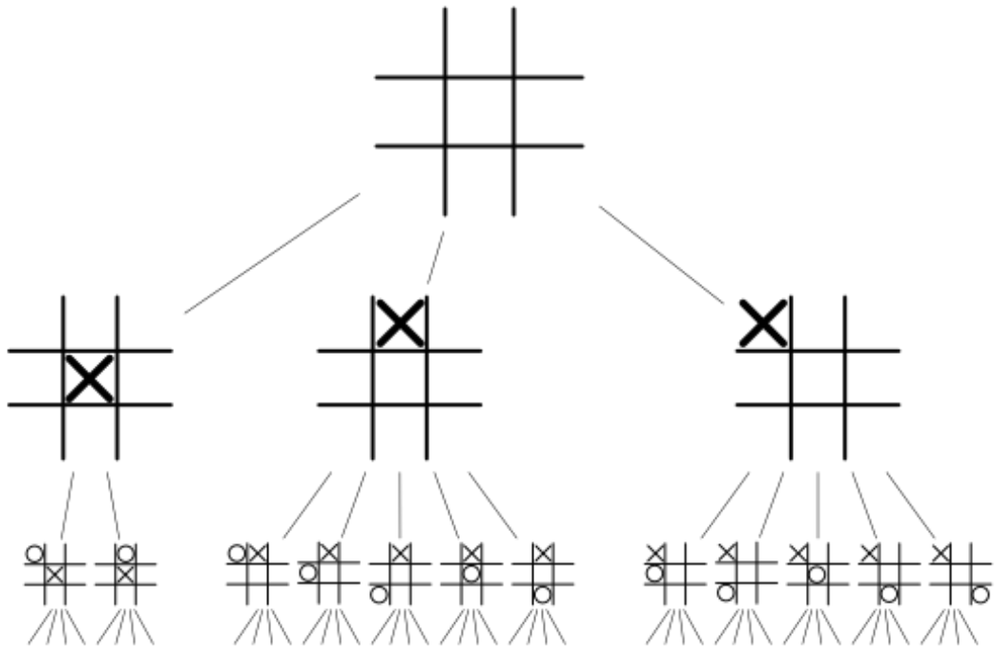




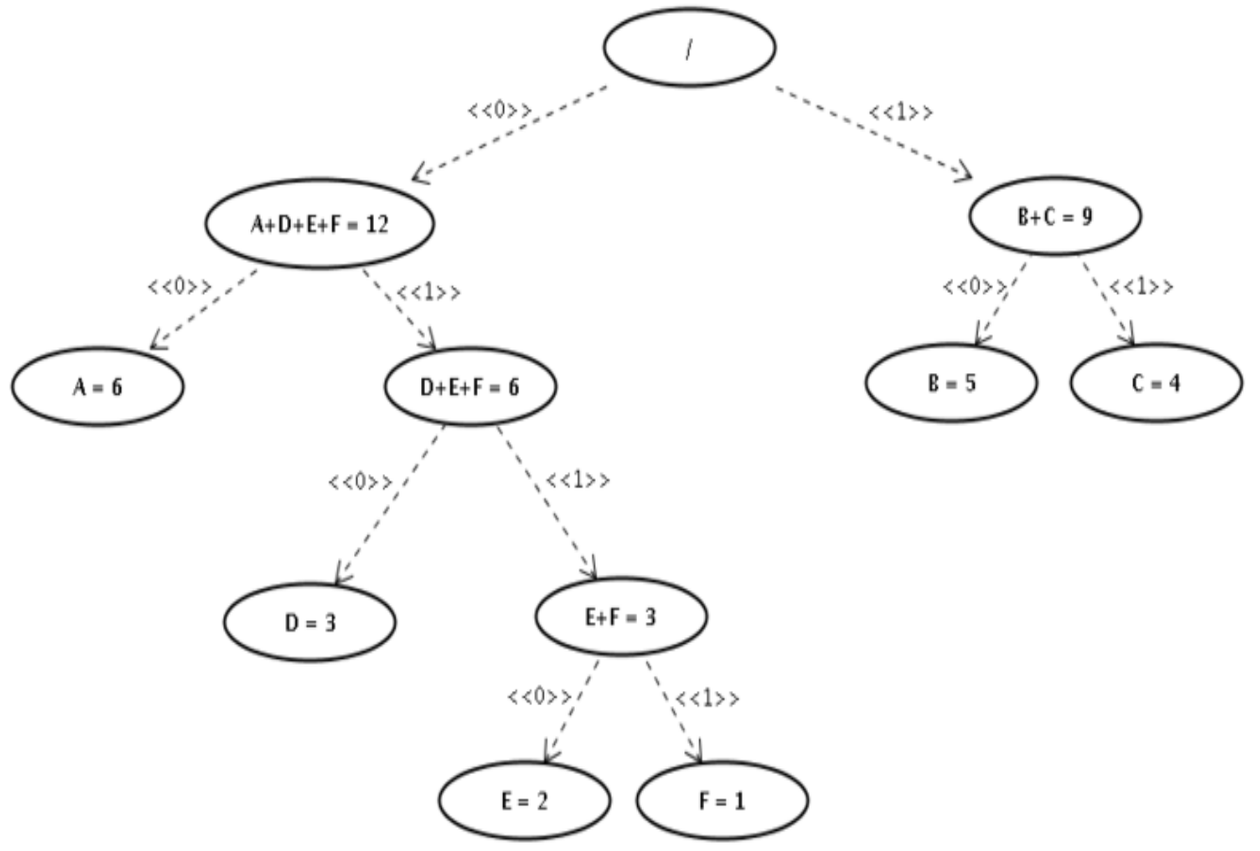
Definições

- 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



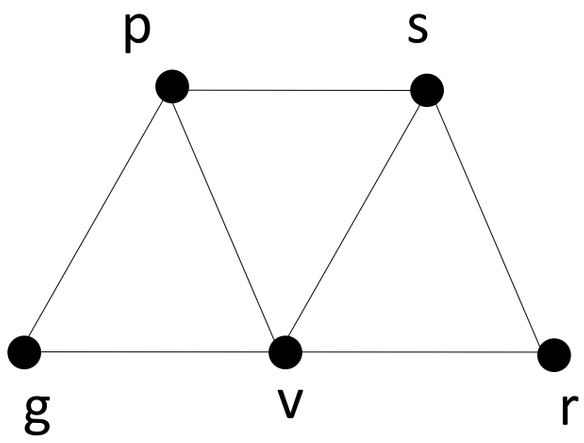
Árvore de decisão



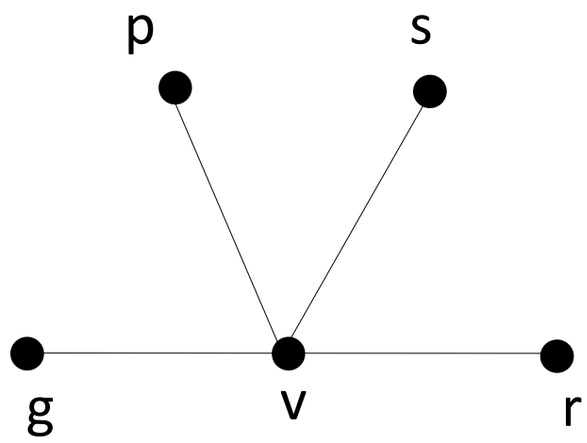
Codificação de Hoffman

Árvore geradora (*spanning tree*)

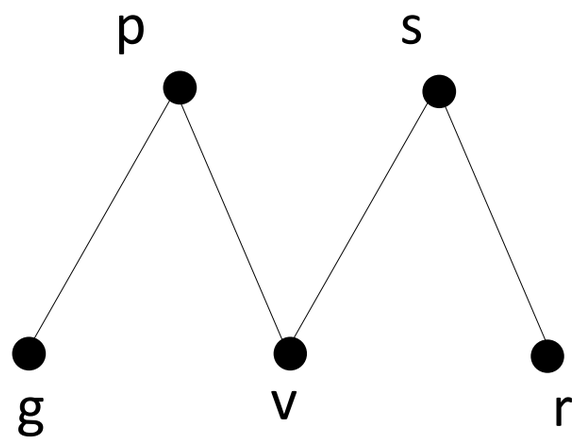
Definição:
Seja G um grafo conexo. Então uma árvore geradora em G é um subgrafo de G que inclui todos os seus vértices e também é uma árvore.



Grafo G



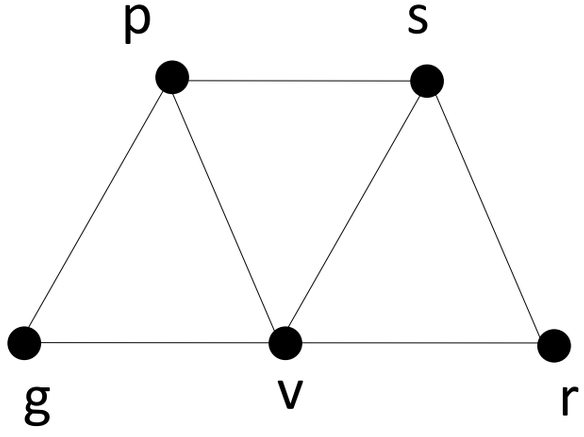
Árvore geradora



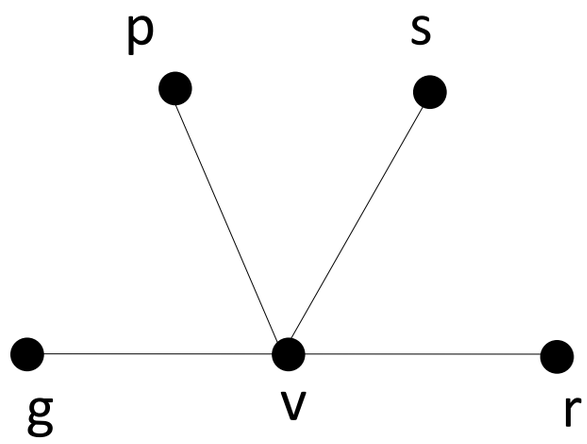
Árvore geradora

Teorema de Cayley

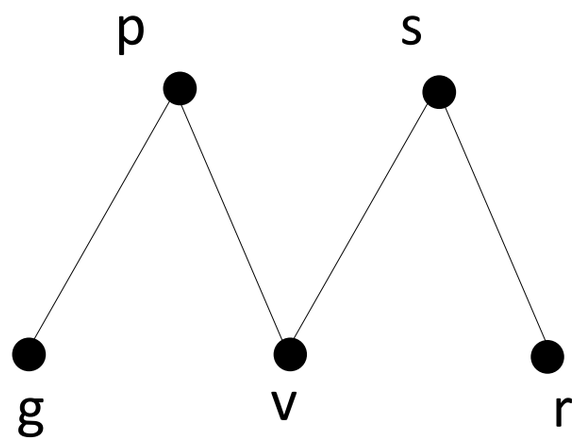
O número de árvores rotuladas com n vértices é igual a n^{n-2}



Grafo G



Árvore geradora



Árvore geradora

Árvore geradora

Método construtivo:

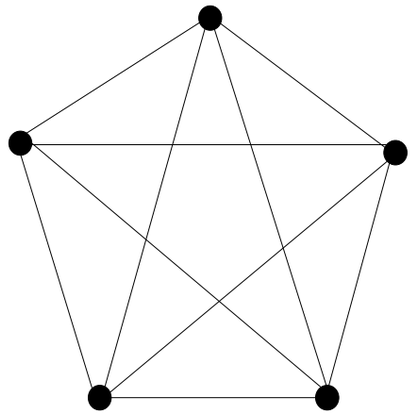
- selecione uma aresta de cada vez, de forma que não sejam criados ciclos;
- continue este procedimento até que todos os vértices sejam incluídos

Método de redução:

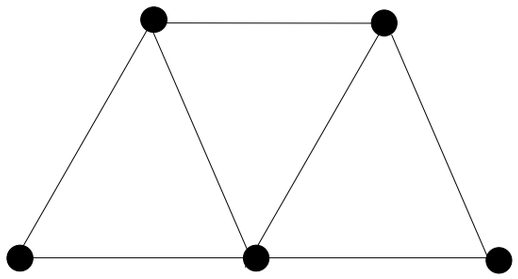
- escolha qualquer ciclo e remova uma de suas arestas
- repita o procedimento até que não reste mais nenhum ciclo.

Exercícios

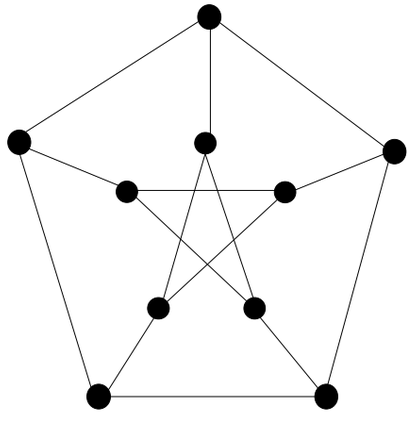
1. Use cada um dos métodos anteriores para construir uma árvore geradora num grafo completo K_5



2. O grafo ao lado possui 125 árvores geradoras. Encontre tantas quantas você conseguir.



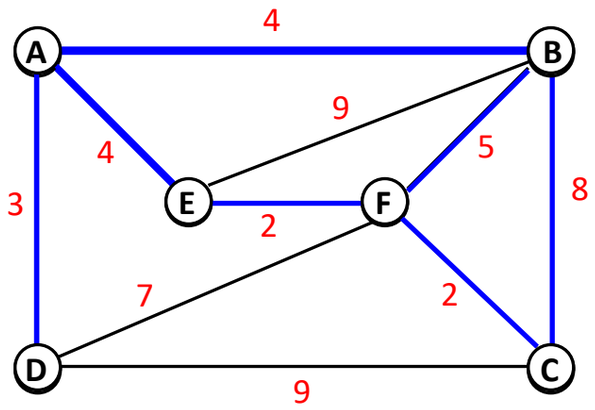
3. Encontre três árvores geradoras no Grafo de Petersen.

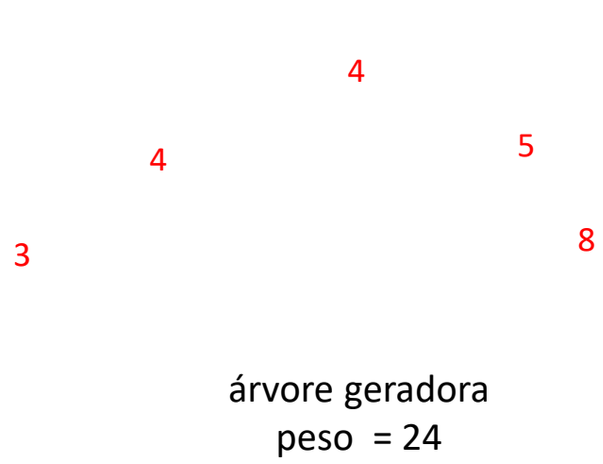


Problema do conector mínimo (*minimum spanning tree*)

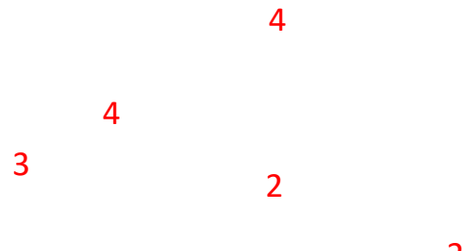
- Suponha que precisamos projetar um sistema de canais de irrigação interconectando determinado número de localidades. O custo de construção de cada canal é conhecido. Por alguns motivos, alguns pares de localidades não podem ser conectados diretamente. Como podemos projetar o sistema que interconecte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

Exemplo:





árvore geradora
peso = 24



árvore geradora
peso = 15

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

Árvore geradora mínima

Algoritmo de Prim:

- gera uma árvore única
- ao longo do algoritmo, o conjunto X sempre é uma árvore

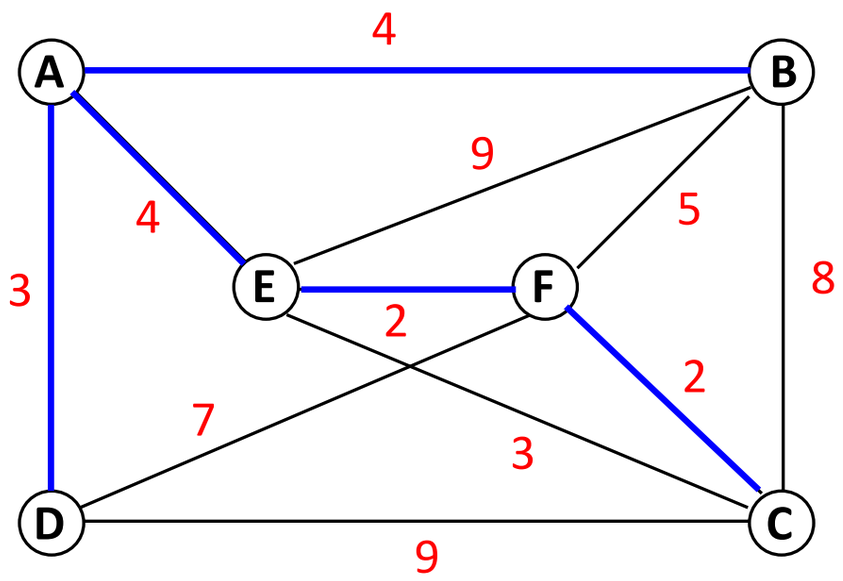
Algoritmo de Kruskal:

- gera uma floresta, antes de gerar a AGM
- existe garantia de ser apenas uma árvore apenas depois da última iteração

Algoritmo de Kruskal

Ideia

- 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



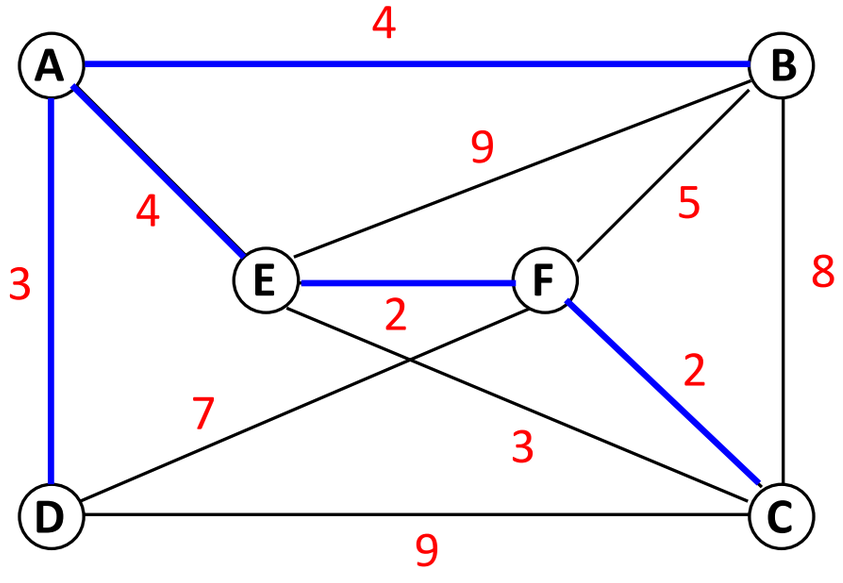
Algoritmo de Kruskal

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

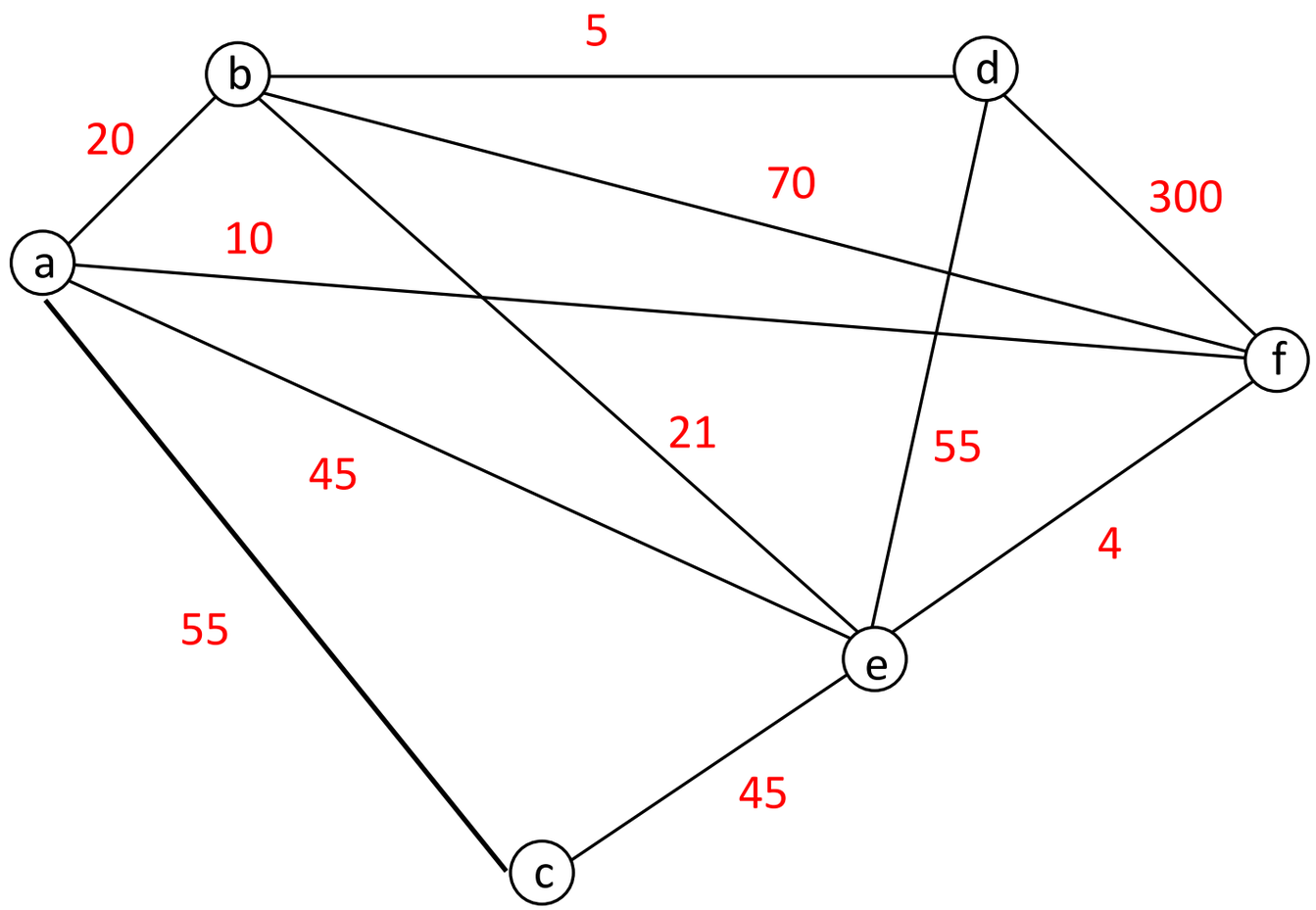
- MAKE-SET(v): cria um conjunto contendo o vértice v, e atribui um identificador único a este conjunto
- FIND-SET(v): retorna o identificador do conjunto que contém o vértice v
- UNION(u,v): une os conjuntos que contém os vértices u e v, e atribui um identificador ao conjunto resultante

Algoritmo de Kruskal

```
KRUSKAL(G, w)
01. A ← ∅
02. para cada vértice v ∈ V|G| faça
03.     MAKE-SET(V);
04. ordene as arestas de E em ordem crescente de custo w
05. para cada aresta(u,v) ∈ E, em ordem crescente de custo faça
06.     se FIND-SET(u) ≠ FIND-SET(v) então
07.         A ← A ∪ [u,v]
08.         UNION(u,v)
09. retorna A
```



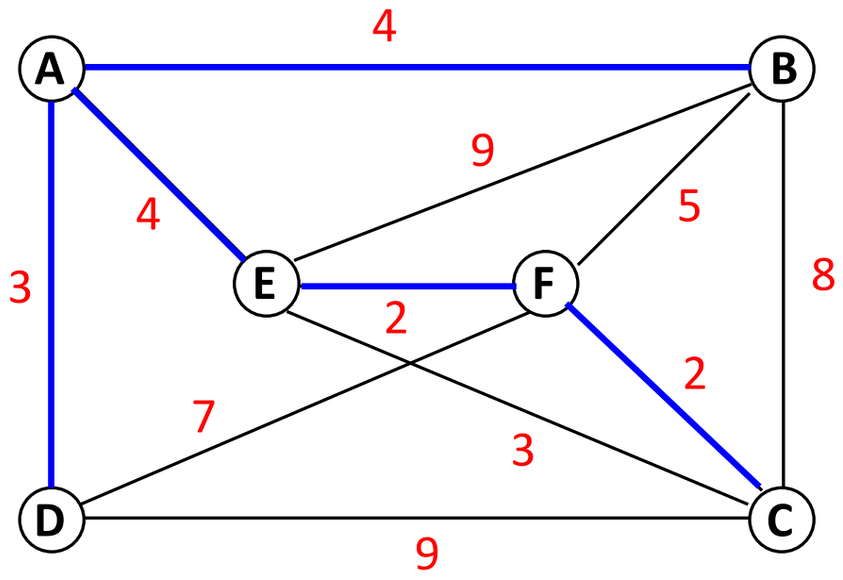
Exercícios



```
KRUSKAL(G, w)
01. A ← ∅
02. para cada vértice v ∈ V|G| faça
03.     MAKE-SET(V);
04. ordene as arestas de E em ordem crescente de custo w
05. para cada aresta(u,v) ∈ E, em ordem crescente de custo faça
06.     se FIND-SET(u) ≠ FIND-SET(v) então
07.         A ← A ∪ [u,v]
08.         UNION(u,v)
09. retorna A
```

Algoritmo de Kruskal

Exemplo:



e	c(e)
(C,F)	2
(E,F)	2
(A,D)	3
(C,E)	3
(A,B)	4
(A,E)	4
(B,F)	5
(D,F)	7
(B,C)	8
(B,E)	9
(C,D)	9

Lista L

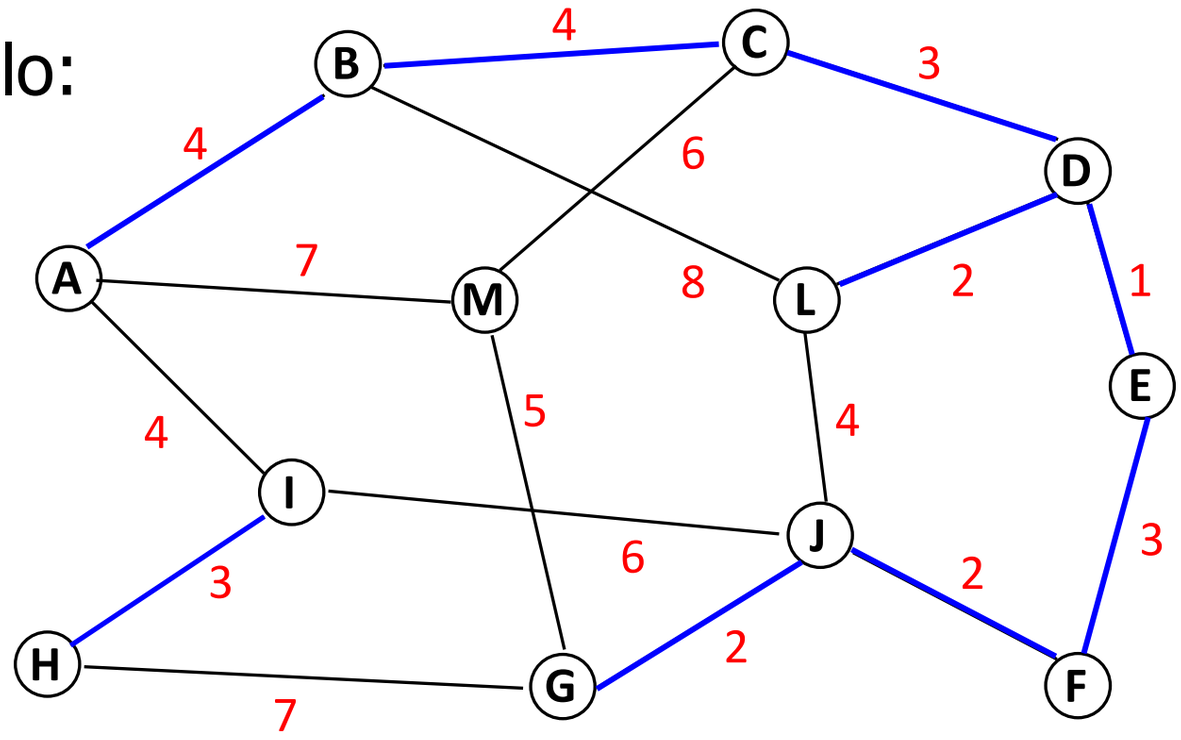
Subárvores

{ A, B, C, D, E, F }

c(F) = 15

Algoritmo de Kruskal

Exemplo:



e	c(e)
(D,E)	1
(D,L)	2
(F,J)	2
(G,J)	2
(C,D)	3
(E,F)	3
(H,I)	3
(A,B)	4
(B,C)	4
...	...

Lista L

Subárvores

{ A, B, C, D, E, F, G, J, L }

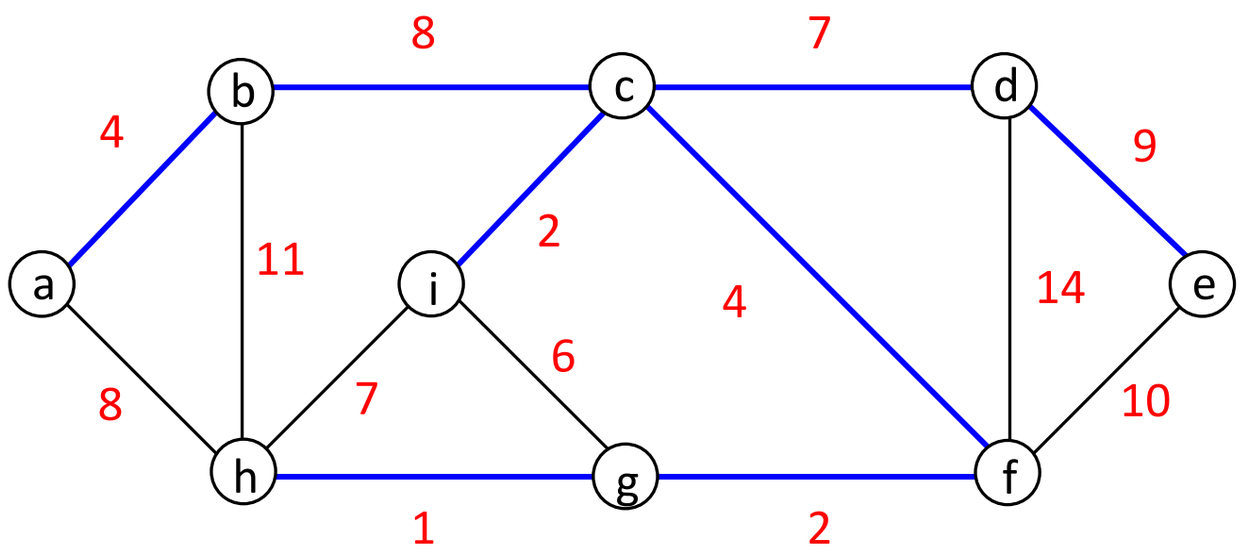
{ H, I } { M }

c(F) = 24

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.



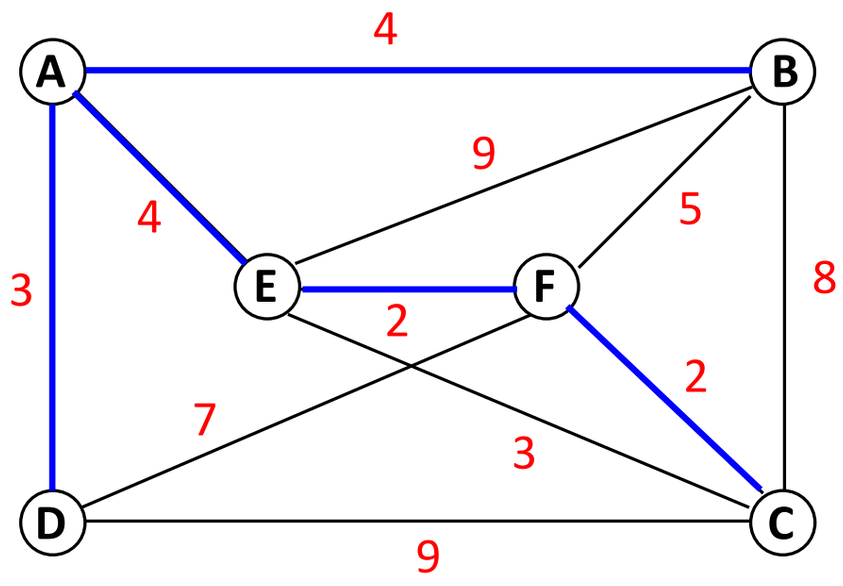
Algoritmo de Prim

Elementos do algoritmo de Prim:

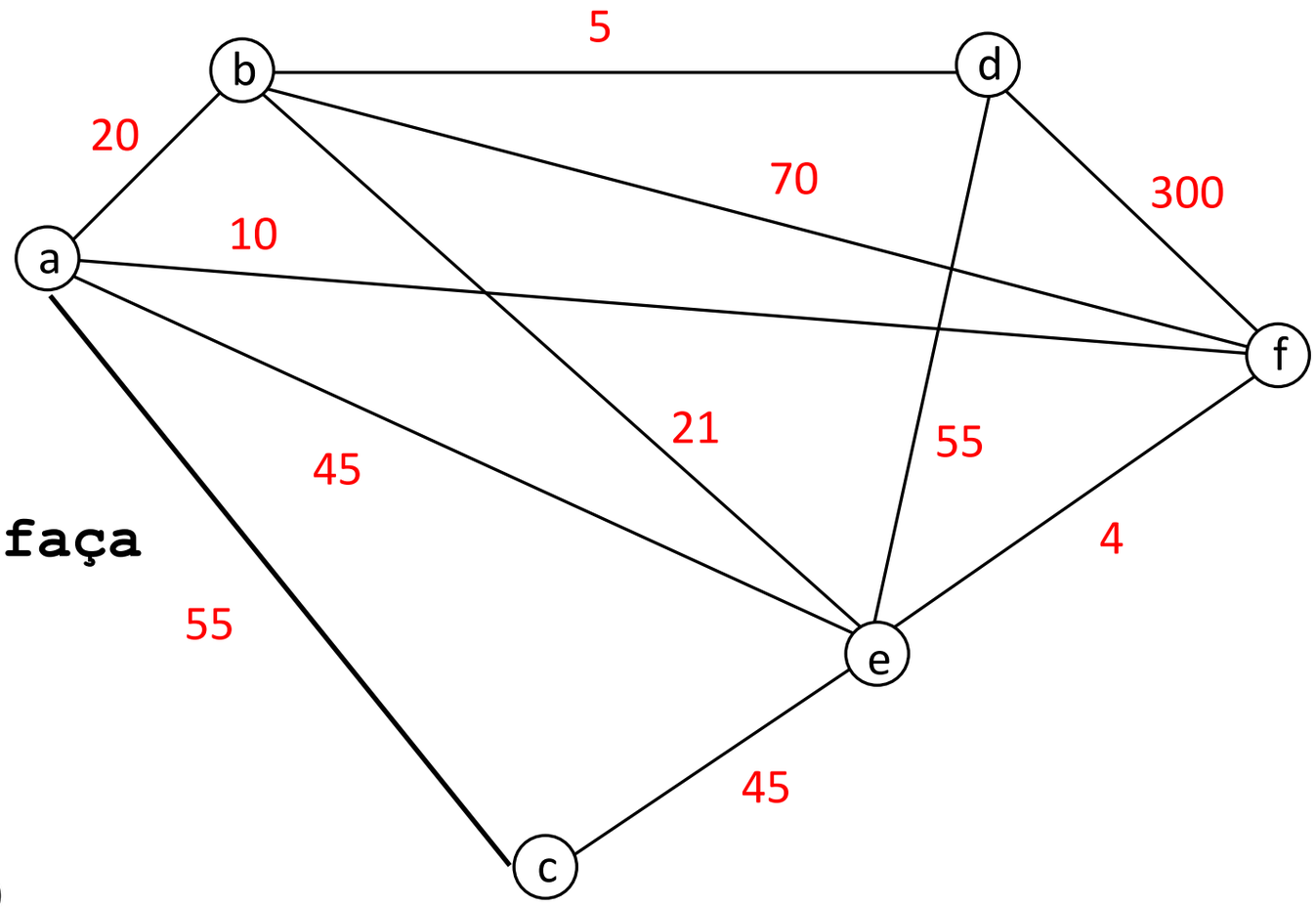
- r: vértice inicial (root, raiz)
- Q: fila de prioridades
- chave(v): custo mínimo (atual) para conectar o vértice v à árvore
- Π(v): vértice predecessor de v na árvore. Indica em qual vértice ele deve se conectar à árvore em custo atual chave(v)

Algoritmo de Prim

```
PRIM(G, w)
01. Q ← V[G]
02. para cada vértice u ∈ Q faça
03.     chave[u] ← ∞;
04. chave[r] ← 0;
05. Π[r] ← NIL
06. enquanto Q ≠ ∅ faça
07.     u ← REMOVE-MINIMO(Q)
08.     para cada v ∈ Adj[u] faça
09.         se v ∈ Q E w(u,v) < chave[v] então
10.             Π[v] ← u
11.             chave[v] ← w(u,v)
```



Algoritmo de Prim




```
PRIM(G, w)
01. Q ← V[G]
02. para cada vértice u ∈ Q faça
03.       chave[u] ← ∞;
04. chave[r] ← 0;
05. Π[r] ← NIL
06. enquanto Q ≠ ∅ faça
07.       u ← REMOVE-MINIMO(Q)
08.       para cada v ∈ Adj[u] faça
09.             se v ∈ Q E w(u,v) < chave[v] então
10.                Π[v] ← u
11.                chave[v] ← w(u,v)
```

Exercício