Prof. Lucas de Souza Batista - DEE/EE/UFMG

Otimização de Redes

Árvore Geradora Mínima

Árvore Geradora Mínima

- Árvore geradora de custo mínimo.
- No problema da AGM, para um conjunto de *n* nós da rede, o objetivo é encontrar o conjunto de *n-1* arcos que conectam todos os nós e não contenha ciclo, de forma que a soma dos fluxos dos arcos seja minimizada.

Árvore Geradora Mínima

- Exemplos de aplicações:
  - projeto de redes de telecomunicações;
  - projeto de rodovias, ferrovias;
  - projeto de redes de distribuição de água e esgoto;
  - projeto de redes de transmissão de energia, etc.

Formulação (Arenales et al. 2007)

- O modelo matemático pode ser entendido como um caso particular do *problema de transporte* em que existe suprimento de *n-1* unidades de um produto em apenas um dos nós e uma demanda de exatamente *1* unidade do produto em cada um dos demais *n-1* nós.
- x_{ij} é a quantidade de produto transportada pela aresta (i,j).
- y_{ij} é uma variável binária que indica se a aresta (i,j) é usada.
- c_{ij} é o custo associado à utilizada da aresta (i,j).

Formulação (Arenales et al. 2007)

$$\min \sum_{(i,j)\in E} c_{ij} y_{ij}$$

$$\sum_{\{j:(1,j)\in E\}} x_{1j} = n - 1$$

$$\sum_{\{i:(i,j)\in E\}} x_{ij} - \sum_{\{k:(j,k)\in E\}} x_{jk} = 1 \ , \quad j = 2, \ldots, n$$

$$x_{ij} \geq 0 \ , \qquad (i,j) \in E$$

$$(n-1) y_{ij} \geq x_{ij} \ , \quad (i,j) \in E$$

$$y_{ij} \leq x_{ij} \ , \qquad (i,j) \in E$$

$$y_{ij} \in \{0,1\} \ , \qquad (i,j) \in E$$

Formulação (Arenales et al. 2007)

- O método Simplex pode ser aplicado à relaxação linear do problema, assumindo $0 \leq y_{ij} \leq 1$.
- Pode gerar uma solução que não seja uma árvore,
- Porém, uma solução alternativa que represente uma árvore pode ser facilmente obtida.

Formulação (Belfiore et al. 2013)

- Assuma um grafo $G = (N,E)$, em que $|N|=n$. Suponha também um subgrafo $G'(N',E')$ de G, tal que $|N'|=m$.
- x_{ij} é igual a *1* se a aresta (i,j) está contida na AGM e igual a *0* caso contrário.
- c_{ij} é o custo associado à utilizada da aresta (i,j).

Formulação (Belfiore et al. 2013)

$$\min \sum_{(i,j)\in E} c_{ij} x_{ij}$$

$$\sum_{(i,j)\in E} x_{ij} = n - 1$$

$$\sum_{(i,j)\in E'} x_{ij} \leq m - 1, \quad \forall G' \subset G \ , \ 2 \leq m \leq n$$

$$x_{ij} \in \{0, 1\} \ , \qquad \forall (i,j) \in E$$

Formulação (Belfiore et al. 2013)

- A solução sempre será uma árvore!
- À medida que o número de nós aumenta, o número de restrições cresce exponencialmente;
- O problema pode se tornar computacionalmente intratável.

Algoritmo de Prim

- Inicia-se com um nó qualquer da rede (que passa a pertencer à árvore) e, a cada iteração, adiciona o nó mais próximo da árvore, conectando-o à árvore por meio de um arco.
- O processo se repete até que todos os nós da rede estejam contidos na árvore.

Algoritmo de Prim

Entradas:

- $G(V,E)$
- $c(i,j)$: custo da aresta (i,j)

Saídas:

- $T(V,E')$

1. $T \leftarrow \emptyset$.
2. $B \leftarrow$ qualquer $v \in V$.
3. **Enquanto** $B \neq V$:

   A. $(u,v) \leftarrow$ aresta (u,v), com $u \in \{V - B\} \wedge v \in B$, tal que $c(u,v)$ seja mínimo.

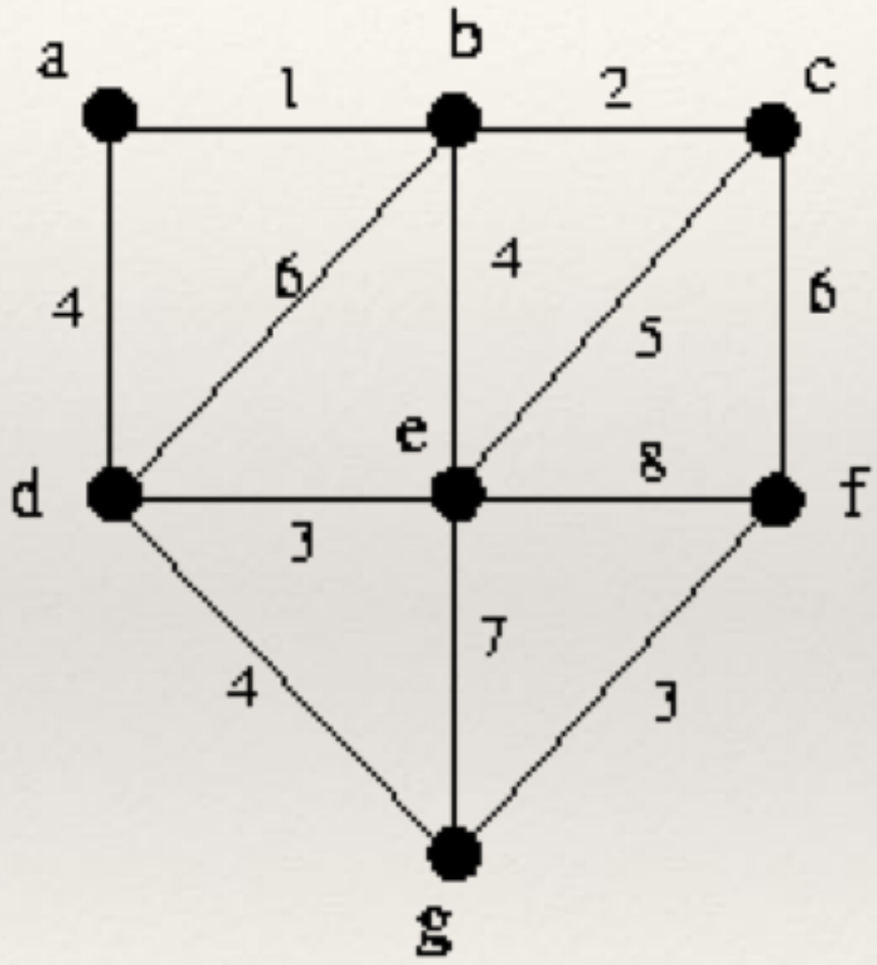
   B. $T \leftarrow T + \{(u,v)\}$.

   C. $B \leftarrow B + \{u\}$.
4. **Retornar** T.

Estudo de complexidade:

- Complexidade global: $O(|V|^2)$.

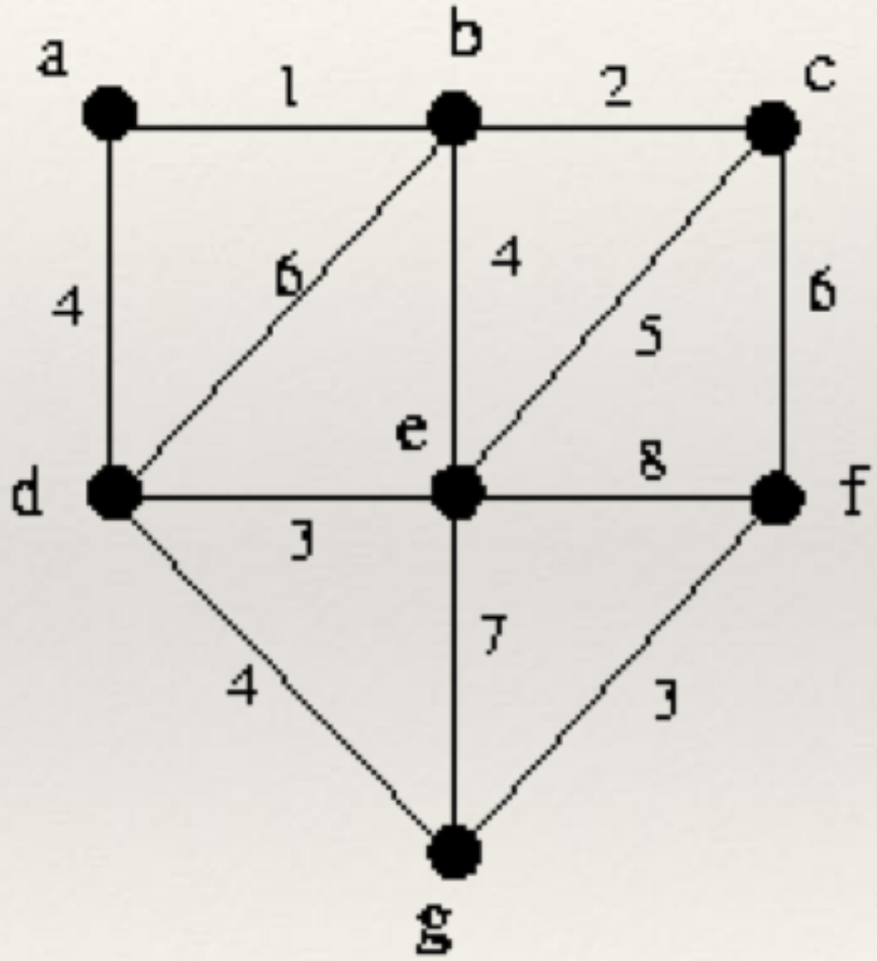
Exemplo

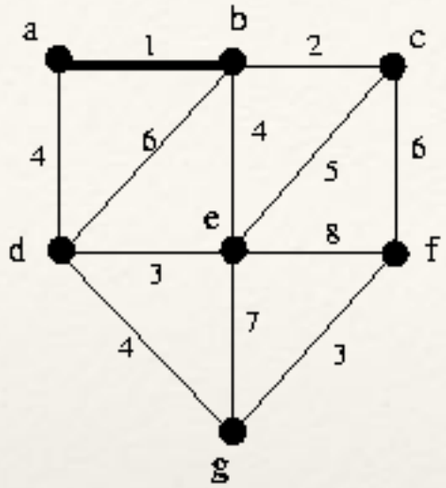


Exemplo

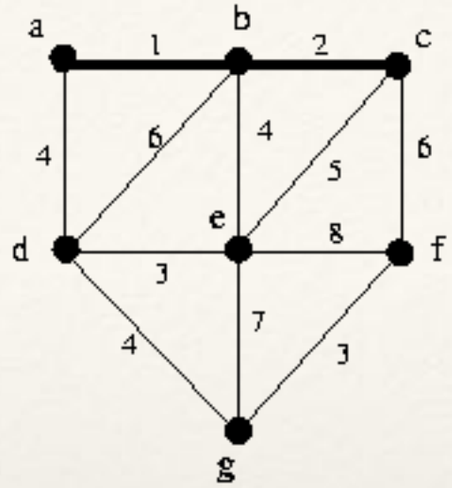
Inicial:

- $T=\emptyset$.
- $B=\{a\}$.

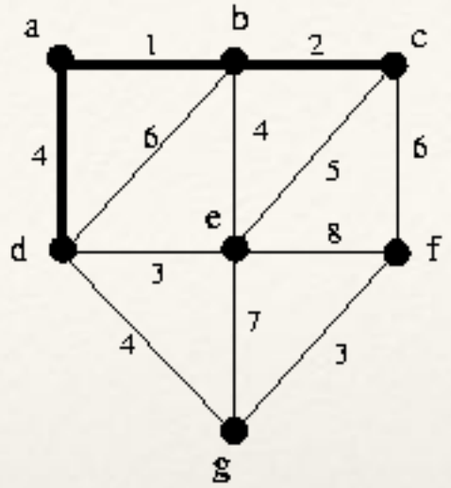



a
b
c
d
e
f
g
1
2
4
6
4
5
6
8
3
7
4
3

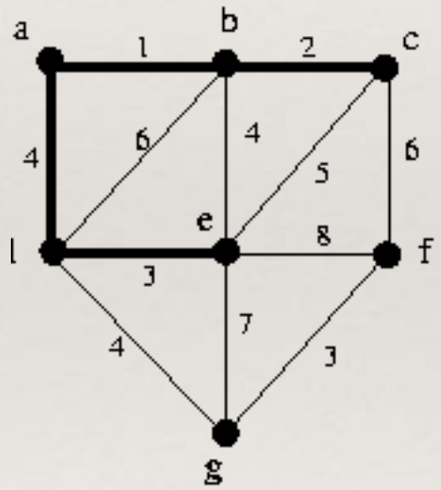
(a)


a
b
c
d
e
f
g
1
2
4
6
4
5
6
8
3
7
4
3

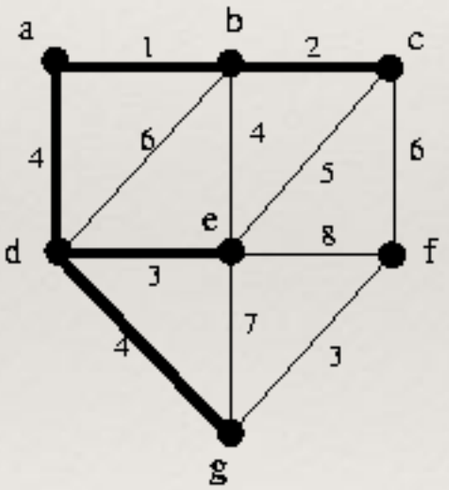
(b)


a
b
c
d
e
f
g
1
2
4
6
4
5
6
8
3
7
4
3

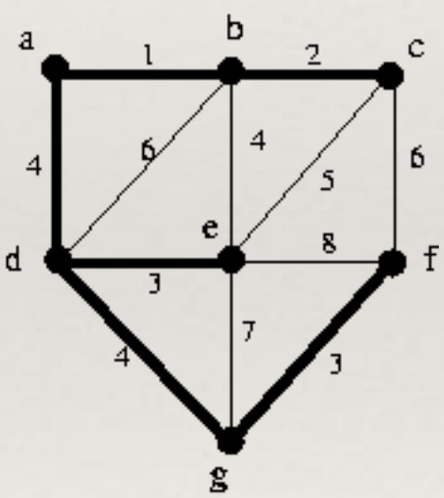
(c)


a
b
c
l
e
f
g
1
2
4
6
4
5
6
8
3
7
4
3

(d)


a
b
c
d
e
f
g
1
2
4
6
4
5
6
8
3
7
4
3

(e)


a
b
c
d
e
f
g
1
2
4
6
4
5
6
8
3
7
4
3

(f)

It.	*B:*	*T:*
-	*{a}*	*∅*
1)	*{a,b}*	*{(a,b)}*
2)	*{a,b,c}*	*{(a,b),(c,b)}*
3)	*{a,b,c,d}*	*{(a,b),(c,b),(a,d)}*
4)	*{a,b,c,d,e}*	*{(a,b),(c,b),(a,d),(d,e)}*
5)	*{a,b,c,d,e,g}*	*{(a,b),(c,b),(a,d),(d,e),(d,g)}*
6)	*{a,b,c,d,e,g,f}*	*{(a,b),(c,b),(a,d),(d,e),(d,g),(f,g)}*

Algoritmo de Kruskal

- O Algoritmo de Kruskal emprega um procedimento guloso em que sempre escolhe, sucessivamente, as arestas de menor custo.
- Esse processo continua até formar a árvore geradora mínima.

Algoritmo de Kruskal

Entradas:

- $G(V,E)$
- $c(i,j)$: custo da aresta (i,j)

Saídas:

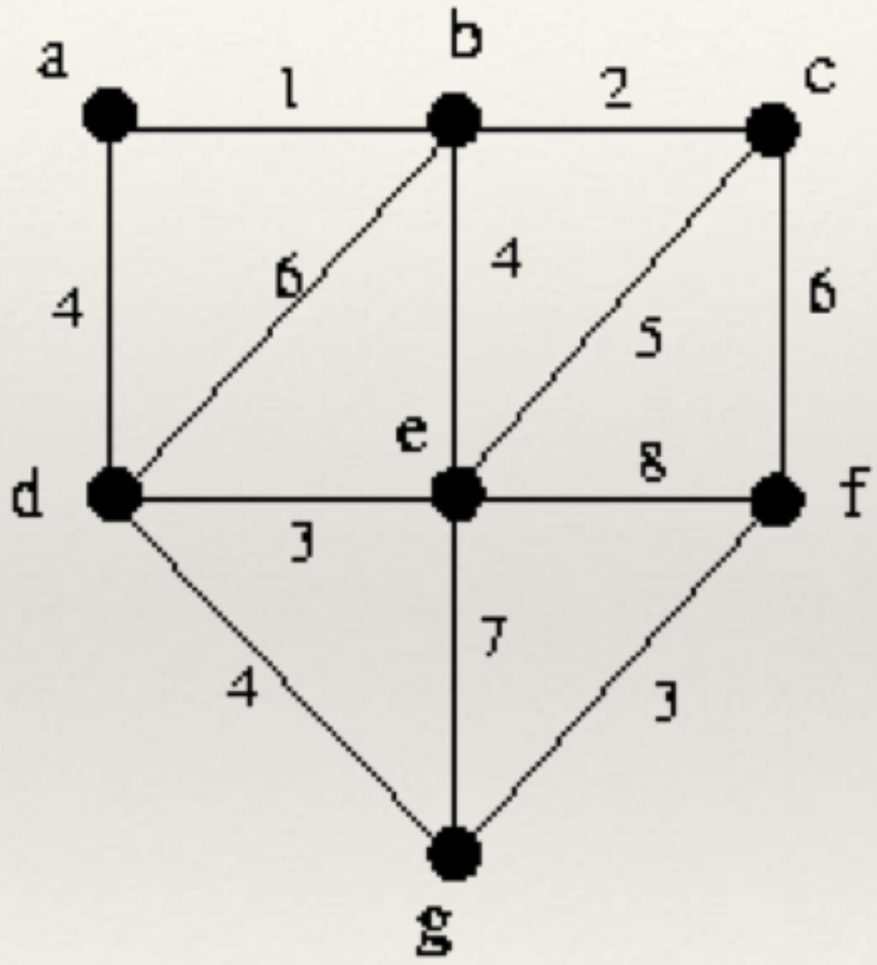
- $T(V,E')$

1. Ordene E em ordem ascendente de $c(i,j)$ e armazene em A.
2. $T \leftarrow \emptyset$.
3. Inicialize o conjunto de componentes. *(cada vértice $v \in V$ é uma componente)*
4. **Enquanto** $(|T| < |V|\text{-}1) \wedge (|A| \neq \emptyset)$:
    A. $(u,v) \leftarrow$ aresta $(u,v) \in A$, tal que $c(u,v)$ seja mínimo.
    B. $A \leftarrow A - (u,v)$.
    C. comp_$u \leftarrow$ **Find_Component**(u). *% índice da subárvore que contém u*
    D. comp_$v \leftarrow$ **Find_Component**(v). *% índice da subárvore que contém v*
    E. **Se** comp_$u \neq$ comp_v:
        I. **Merge**(comp_u,comp_v).
        II. $T \leftarrow T \cup \{(u,v)\}$.
5. **Retornar** T.

Estudo de complexidade:

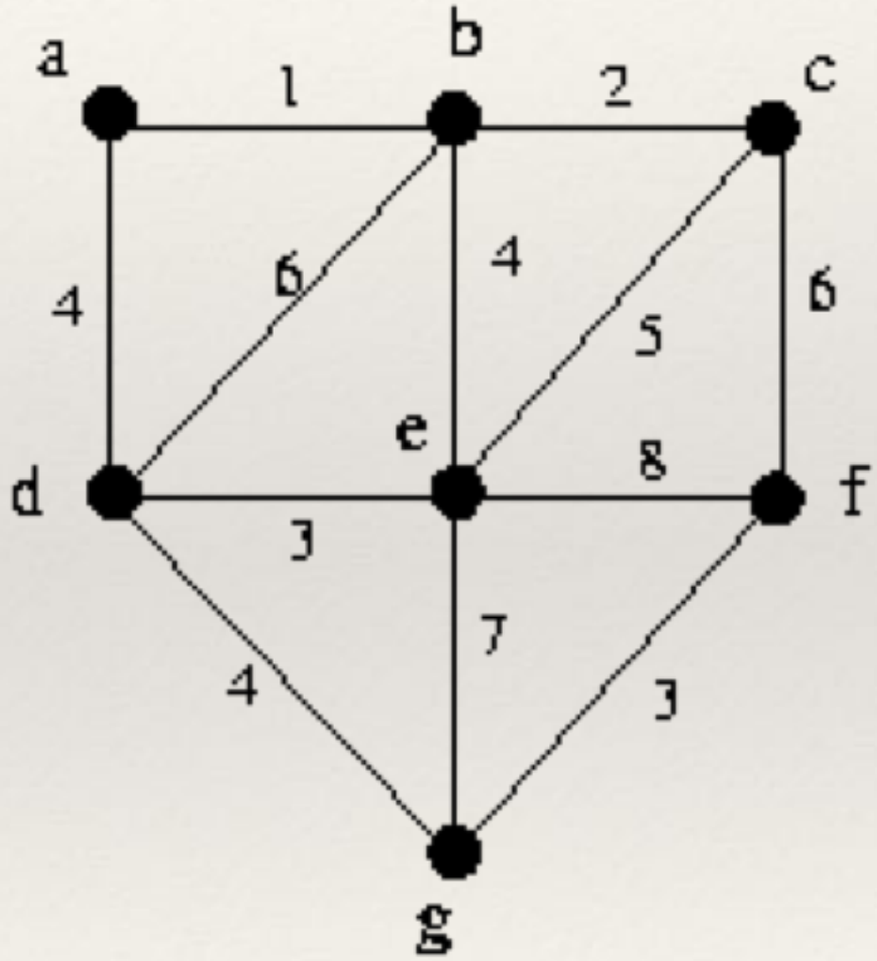
- Etapas:
  - $O(|E| \log |E|)$ para ordenar as arestas.
    - Como $(|V|-1) \leq |E| \leq (|V|(|V|-1)/2)$:
      - Complexidade: $O(|E| \log |V|)$.
  - $O(|E|)$ para inicializar o conjunto de componentes.
  - No pior caso, o número de chamadas da função find_component é 2x número de arestas no grafo.
    - Cada chamada exige um tempo em $O(\log |V|)$:
      - Complexidade: $O(|E| \log |V|)$.
  - O número de chamadas da função merge é exatamente $|E|-1$:
    - Complexidade: $O(|V| \log |V|)$.
  - $O(|E|)$ para o resto das operações.
- Complexidade global: $O(|E| \log |V|)$.

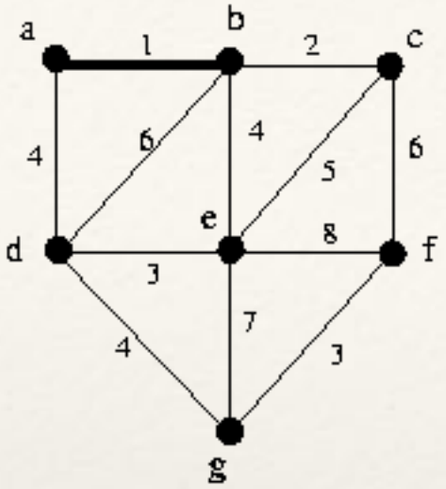
Exemplo



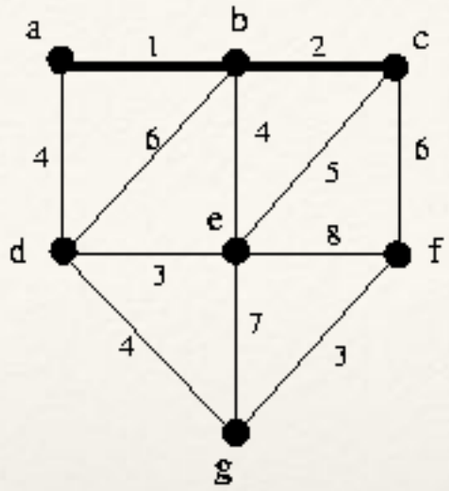
Exemplo

Arestas ordenadas: *(a,b),(b,c),(d,e),(f,g),(a,d),(b,e),(d,g),(c,e),(b,d),(c,f),(e,g),(e,f).*

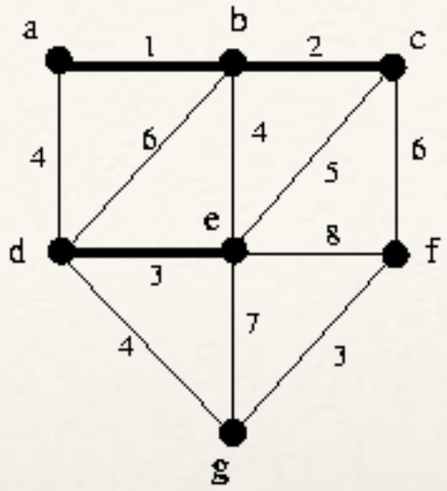



a
b
c
d
e
f
g
1
2
4
6
4
5
6
3
8
7
4
3

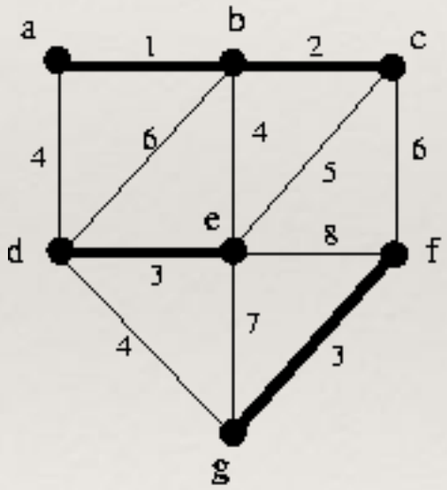
(a)


a
b
c
d
e
f
g
1
2
4
6
4
5
6
3
8
7
4
3

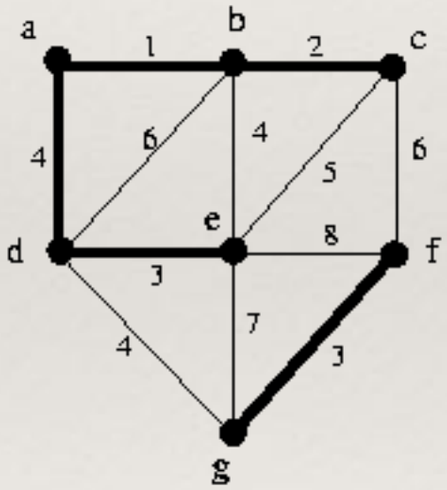
(b)


a
b
c
d
e
f
g
1
2
4
6
4
5
6
3
8
7
4
3

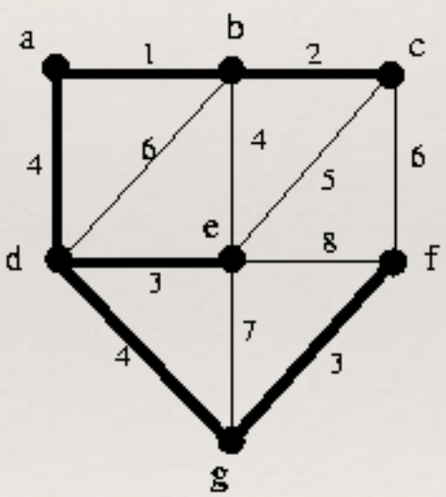
(c)


a
b
c
d
e
f
g
1
2
4
6
4
5
6
3
8
7
4
3

(d)


a
b
c
d
e
f
g
1
2
4
6
4
5
6
3
8
7
4
3

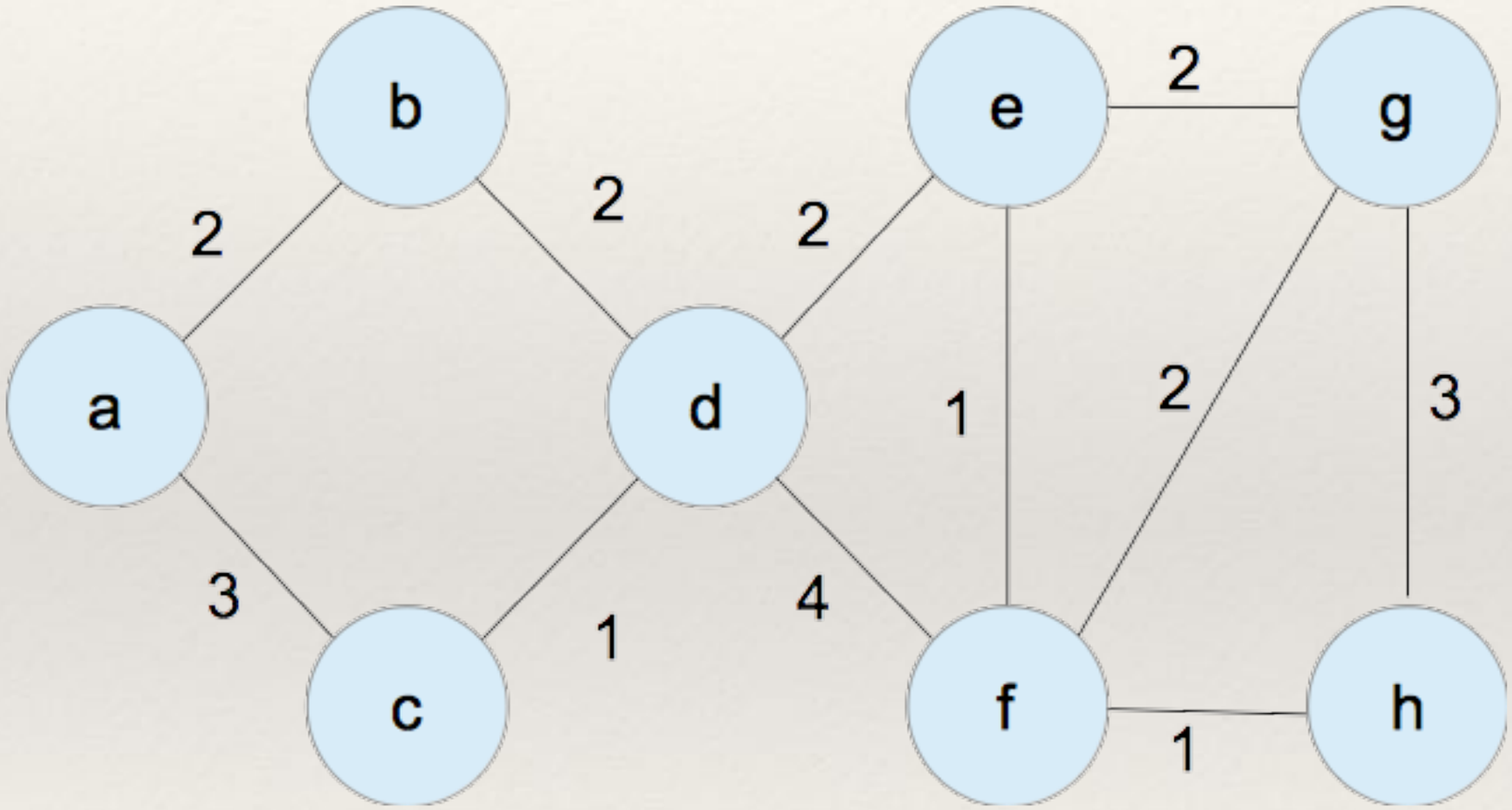
(e)


a
b
c
d
e
f
g
1
2
4
6
4
5
6
3
8
7
4
3

(f)

It.	*(u,v):*	*T:*
-	-	*{a} {b} {c} {d} {e} {f} {g}*
1)	*(a,b)*	*{a,b} {c} {d} {e} {f} {g}*
2)	*(b,c)*	*{a,b,c} {d} {e} {f} {g}*
3)	*(d,e)*	*{a,b,c} {d,e} {f} {g}*
4)	*(f,g)*	*{a,b,c} {d,e} {f,g}*
5)	*(a,d)*	*{a,b,c,d,e} {f,g}*
6)	*(b,e)*	**Rejeitada**
7)	*(d,g)*	*{a,b,c,d,e,f,g}*

Exercício



Referências

- [Arenales et al, 2007] M. Arenales; V. Armentano; R. Morabito; H. Yanasse. **Pesquisa Operacional para Cursos de Engenharia**, Editora Campus / Elsevier, 2007.
- [Belfiore et al, 2013] P. Belfiore; L. P. Fávero. Pesquisa Operacional para Cursos de Engenharia, Editora Campus / Elsevier, 2013.
- [Goldbarg et al, 2005] M. C. Goldbarg; H. P. Luna. **Otimização Combinatória e Programação Linear - Modelos e Algoritmos**, 2a ed., Editora Campus / Elsevier, 2005.
- [Hillier et al, 2013] F. S. Hillier; G. J. Lieberman. **Introdução à Pesquisa Operacional**, 9a ed., Editora Mc Graw Hill, 2013.