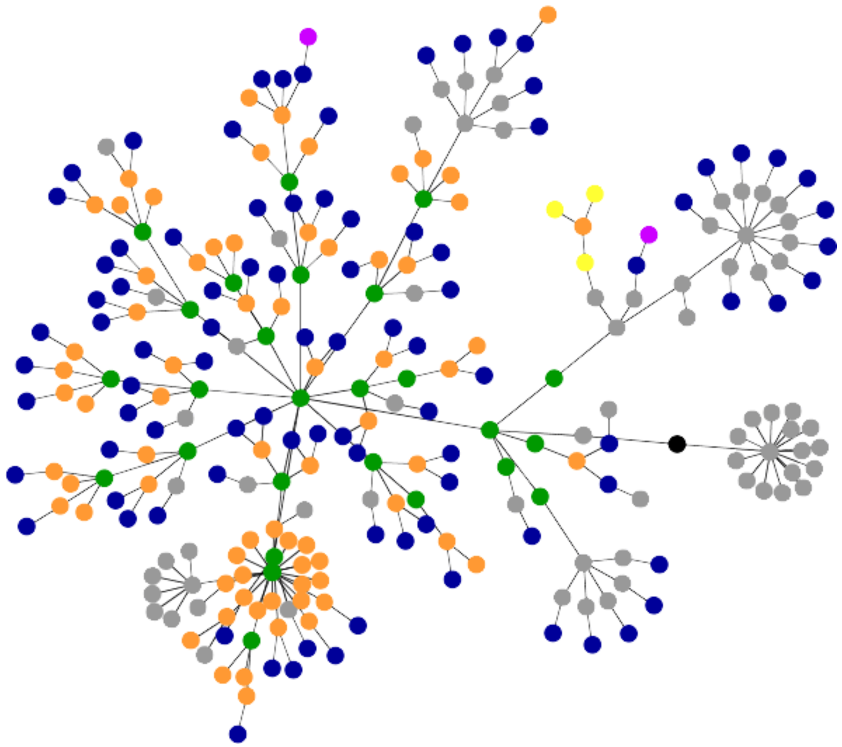
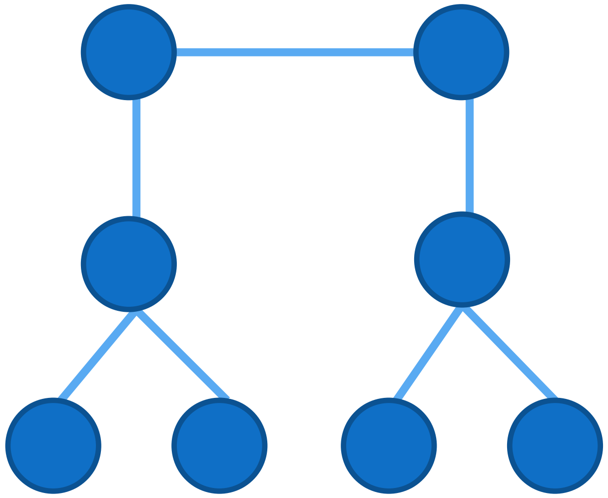


Árvores

Árvore

Grafo conexo e acíclico

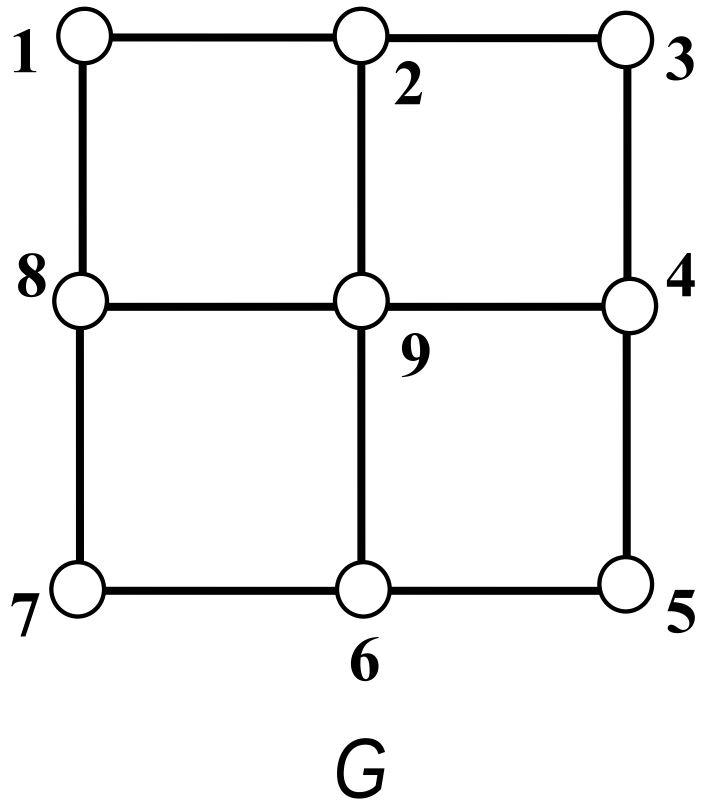


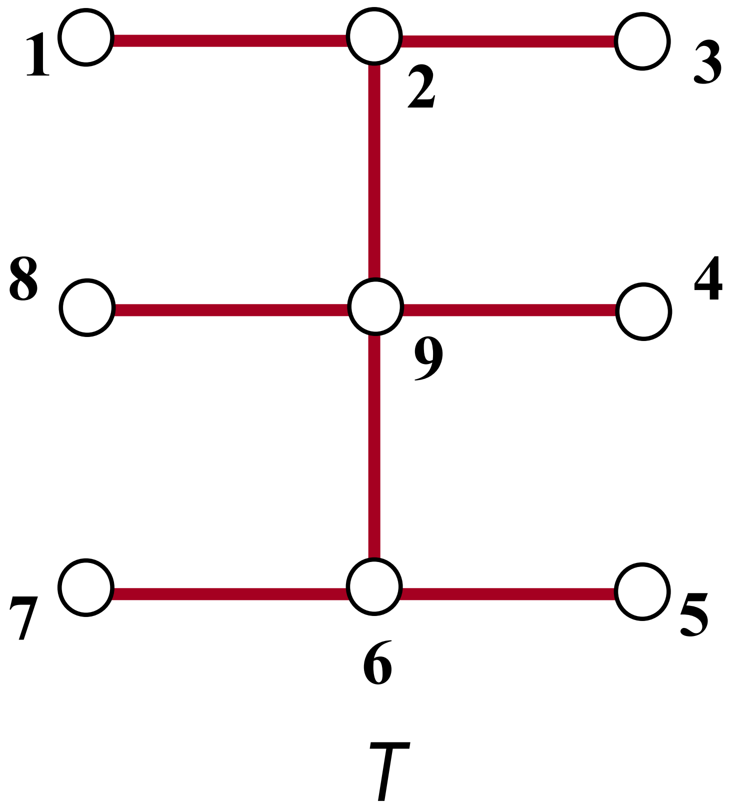
Teorema de Caracterização de Árvores

1. *G* é uma árvore
2. Existe exatamente um caminho entre cada par de vértices de *G*
3. *G* é conexo e m=n-1 (m=número de arestas, n=número de vértices)
4. *G* é acíclico e m=n-1
5. *G* é acíclico e se quaisquer dois vértices não adjacentes de *G* forem conectados por uma aresta, então o grafo resultante conterá exatamente um ciclo.

Árvore Geradora

Uma ***Árvore Geradora*** **de um grafo *G* = (*V*,*E*) é um subgrafo gerador *T*=(*V*,E_T) conexo e acíclico.**





Árvores Geradoras

Considere G' um subgrafo gerador de G. Então as seguintes proposições são equivalentes:

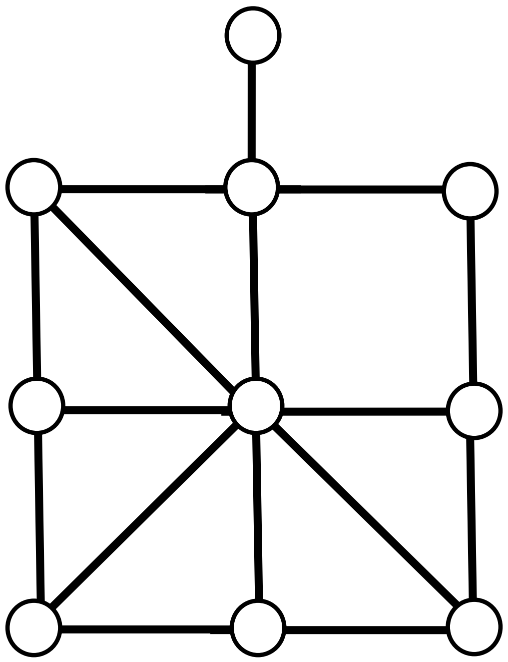
1. ***G'* é uma árvore geradora de G**
2. **Existe exatamente um caminho entre cada par de vértices de *G'***
3. ***G'* é conexo e m=n-1**
4. ***G'* é acíclico e m=n-1**
5. ***G'* é acíclico e se quaisquer dois vértices não adjacentes de *G* forem conectados por uma aresta, então o grafo resultante conterá exatamente um ciclo.**

Árvores Geradoras

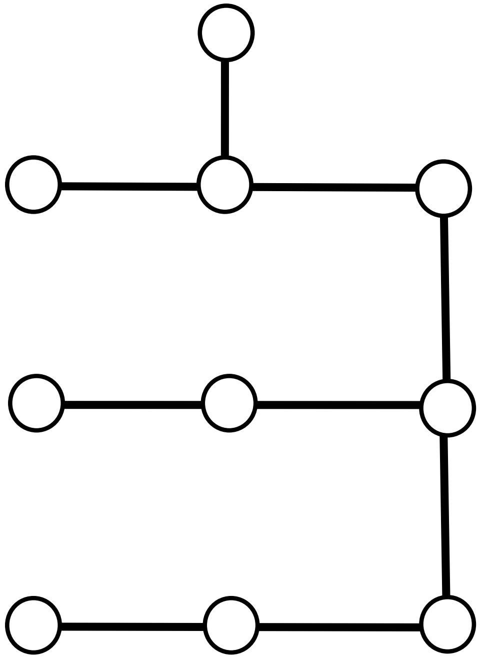
Teorema: Um grafo G é conexo se e somente se possui árvore geradora

Co-Árvore Geradora

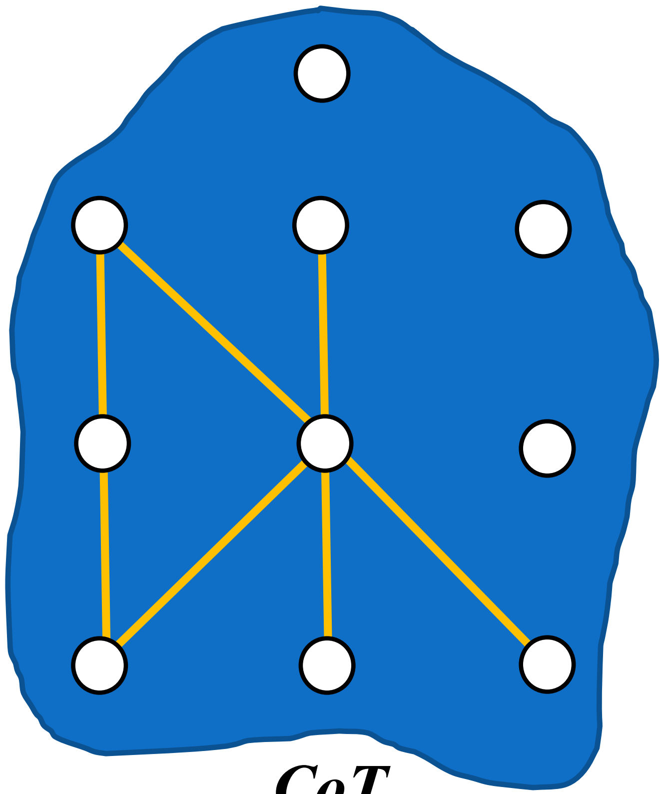
Dado G=(V,E) e T =(V,E_T), uma árvore geradora de G, o grafo complementar de T em relação a G é denominado *Co-Árvore Geradora*.



G



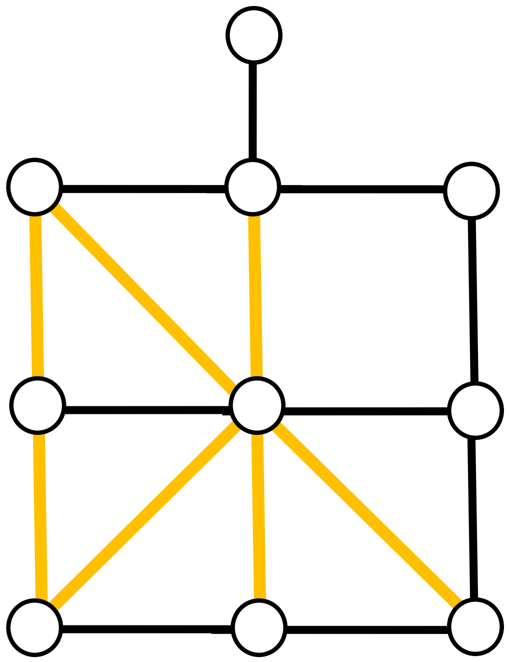
T



CoT

Co-Árvore Geradora

As arestas de uma co-árvore geradora de *G* são chamadas *elos*.

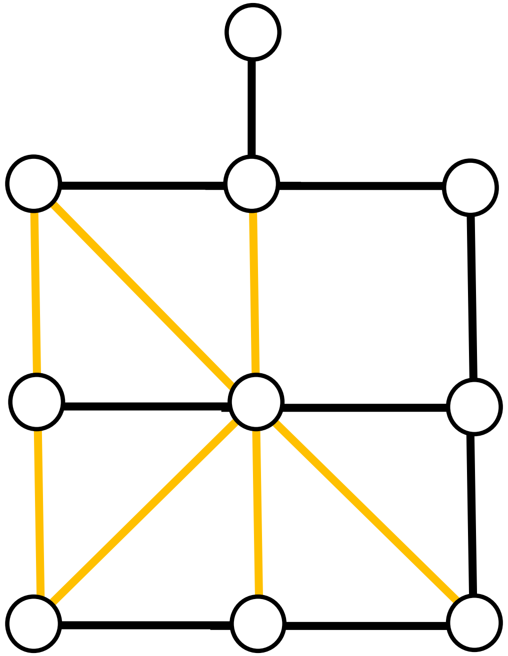


Co-Árvore Geradora

As arestas de uma co-árvore geradora de *G* são chamadas *elos*.

***número ciclomático de um grafo* = número de elos**

número ciclomático(*G*) = número de arestas de *G* – número de arestas da árvore geradora



número ciclomático(*G*) = $m - (n - 1)$

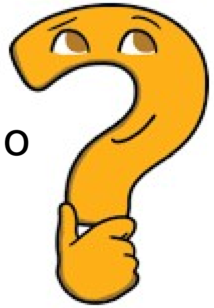
número ciclomático(*G*) = $m - n + 1$

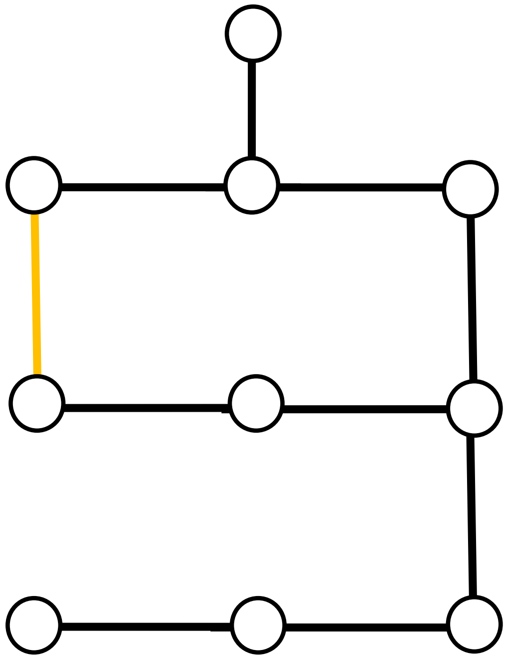
Ciclos Fundamentais

A adição de um *elo* a uma *árvore* produz um *único ciclo* no grafo resultante.

Tais ciclos são denominados *ciclos fundamentais*.

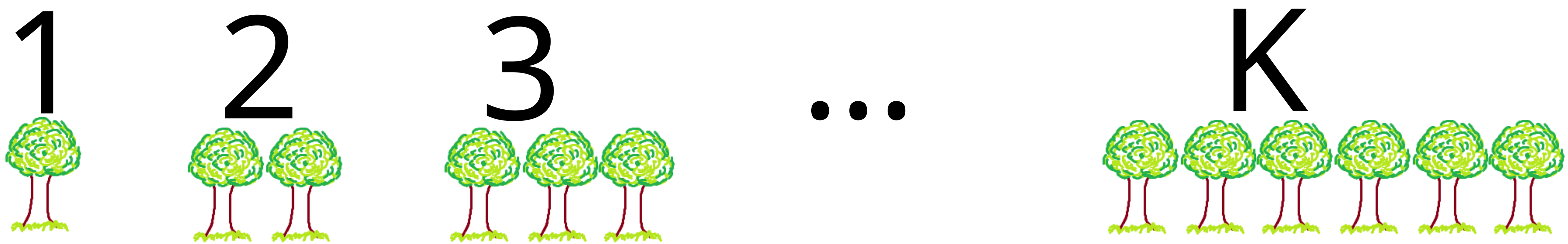
Quantos ciclos fundamentais existem no grafo?





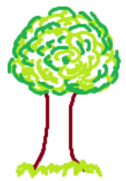
k-árvore

Uma ***k-árvore*** **é um grafo acíclico que contém *k* componentes.**



k-árvore

Uma ***k-árvore*** é um grafo acíclico que contém *k* componentes conexos.



Uma ***1-árvore*** é uma árvore.

Se uma *k*-árvore é um subgrafo gerador de um grafo *G*, então ela é dita ***k-árvore geradora***.

Rank e Nulidade

G=(V,E), onde $|V| = n$ e $|E| = m$. G possui k componentes conexos.

$$rank(G) = n - k$$

$$nulidade(G) = m - n + k$$

Obs. rank(G) + nulidade(G) = m

Problema do Conector

Problema. Uma rede ferroviária conectando *n* cidades será construída. Dado o custo c_{ij} de construir um trecho de ferrovia que liga as cidades *i* e *j*, encontre uma configuração para esta rede tal que o custo total de construção seja minimizado.

custo c_{ij}

Problema do Conector

Problema. Uma rede ferroviária conectando *n* cidades será construída. Dado o custo c_{ij} de construir um trecho de ferrovia que liga as cidades *i* e *j*, encontre uma configuração para esta rede tal que o custo total de construção seja minimizado.

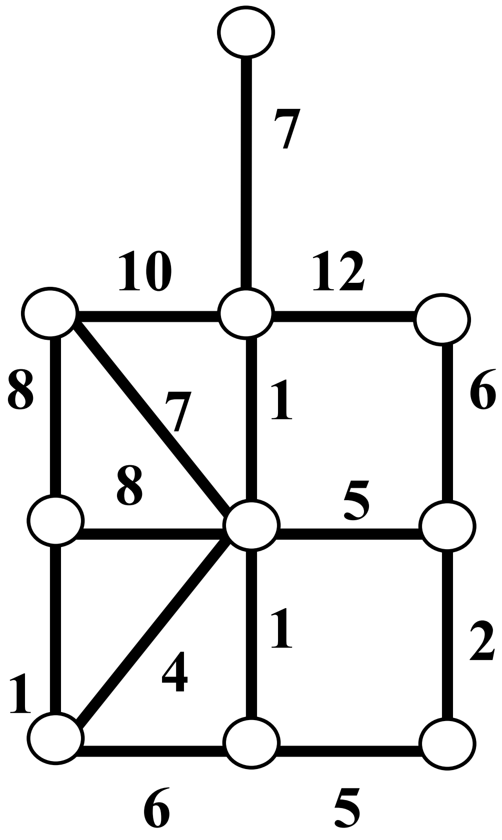
Modele este problema em grafos.

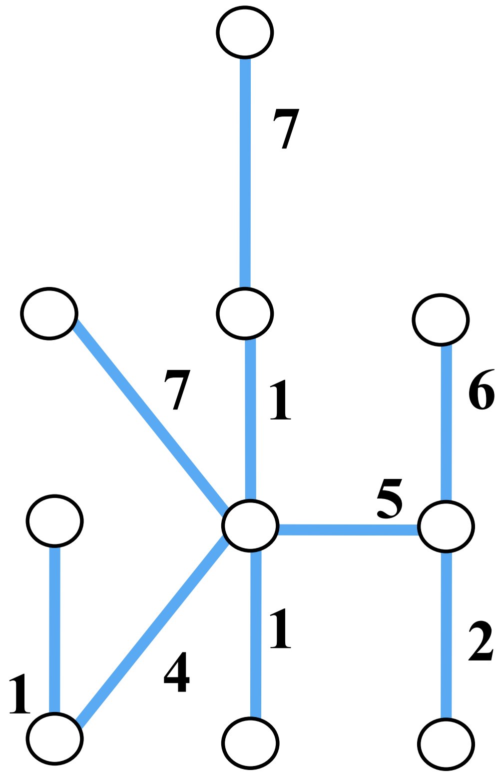
1. Considere a situação em que $c_{ij} = 1$ para todos os pares (i,j)
2. Considere a situação em que $c_{ij} \geq 1$ para todos os pares (i,j)

Apresente um algoritmo para solucioná-lo.

Árvore Geradora Mínima

Árvore geradora de menor custo, dentre todas as possíveis em um grafo ***G* ponderado em arestas.**





Custo = 34

Árvore Geradora Mínima

Um algoritmo para árvore geradora mínima

Como você faria?

Qual seria a complexidade?

Árvore Geradora Mínima

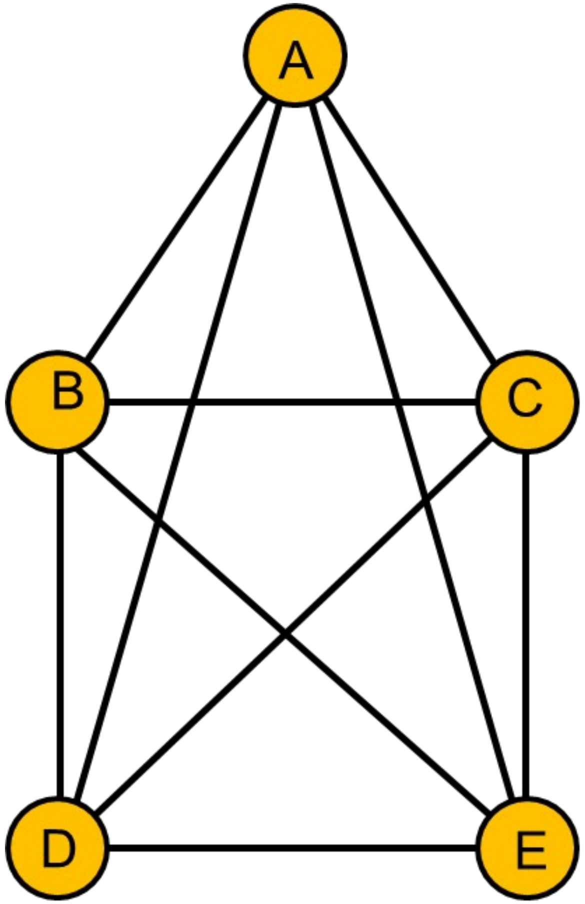
Como Kruskal (1956) pensou?

Ordena as *m* arestas do grafo (*ordem crescente*).
Pega de uma em uma para incluir na árvore, desde de que não forme ciclo.
Pare quando tiver incluído *n*-1 arestas.

Kruskal, J. B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, *7*(1), 48-50.

Árvore Geradora Mínima

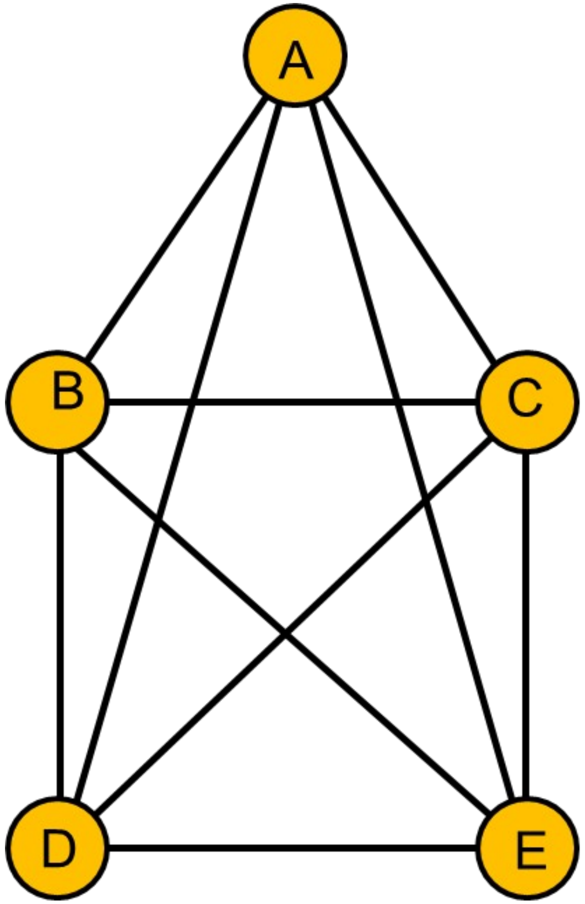
Método de Kruskal



Trecho	C_{IJ}
A,B	15
A,D	22
B,C	0
A,E	86
A,C	85
B,D	20
B,E	86
C,D	17
C,E	90
D,E	23

Árvore Geradora Mínima

Método de Kruskal



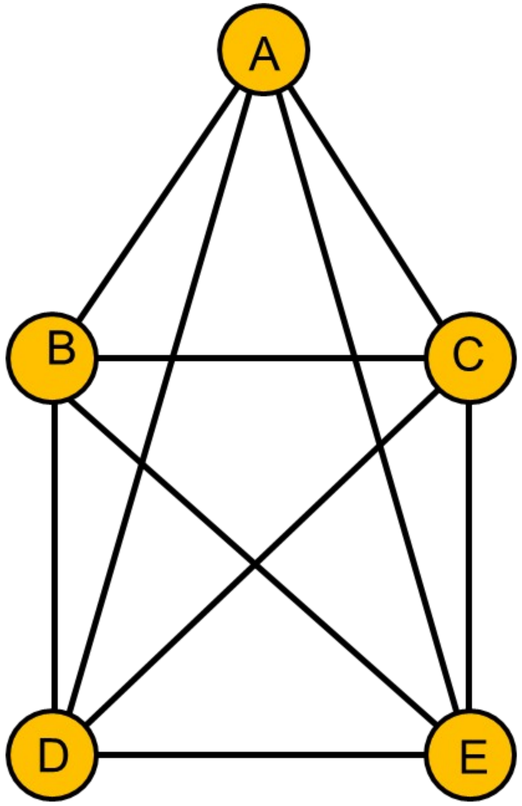
Trecho	C_{IJ}
A,B	15
A,D	22
B,C	0
A,E	86
A,C	85
B,D	20
B,E	86
C,D	17
C,E	90
D,E	23

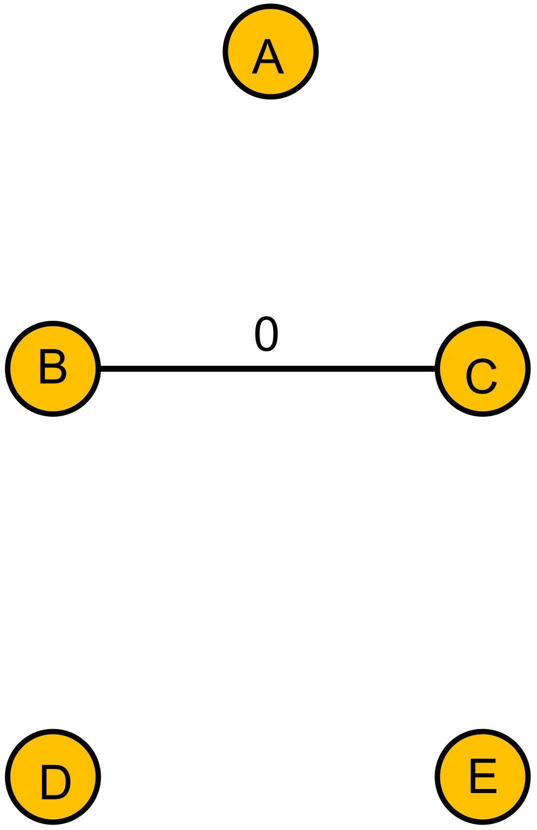
Ordena as arestas

Trecho	C_{IJ}
B,C	0
A,B	15
C,D	17
B,D	20
A,D	22
D,E	23
A,C	85
B,E	86
A,C	85
C,E	90

Árvore Geradora Mínima

Método de Kruskal

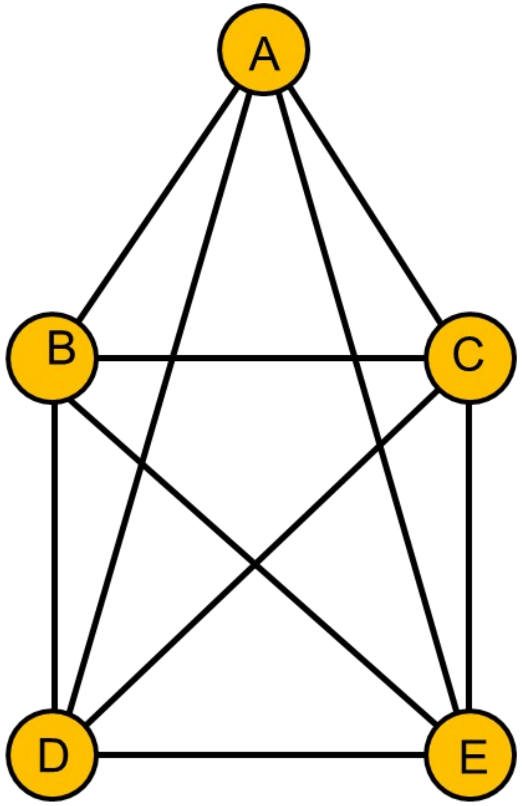


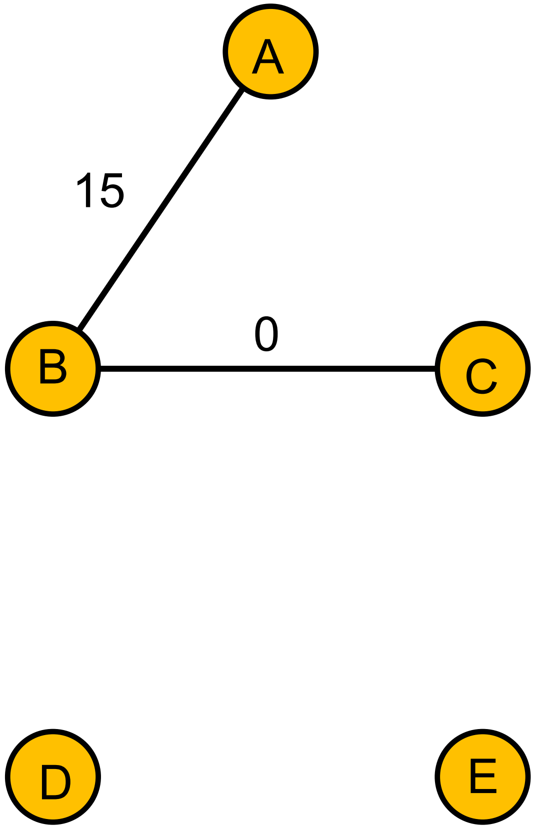


Trecho	C_{IJ}
~~B,C~~	~~0~~
A,B	15
C,D	17
B,D	20
A,D	22
D,E	23
A,C	85
B,E	86
A,C	85
C,E	90

Árvore Geradora Mínima

Método de Kruskal

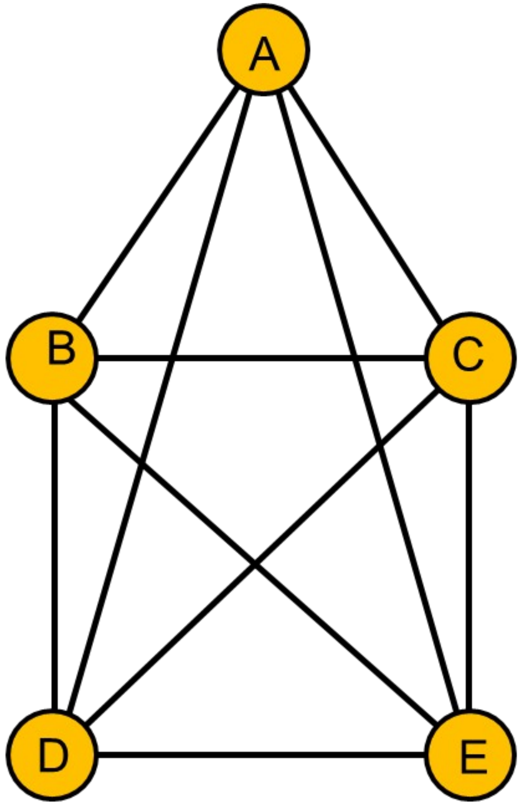


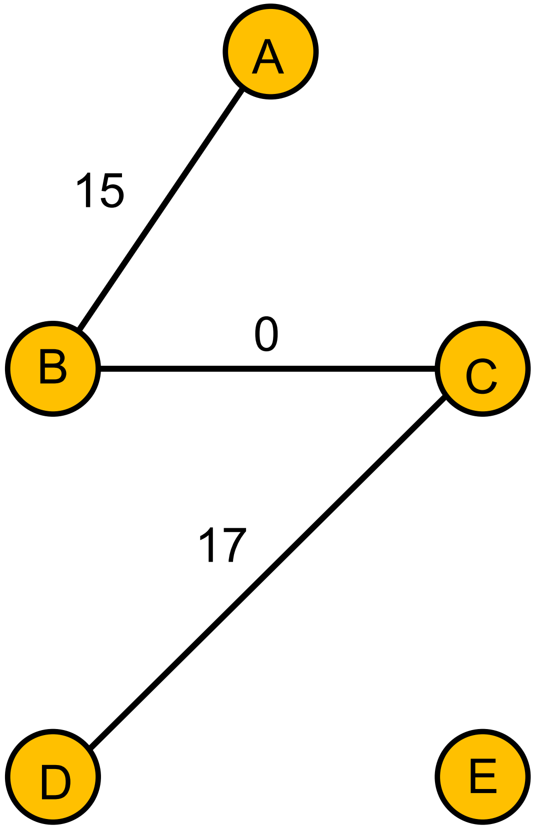


Trecho	C_{IJ}
~~B,C~~	~~0~~
~~A,B~~	~~15~~
C,D	17
B,D	20
A,D	22
D,E	23
A,C	85
B,E	86
A,C	85
C,E	90

Árvore Geradora Mínima

Método de Kruskal

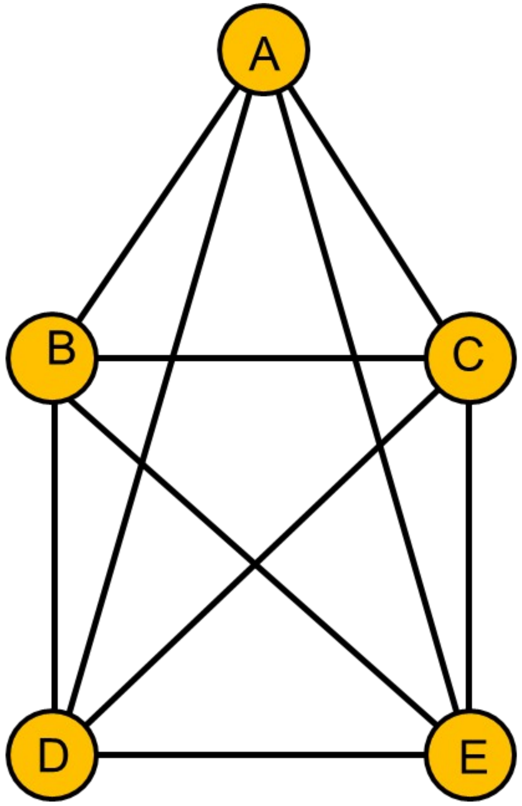


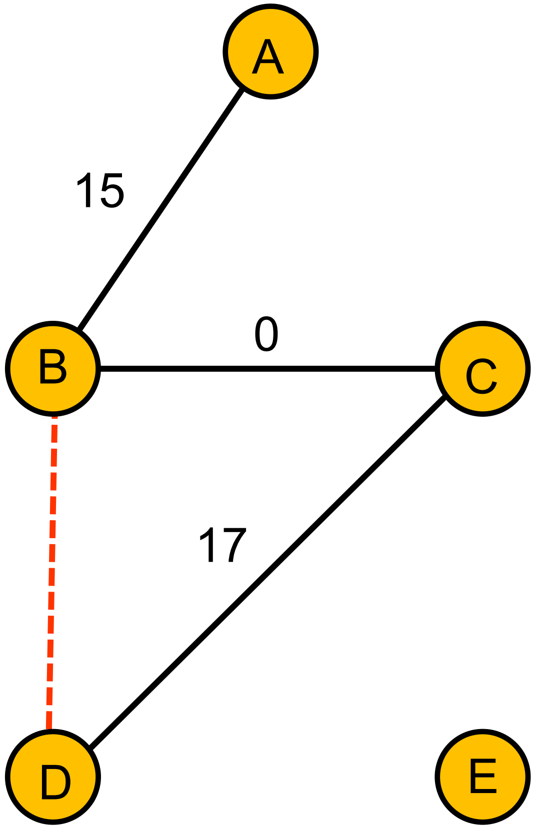


Trecho	C_{IJ}
~~B,C~~	~~0~~
~~A,B~~	~~15~~
~~C,D~~	~~17~~
B,D	20
A,D	22
D,E	23
A,C	85
B,E	86
A,C	85
C,E	90

Árvore Geradora Mínima

Método de Kruskal

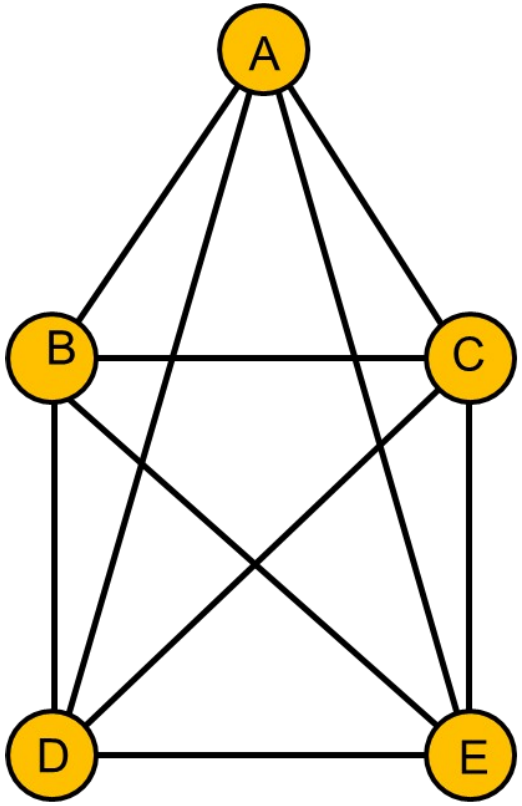


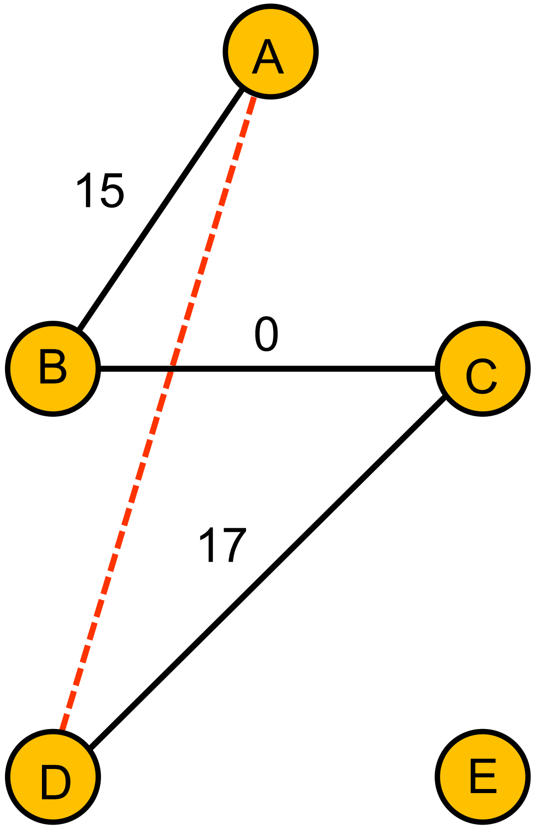


Trecho	C_{IJ}
~~B,C~~	~~0~~
~~A,B~~	~~15~~
~~C,D~~	~~17~~
~~B,D~~	~~20~~
A,D	22
D,E	23
A,C	85
B,E	86
A,C	85
C,E	90

Árvore Geradora Mínima

Método de Kruskal

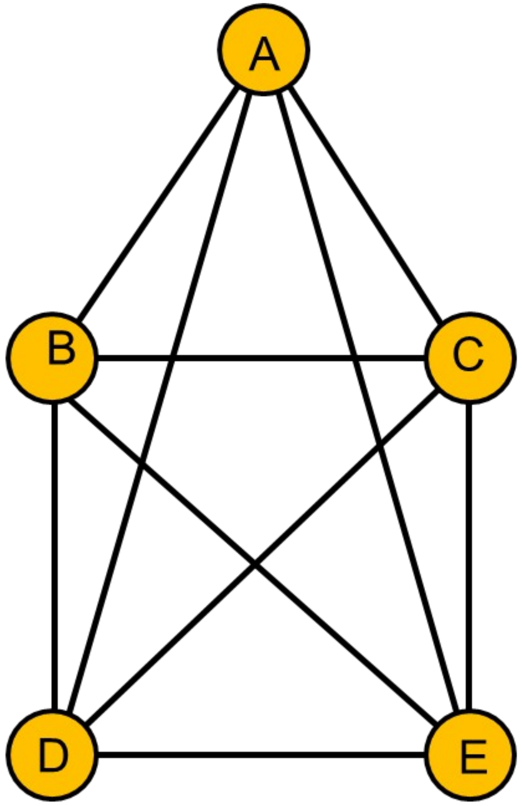


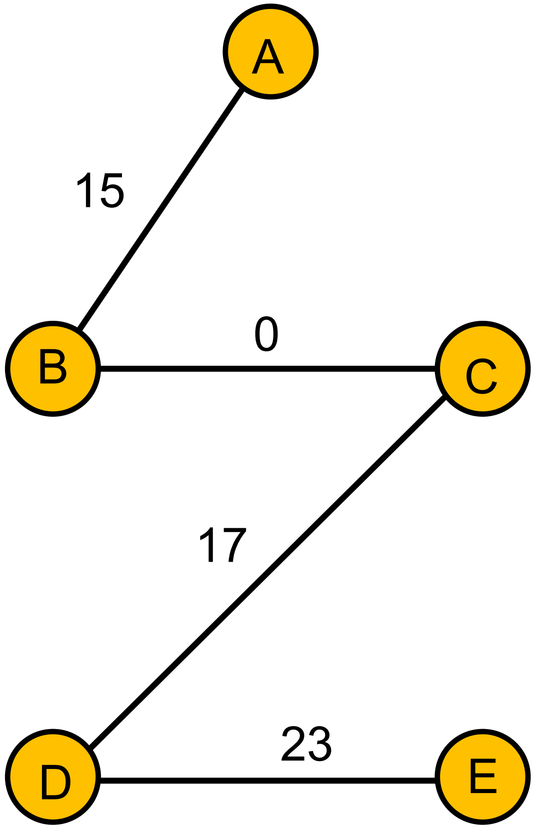


Trecho	C_{IJ}
~~B,C~~	~~0~~
~~A,B~~	~~15~~
~~C,D~~	~~17~~
~~B,D~~	~~20~~
~~A,D~~	~~22~~
D,E	23
A,C	85
B,E	86
A,C	85
C,E	90

Árvore Geradora Mínima

Método de Kruskal

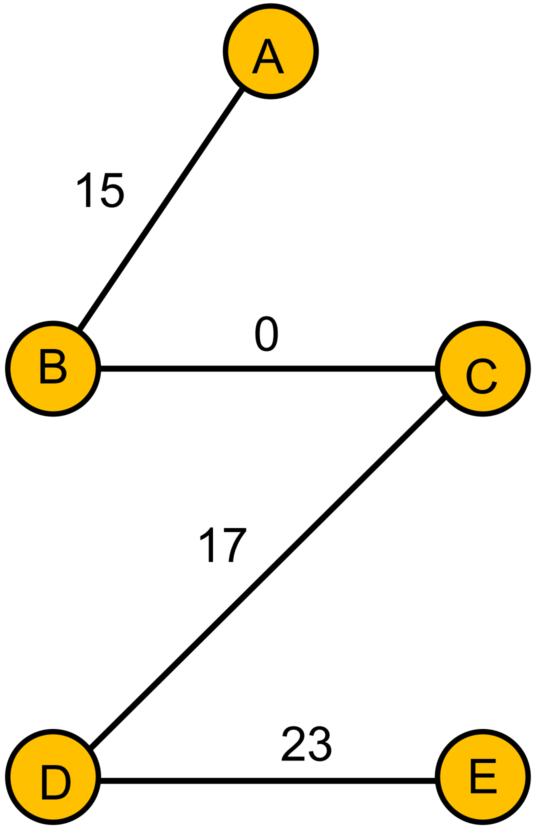




Trecho	C_{IJ}
~~B,C~~	~~0~~
~~A,B~~	~~15~~
~~C,D~~	~~17~~
~~B,D~~	~~20~~
~~A,D~~	~~22~~
~~D,E~~	~~23~~
A,C	85
B,E	86
A,C	85
C,E	90

Árvore Geradora Mínima

Método de Kruskal



Custo = 55

Algoritmo de Kruskal

O método de Kruskal sempre gera a árvore geradora mínima?
(Prova que o algoritmo é correto)

Qual a complexidade deste método?

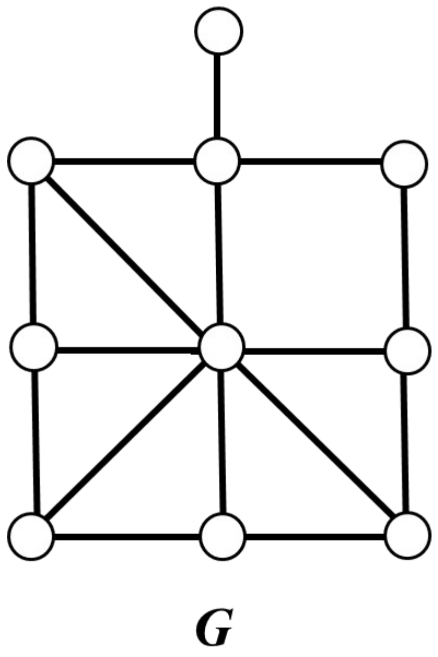
Árvore Geradora Mínima

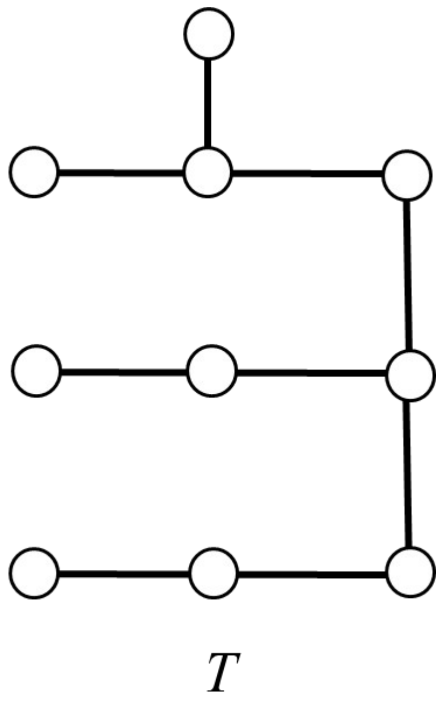
Antes de responder estas perguntas, vejamos alguns resultados teóricos sobre as árvores geradoras mínimas.

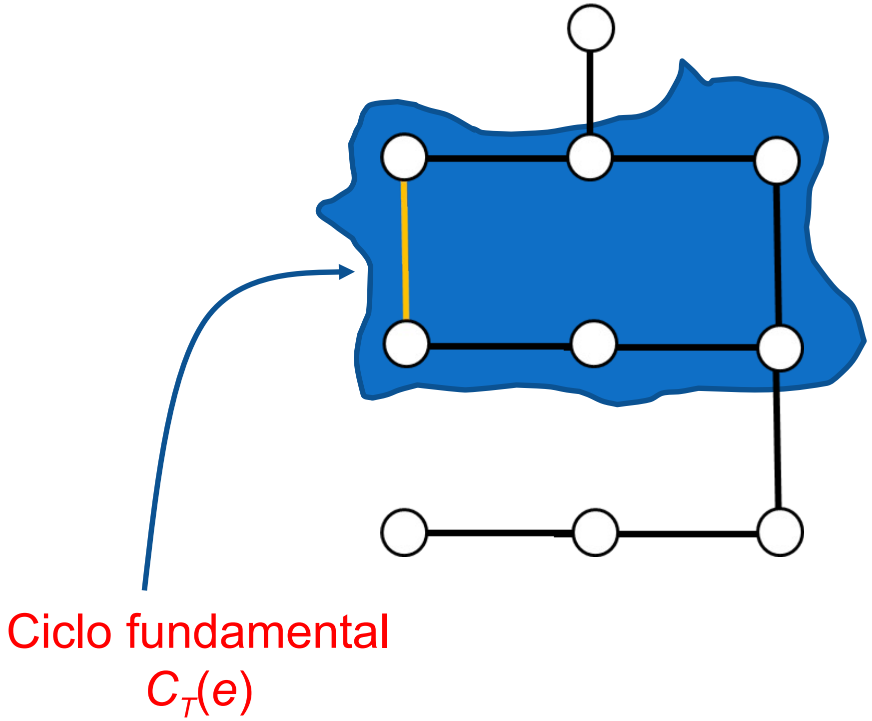
Veremos o algoritmo de Kruskal e outros adiante.

Árvore Geradora Mínima

Dado um grafo G conexo e uma árvore geradora T de G, considere $C_T(e)$ o ciclo fundamental obtido pela adição do elo e à T e $w(e)$ o peso da aresta e.

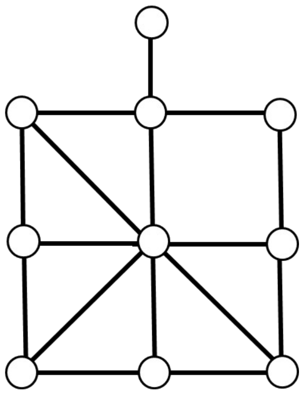




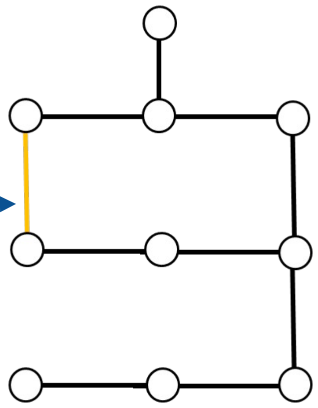


Árvore Geradora Mínima

Dado um grafo G conexo e uma árvore geradora T de G, considere $C_T(e)$ o ciclo fundamental obtido pela adição do elo e à T e $w(e)$ o peso da aresta e.



G



Teorema. Uma árvore geradora T de G é mínima se e somente se para cada aresta e em $G \backslash T$, tem-se $w(e) \geq w(f)$ para todas as arestas f em $C_T(e)$.

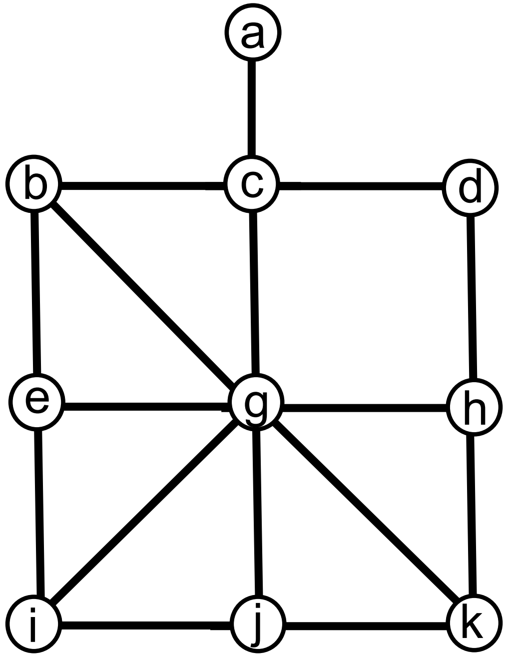
elo

Ciclo fundamental
$C_T(e)$

Árvore Geradora Mínima

Considere **$G = (V,E)$.**

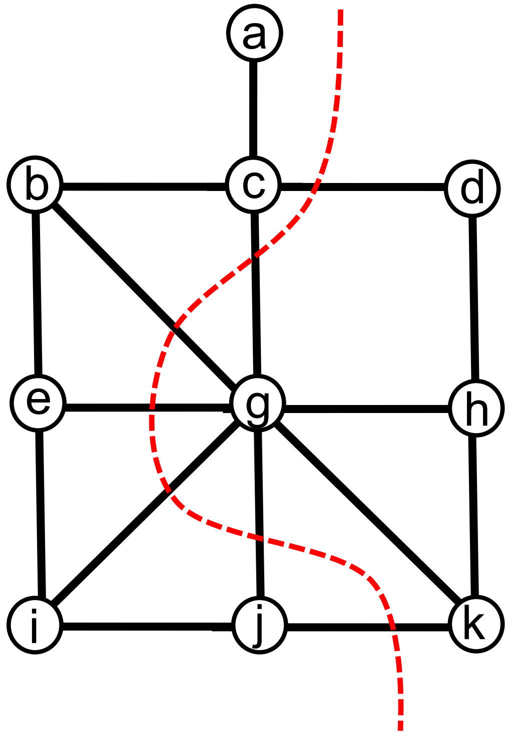
Um corte é uma partição $S = X \cup X'$ de $\boldsymbol{V}$ em 2 subconjuntos não vazios disjuntos.



Exemplo de corte:

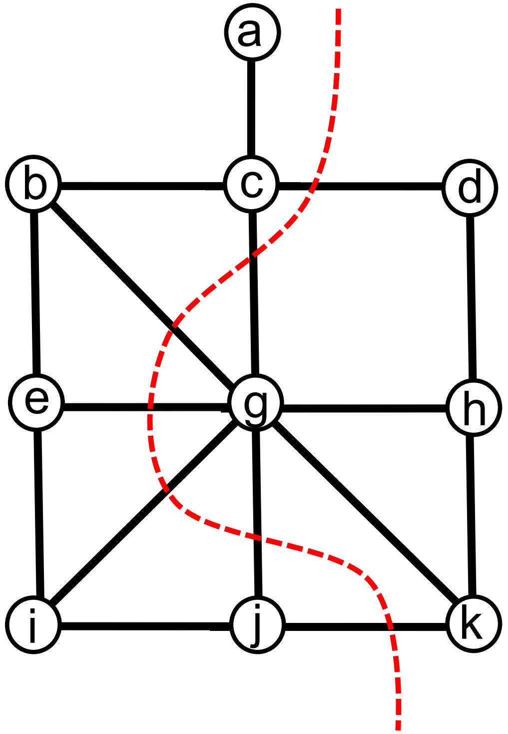
X = {a, b, c, e, i, j}

X’ = {d, g, h, k}



Árvore Geradora Mínima

Considere $E(S)$ o conjunto de arestas que ligam um vértice de X a um vértice em X'. Um conjunto de tais arestas será chamado de co-ciclo.



X = {a, b, c, e, i, j}
X' = {d, g, h, k}

Co-ciclo

$E(S)$ = {(b,g), (c,d), (c,g), (e,g), (i,g), (g,j), (j,k)}

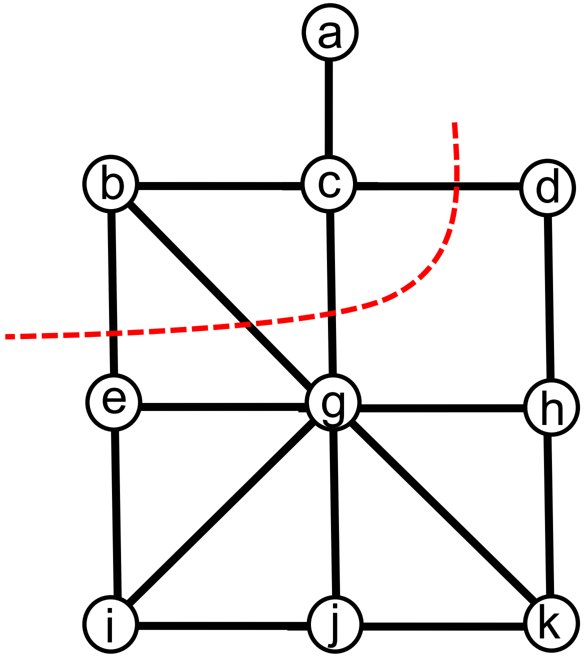
Árvore Geradora Mínima

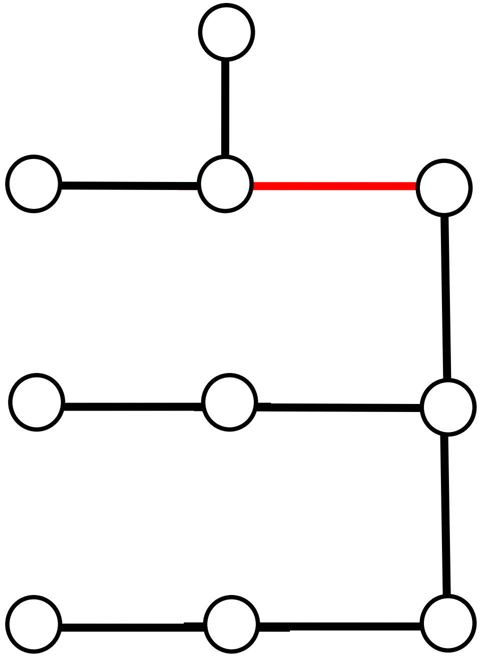
Lema. Seja G um grafo conexo e T uma árvore geradora de G. Para cada aresta e de T, existe exatamente um corte $S_T(e)$ de G tal que e é a única aresta que T tem em comum com o co-ciclo correspondente $E(S_T(e))$.

$e = (c,d)$

Co-ciclo

$E(S_T(e)) = \{(b,e), (b,g), (c,d), (c,g),\}$





Prova: Exercício

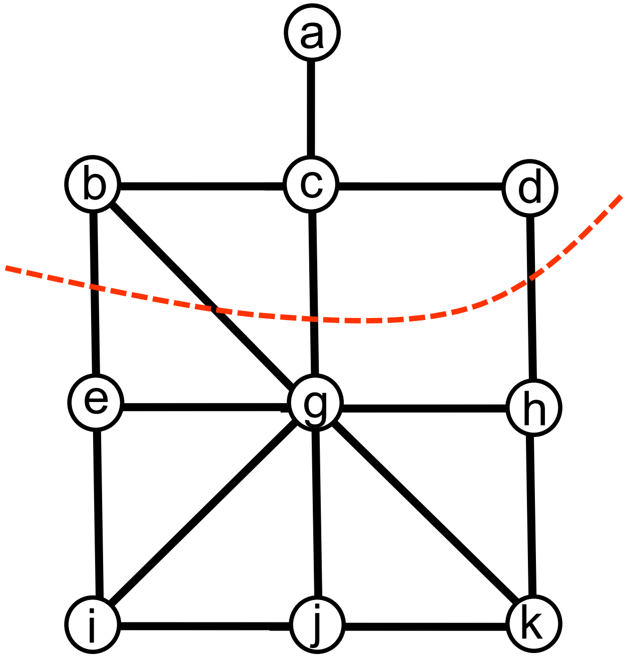
Árvore Geradora Mínima

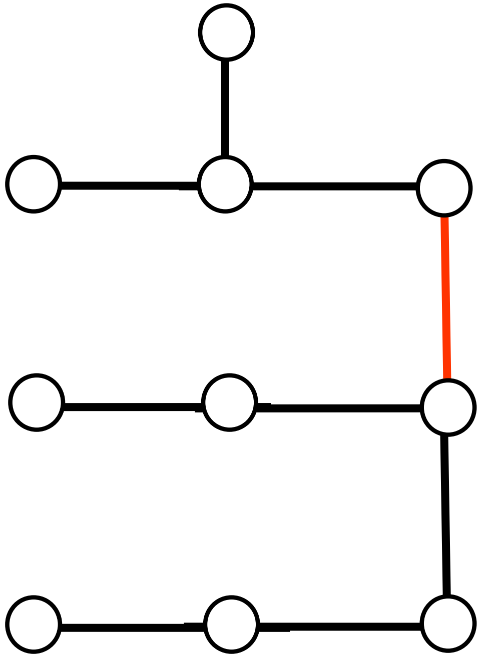
Teorema. Considere G um grafo conexo. Uma árvore geradora T de G é mínima se e somente se para cada aresta e de T, temos $w(e) \leq w(f)$ para cada aresta f em $E(S_T(e))$.

$e = (d,h)$

Co-ciclo

$E(S_T(e)) = \{(b,e), (b,g), (c,g), (d,h)\}$





Prova: Exercício

Árvore Geradora Mínima

Existe algoritmo polinomial para o problema da determinação da *árvore geradora mínima* de um grafo *G*.

Algoritmos clássicos:

- Kruskal
- Prim
- Borüvka

Algoritmo de Kruskal

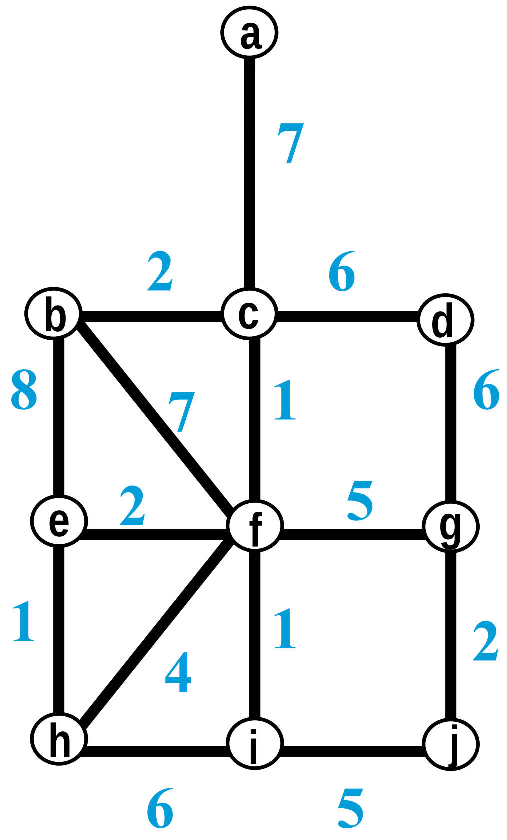
Dados G =(N,M), D =[d_{ij}] a matriz de pesos das arestas de G.

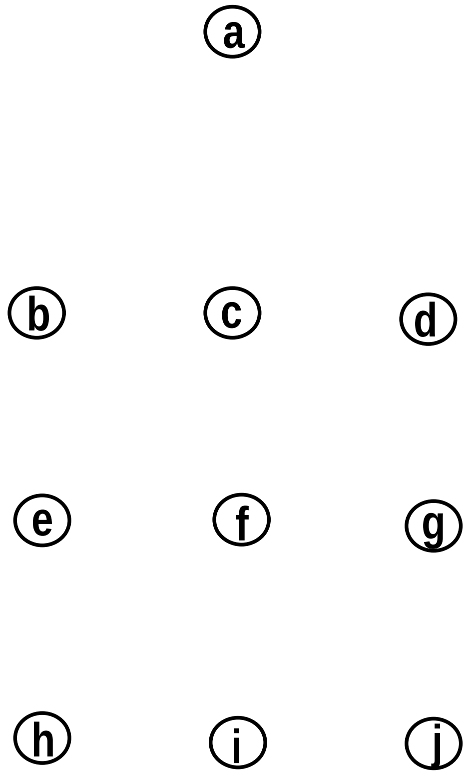
Ordene as arestas de G em ordem não decrescente dos pesos d_{ij} no vetor H = (h_i), i =1,2,...,m

T$\leftarrow$ {h_1}
$i \leftarrow 2$
Enquanto |T| < n-1 **faça**
  Se T $\cup$ h_i é um grafo acíclico **então**
    T $\leftarrow$ T $\cup$ {h_i}
    $i \leftarrow i + 1$
   Caso Contrário
    $i \leftarrow i + 1$
  Escrever T {arestas da árvore geradora mínima}

Algoritmo de Kruskal

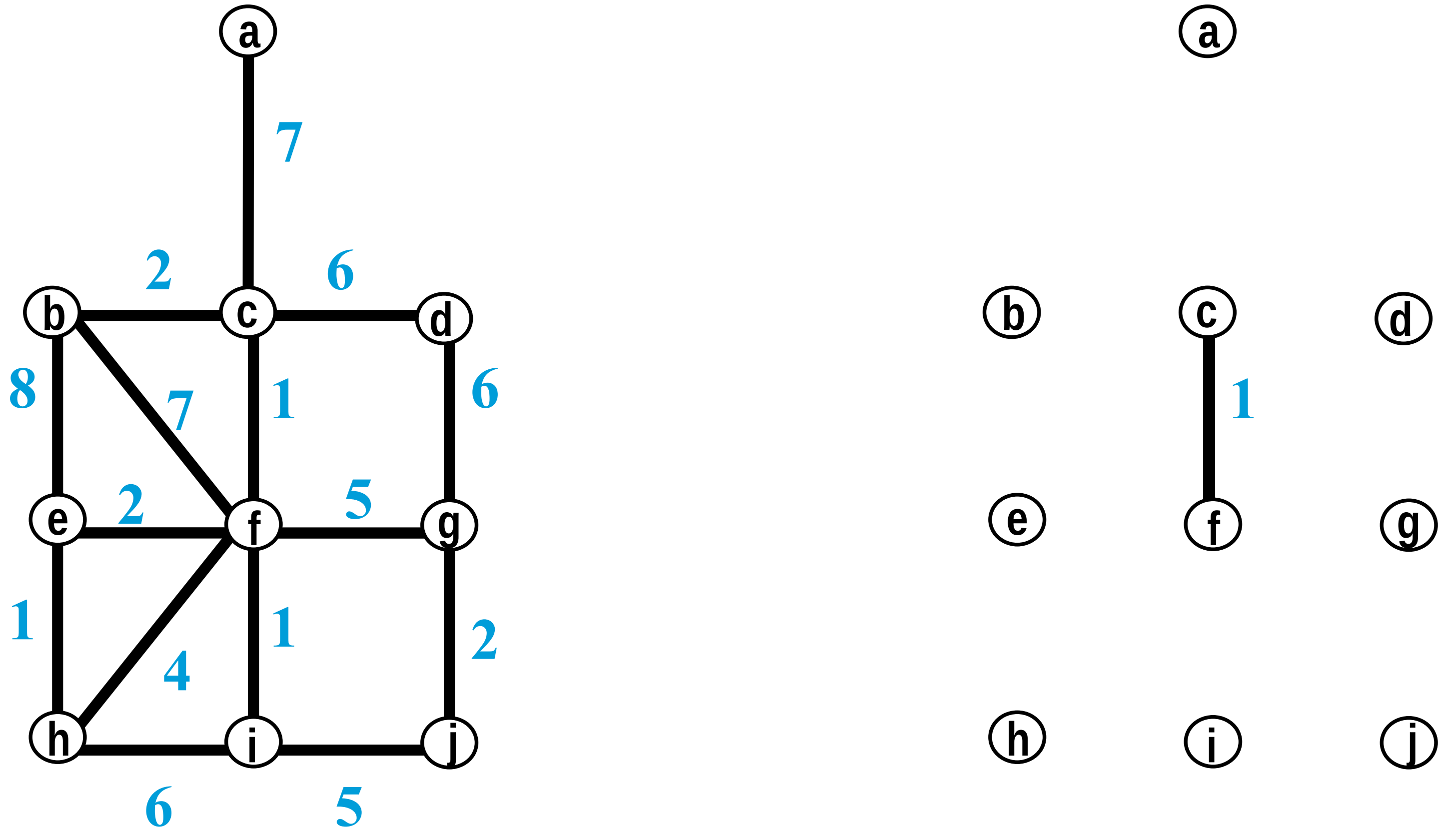
(c,f), (e,h), (f,i), (b,c), (e,f), (g,j), (f,h), (f,g), (i,j), (c,d), (d,g), (h,i), (a,c), (b,f), (b,e)





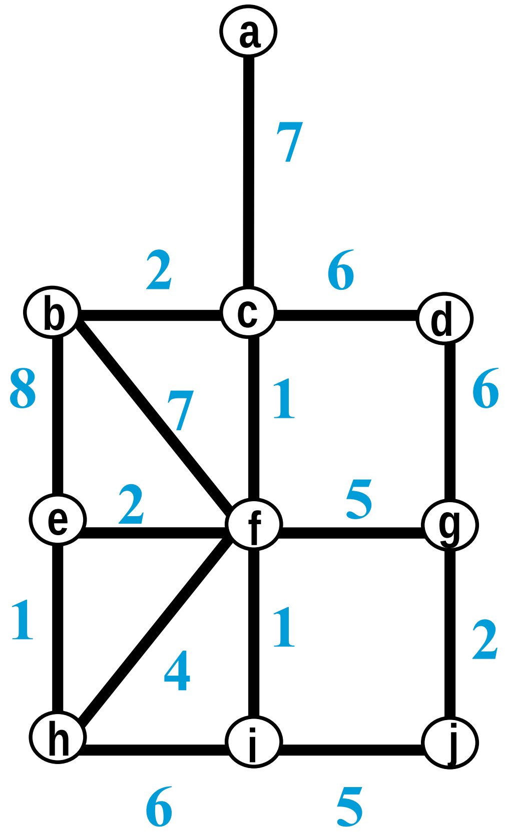
Algoritmo de Kruskal

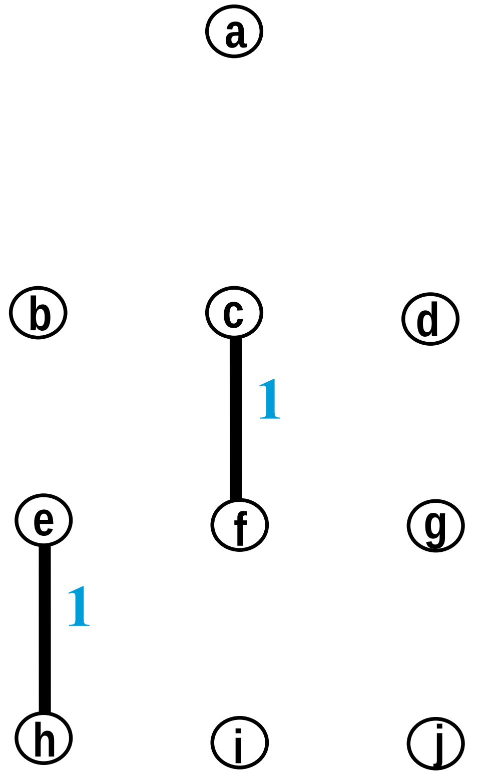
~~(c,f)~~, (e,h), (f,i), (b,c), (e,f), (g,j), (f,h), (f,g), (i,j), (c,d), (d,g), (h,i), (a,c), (b,f), (b,e)



Algoritmo de Kruskal

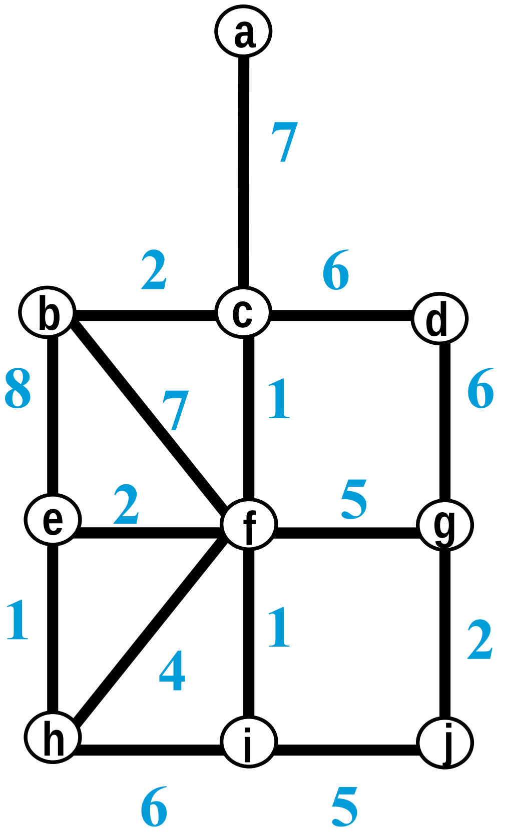
~~(c,f)~~, ~~(e,h)~~, (f,i), (b,c), (e,f), (g,j), (f,h), (f,g), (i,j), (c,d), (d,g), (h,i), (a,c), (b,f), (b,e)

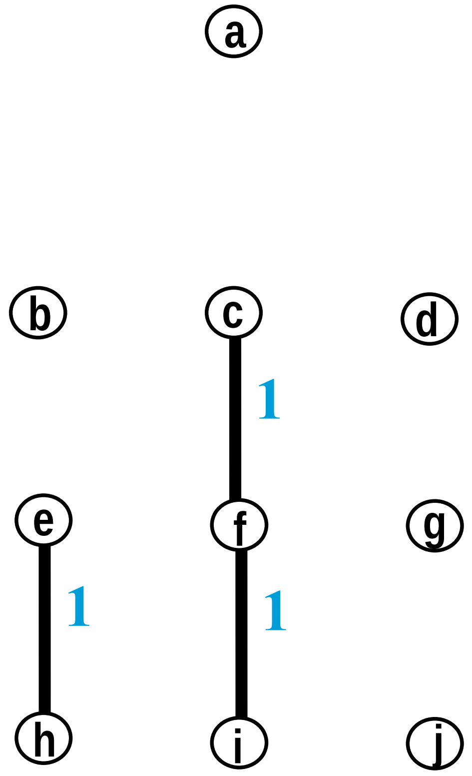




Algoritmo de Kruskal

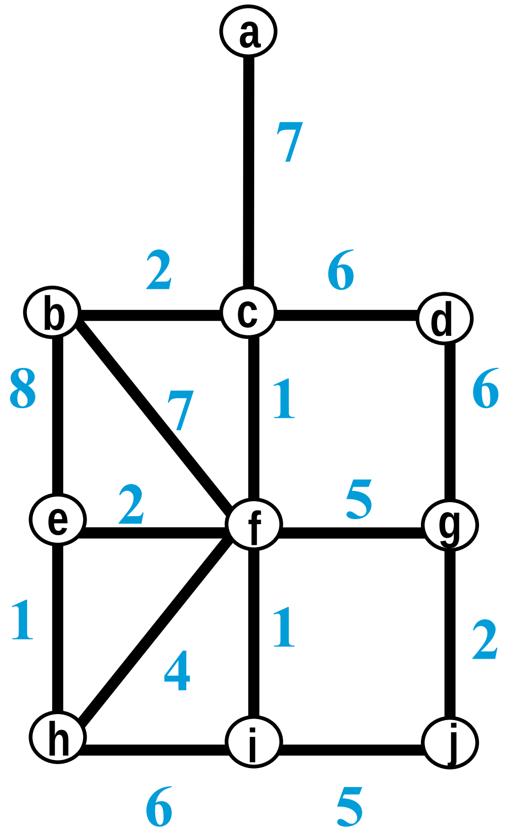
~~(c,f)~~, ~~(e,h)~~, ~~(f,i)~~, (b,c), (e,f), (g,j), (f,h), (f,g), (i,j), (c,d), (d,g), (h,i), (a,c), (b,f), (b,e)

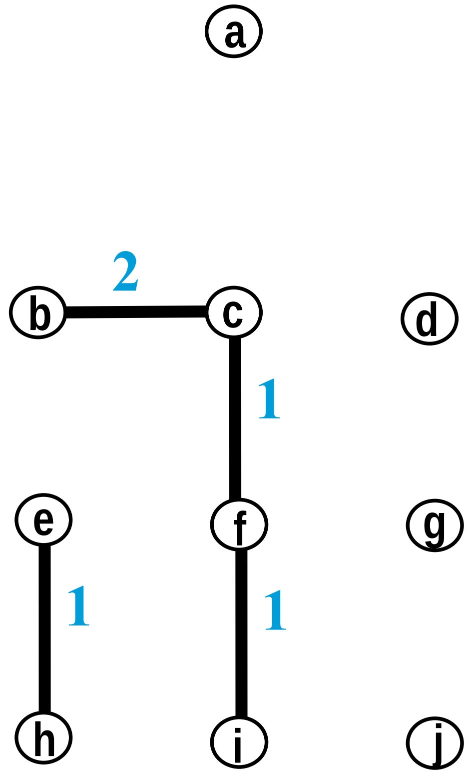




Algoritmo de Kruskal

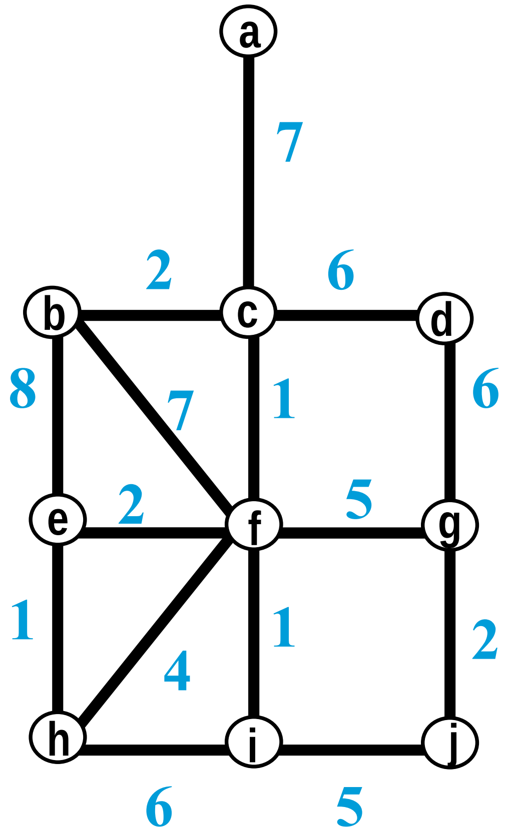
~~(c,f)~~, ~~(e,h)~~, ~~(f,i)~~, ~~(b,c)~~, (e,f), (g,j), (f,h), (f,g), (i,j), (c,d), (d,g), (h,i), (a,c), (b,f), (b,e)

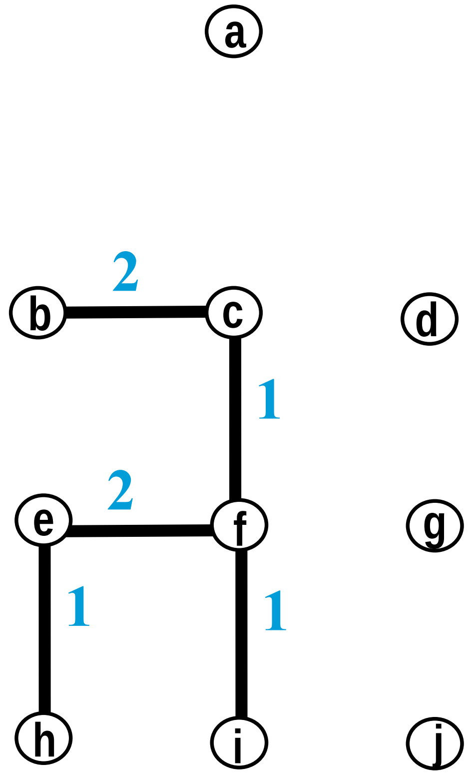




Algoritmo de Kruskal

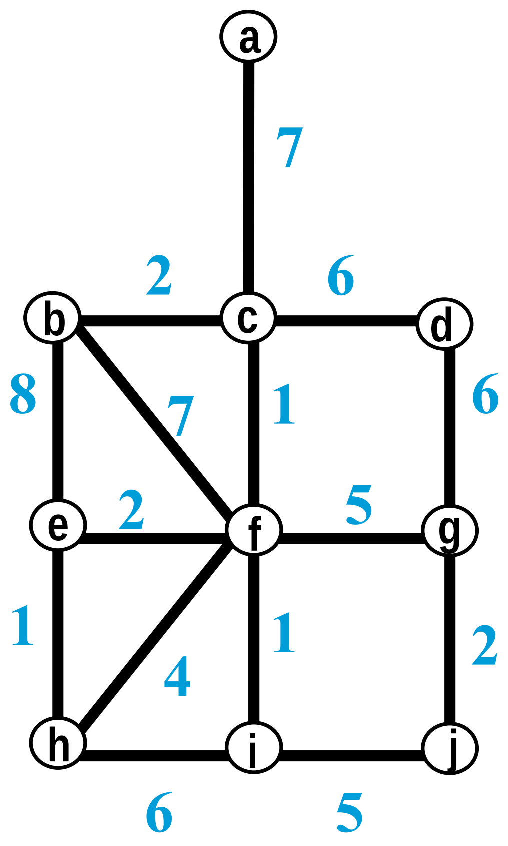
~~(c,f)~~, ~~(e,h)~~, ~~(f,i)~~, ~~(b,c)~~, ~~(e,f)~~, (g,j), (f,h), (f,g), (i,j), (c,d), (d,g), (h,i), (a,c), (b,f), (b,e)

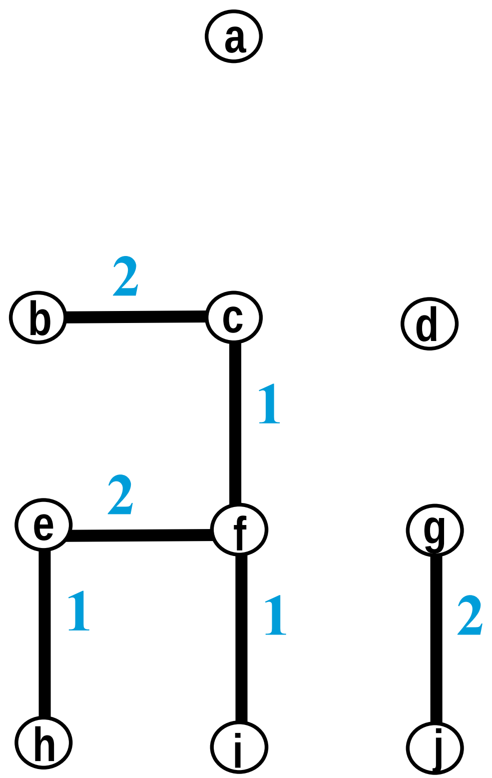




Algoritmo de Kruskal

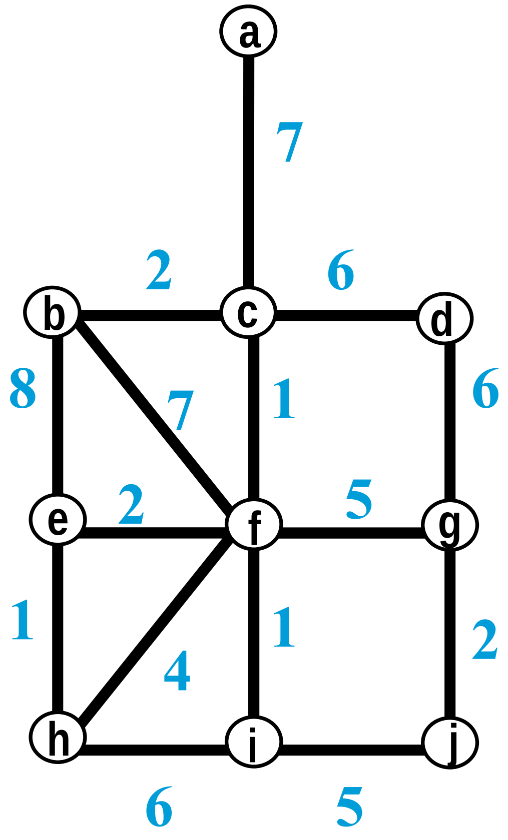
~~(c,f)~~, ~~(e,h)~~, ~~(f,i)~~, ~~(b,c)~~, ~~(e,f)~~, ~~(g,j)~~, (f,h), (f,g), (i,j), (c,d), (d,g), (h,i), (a,c), (b,f), (b,e)

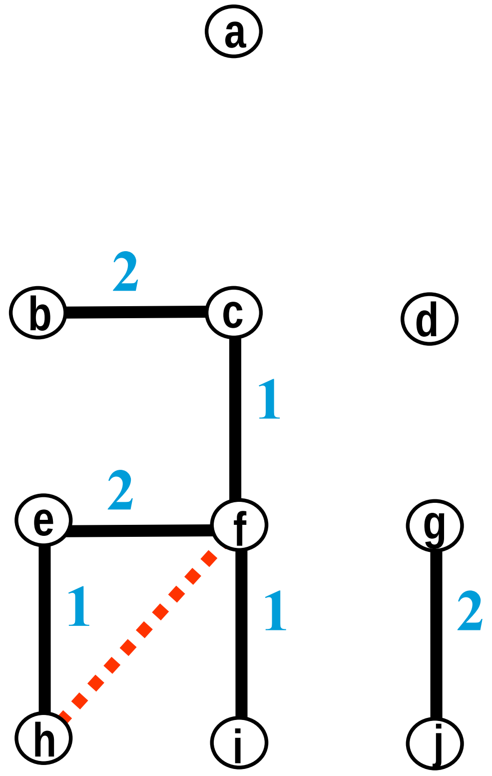




Algoritmo de Kruskal

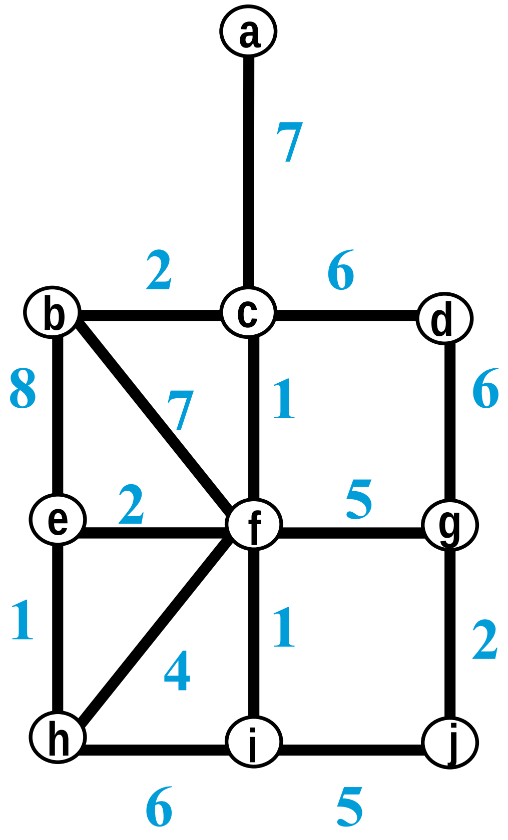
~~(c,f)~~, ~~(e,h)~~, ~~(f,i)~~, ~~(b,c)~~, ~~(e,f)~~, ~~(g,j)~~, ~~(f,h)~~, (f,g), (i,j), (c,d), (d,g), (h,i), (a,c), (b,f), (b,e)

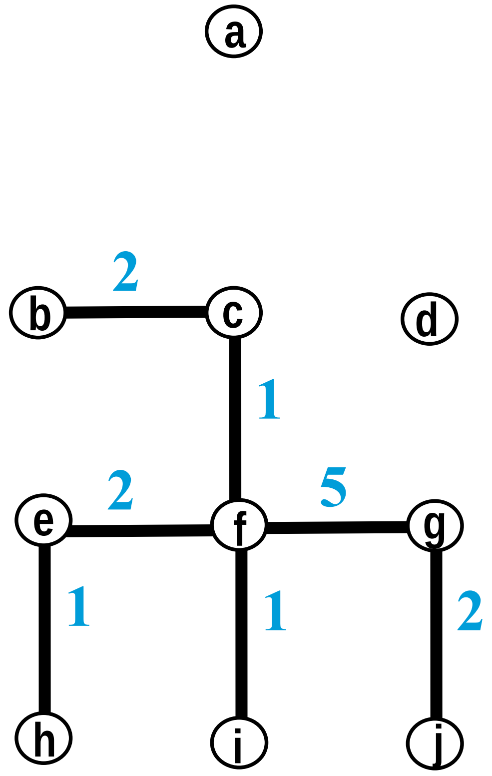




Algoritmo de Kruskal

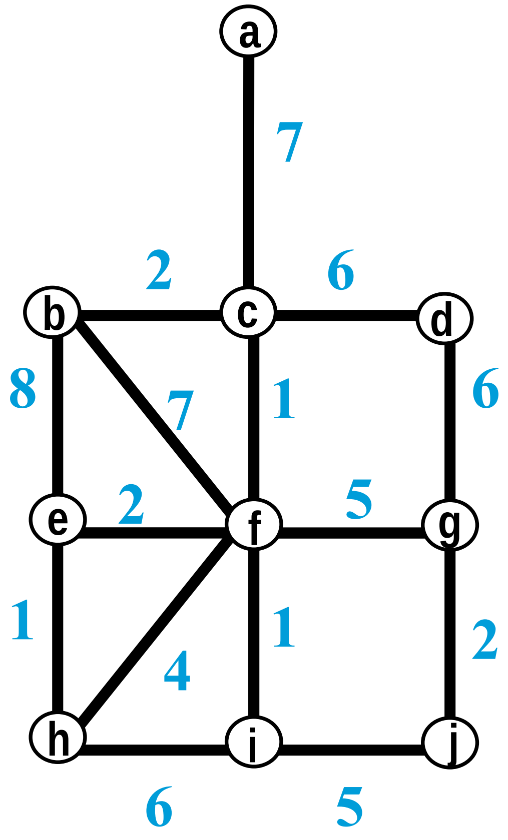
~~(c,f)~~, ~~(e,h)~~, ~~(f,i)~~, ~~(b,c)~~, ~~(e,f)~~, ~~(g,j)~~, ~~(f,h)~~, ~~(f,g)~~, (i,j), (c,d), (d,g), (h,i), (a,c), (b,f), (b,e)

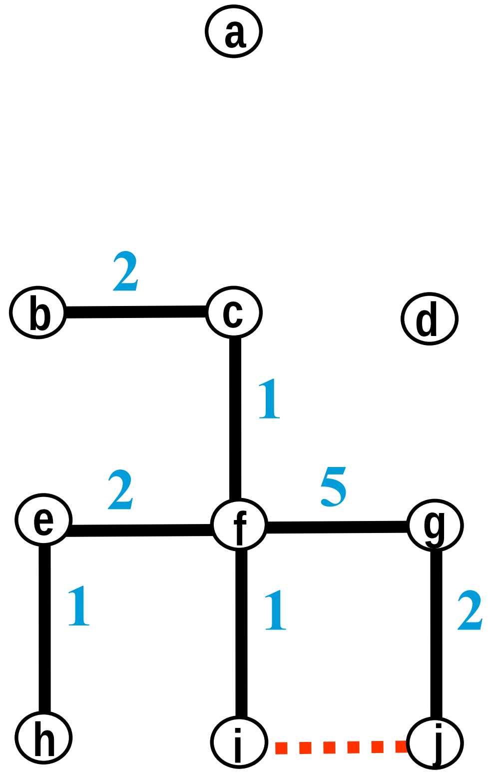




Algoritmo de Kruskal

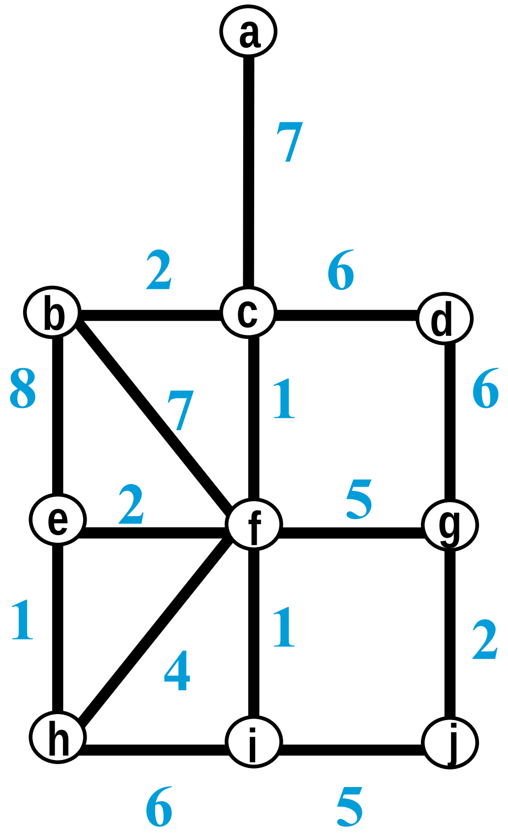
~~(c,f)~~, ~~(e,h)~~, ~~(f,i)~~, ~~(b,c)~~, ~~(e,f)~~, ~~(g,j)~~, ~~(f,h)~~, ~~(f,g)~~, ~~(i,j)~~, (c,d), (d,g), (h,i), (a,c), (b,f), (b,e)

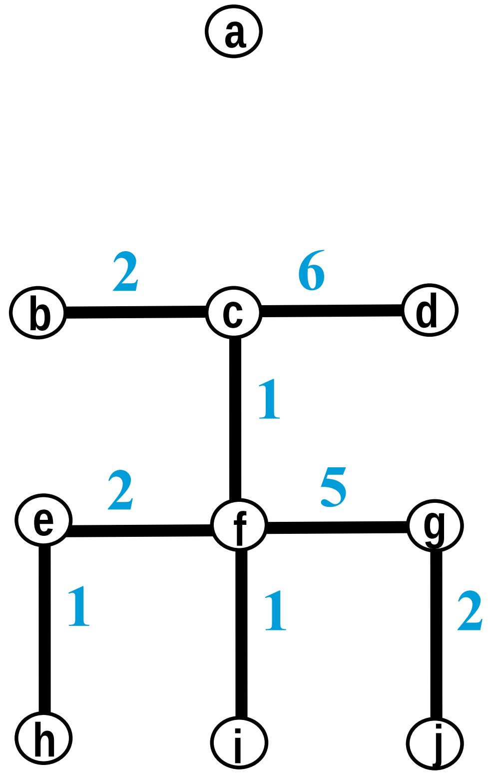




Algoritmo de Kruskal

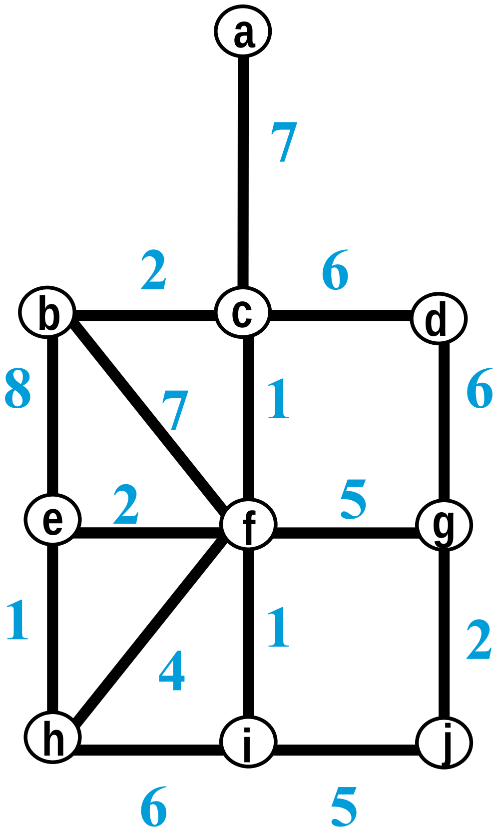
~~(c,f)~~, ~~(e,h)~~, ~~(f,i)~~, ~~(b,c)~~, ~~(e,f)~~, ~~(g,j)~~, ~~(f,h)~~, ~~(f,g)~~, ~~(i,j)~~, ~~(c,d)~~, (d,g), (h,i), (a,c), (b,f), (b,e)

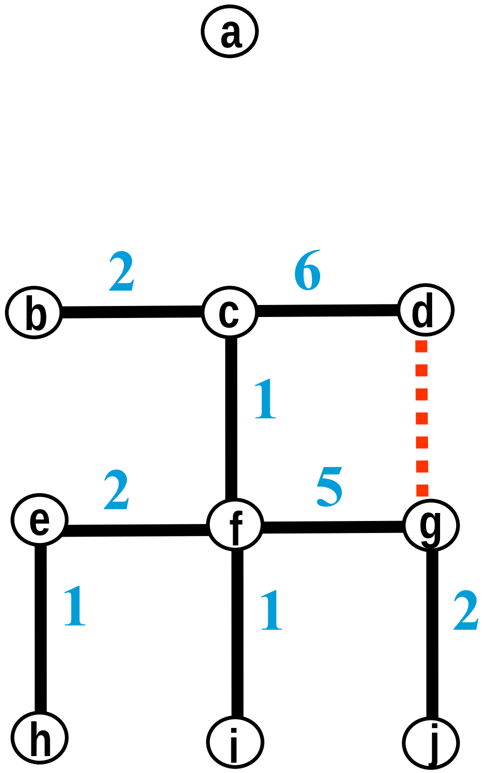




Algoritmo de Kruskal

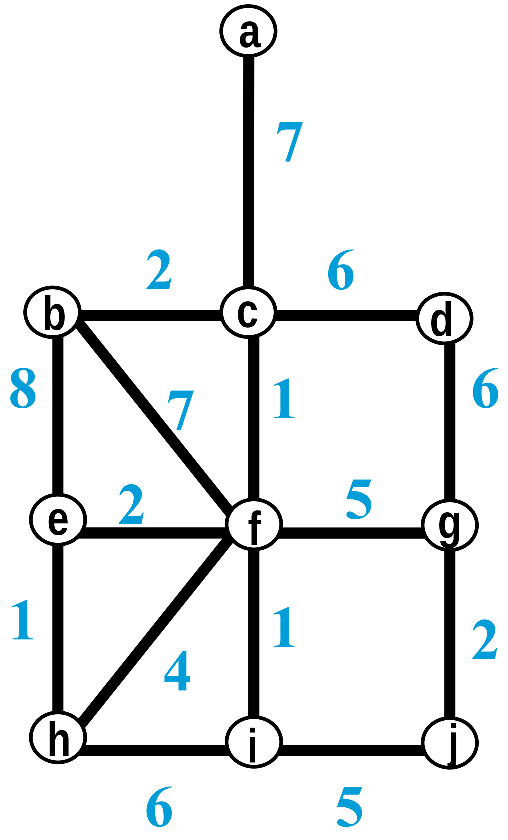
~~(c,f)~~, ~~(e,h)~~, ~~(f,i)~~, ~~(b,c)~~, ~~(e,f)~~, ~~(g,j)~~, ~~(f,h)~~, ~~(f,g)~~, ~~(i,j)~~, ~~(c,d)~~, ~~(d,g)~~, (h,i), (a,c), (b,f), (b,e)

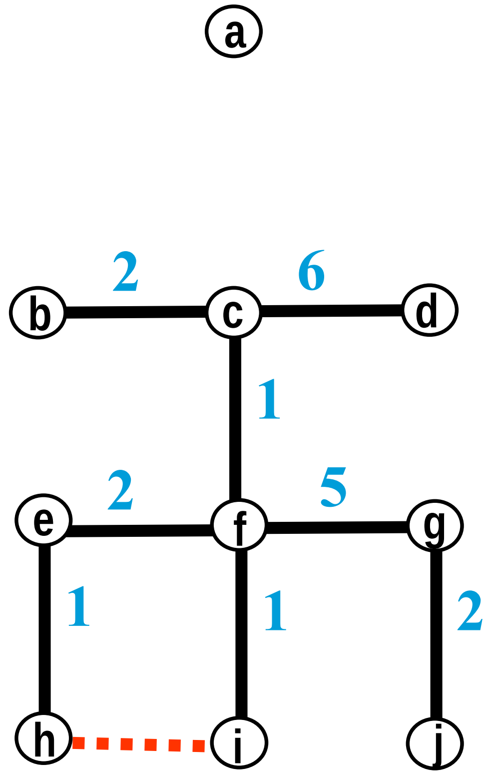




Algoritmo de Kruskal

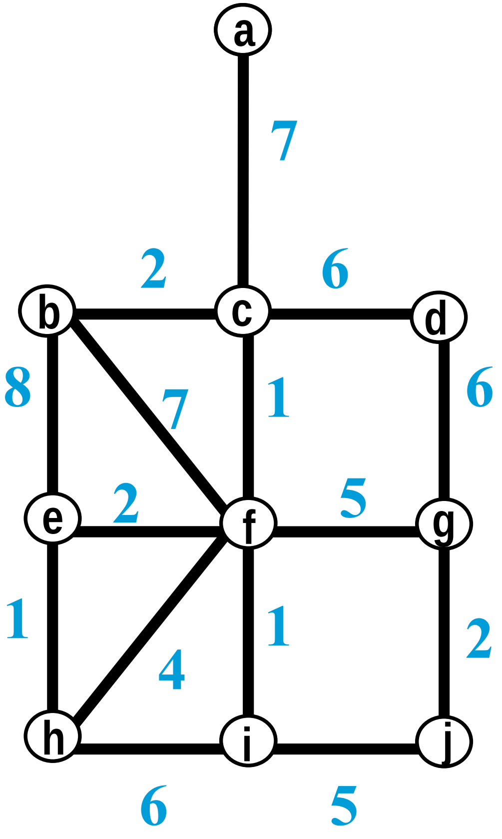
~~(c,f)~~, ~~(e,h)~~, ~~(f,i)~~, ~~(b,c)~~, ~~(e,f)~~, ~~(g,j)~~, ~~(f,h)~~, ~~(f,g)~~, ~~(i,j)~~, ~~(c,d)~~, ~~(d,g)~~, ~~(h,i)~~, (a,c), (b,f), (b,e)

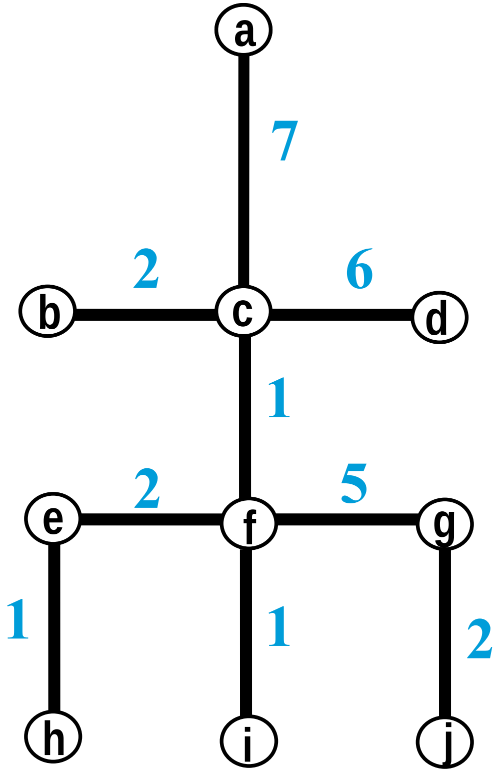




Algoritmo de Kruskal

~~(c,f)~~, ~~(e,h)~~, ~~(f,i)~~, ~~(b,c)~~, ~~(e,f)~~, ~~(g,j)~~, ~~(f,h)~~, ~~(f,g)~~, ~~(i,j)~~, ~~(c,d)~~, ~~(d,g)~~, ~~(h,i)~~, ~~(a,c)~~, (b,f), (b,e)





Algoritmo de Kruskal

Este algoritmo constrói uma árvore? Se sim, a árvore é mínima?

Dados $G=(N,M)$, $D=[d_{ij}]$ a matriz distância de G.
Ordene as arestas de G em ordem não decrescente das distâncias d_{ij} no vetor $H=(h_i)$, $i=1,2,\dots,m$

$T \leftarrow \{h_1\}$

$i \leftarrow 2$

Enquanto $|T| < n-1$ **faça**

  Se $T \cup h_i$ é um grafo acíclico **então**

    $T \leftarrow T \cup \{h_i\}$

    $i \leftarrow i+1$

  Caso Contrário

    $i \leftarrow i+1$

Escrever T {arestas da árvore geradora mínima}

???

Algoritmo de Kruskal

Prova da correção do algoritmo

Este algoritmo constrói uma árvore, pois adiciona *n*-1 arestas sem formar ciclo.

Dados $G = (N,M)$, $D = [d_{ij}]$ a matriz distância de G.
Ordene as arestas de G em ordem não decrescente das distâncias d_{ij} no vetor $H = (h_i)$, $i = 1,2,...,m$
$T \leftarrow \{h_1\}$
$i \leftarrow 2$
Enquanto $|T| < n-1$ **faça**
    Se $T \cup h_i$ é um grafo acíclico **então**
        $T \leftarrow T \cup \{h_i\}$
        $i \leftarrow i + 1$
    Caso Contrário
        $i \leftarrow i + 1$
Escrever T {arestas da árvore geradora mínima}

***G* é uma árvore**
se e somente se
***G* é acíclico e possui *n*-1 arestas**

Algoritmo de Kruskal

Prova da correção do algoritmo

Suponha que a árvore não é mínima.
Então, a adição de alguma aresta (i,j) fez com que a árvore mínima, T_{min}, não fosse obtida. Uma vez que o algoritmo não cria ciclos, i e j estavam em componentes diferentes quando a aresta foi inserida.
Então, alguma aresta no caminho entre i e j não foi incluída pelo algoritmo, a qual produziria a árvore T_{min}.
Essa aresta teria um custo menor que (i,j). Então, o algoritmo teria desconsiderado esta aresta de peso menor.

ABSURDO.

Algoritmo de Kruskal

Qual é a complexidade?

Dados $G=(N,M)$, $D=[d_{ij}]$ a matriz distância de G.
Ordene as arestas de G em ordem não decrescente das distâncias d_{ij} no vetor H = (h_i), $i=1,2,\ldots,m$

$T \leftarrow \{h_1\}$

$i \leftarrow 2$

Enquanto $|T| < n-1$ **faça**

    Se $T \cup h_i$ é um grafo acíclico **então**

        $T \leftarrow T \cup \{h_i\}$

        $i \leftarrow i + 1$

    Caso Contrário

        $i \leftarrow i + 1$

Escrever T {arestas da árvore geradora mínima}

Algoritmo de Kruskal

Complexidade (sem estruturas de dados especiais)

Ordenação das arestas: $O(m \log m) = O(m \log n)$

O "enquanto" faz $O(m)$ iterações.

Verificação de ciclo: $O(?)$

Dados $G = (N,M)$, $D = [d_{ij}]$ a matriz distância de G.
Ordene as arestas de G em ordem não decrescente das distâncias d_{ij} no vetor H = (h_i), $i = 1,2,\ldots,m$
$T \leftarrow \{h_1\}$
$i \leftarrow 2$
Enquanto $|T| < n-1$ **faça**
    Se $T \cup h_i$ é um grafo acíclico **então**
        $T \leftarrow T \cup \{h_i\}$
        $i \leftarrow i + 1$
    Caso Contrário
        $i \leftarrow i + 1$
Escrever T {arestas da árvore geradora mínima}

Algoritmo de Kruskal

Complexidade (sem estruturas de dados especiais)

Verificação de ciclo: utilizando um vetor para controlar as diferentes componentes.

$O(1)$ para ver se os vértices pertencem a mesma componente
$O(n)$ para atualizar a estrutura (é atualizada n vezes)

$$O(n^2)$$

Algoritmo de Kruskal

Complexidade (sem estruturas de dados especiais)

$O(m \log n) + O(m - n + n^2)$

Final: $O(n^2)$

Obs: A verificação de ciclo pode ser implementada em $O(n \log n)$ utilizando dois vetores

Algoritmo de Kruskal

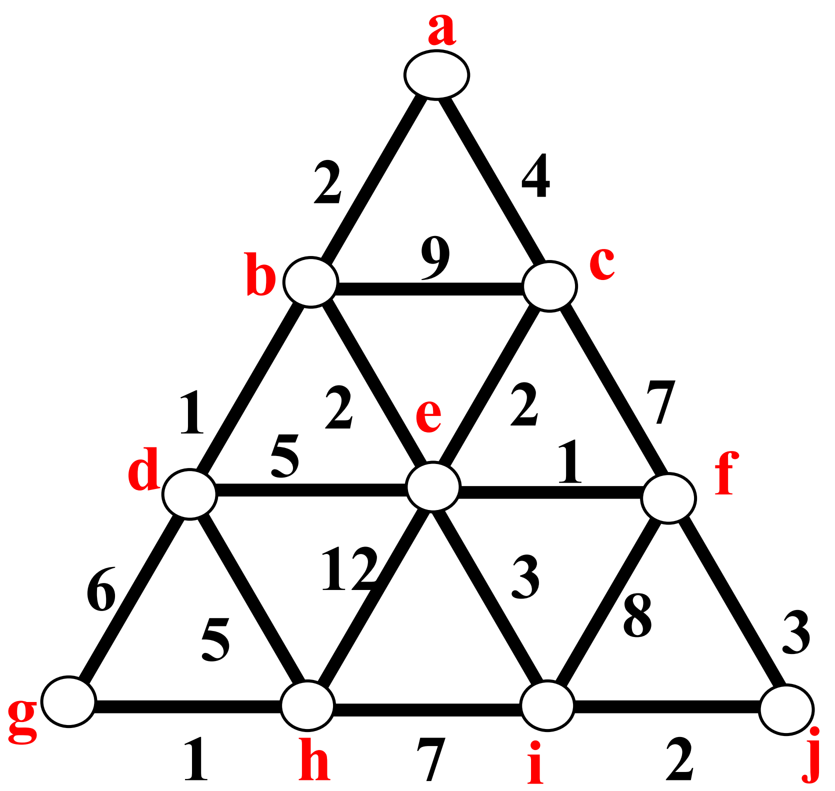
Complexidade

Pesquisa: Qual a complexidade do algoritmo usando estruturas de dados mais sofisticadas?

Dica: Verificar o uso de “*conjuntos disjuntos*” (*union-find*).

Algoritmo de Kruskal

Exercício Kruskal



G