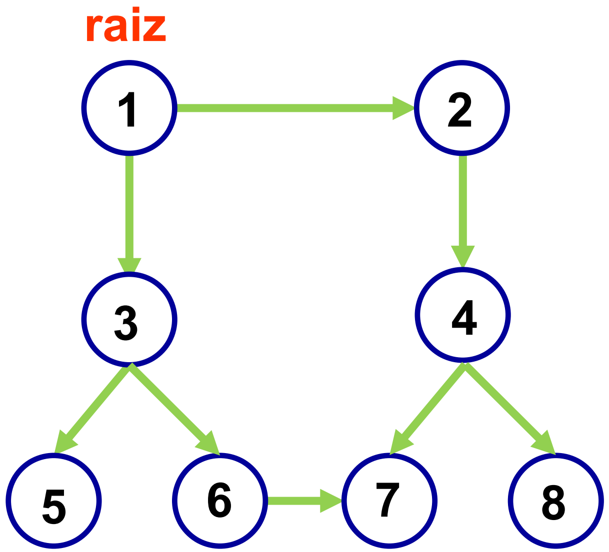
Árvore Geradora Mínima de Grafos Direcionados

AGM em Digrafos

Um vértice *v* em um grafo direcionado *G* é dito *raiz de G* se existe caminho direcionado de *v* para todo nó de *G*.

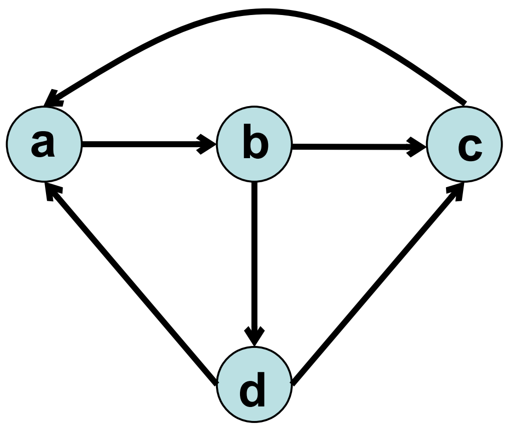
AGM em Digrafos

Um vértice *v* em um grafo direcionado *G* é dito ***raiz de G*** se existe caminho direcionado de *v* para todo nó de *G*.

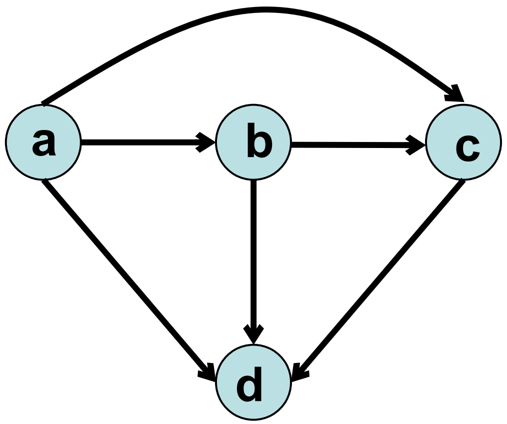


AGM em Digrafos

Um digrafo D é dito **fortemente conexo**, se para todo par u,w ∈ V existe caminho direcionado de u para w e de w para u.



Digrafo Fortemente Conexo

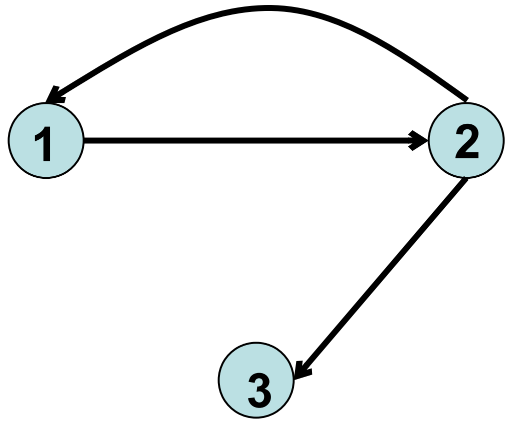


Digrafo Conexo

AGM em Digrafos

Um digrafo D é dito **quase-fortemente conexo**, se para todo par u,w ∈ V existe um vértice *v* tal que existe um caminho direcionado de *v* para u e de *v* para w.

Obs: *v* não é necessariamente diferente de u ou de w.



u	w	v
1	2	1
1	3	2
2	3	2

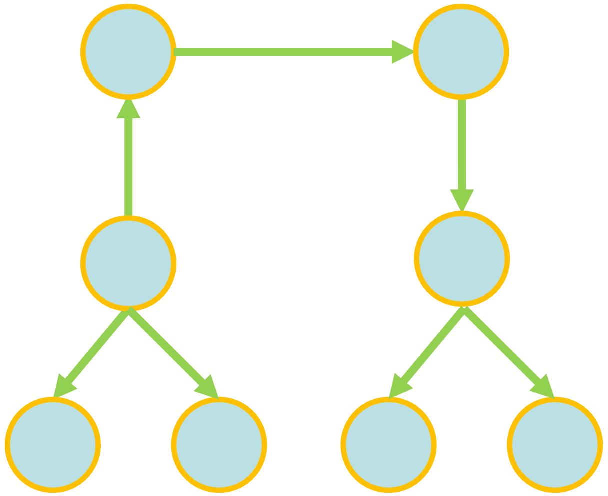
AGM em Digrafos

Se um grafo contém uma raiz, então ele é *quase-fortemente conexo*.

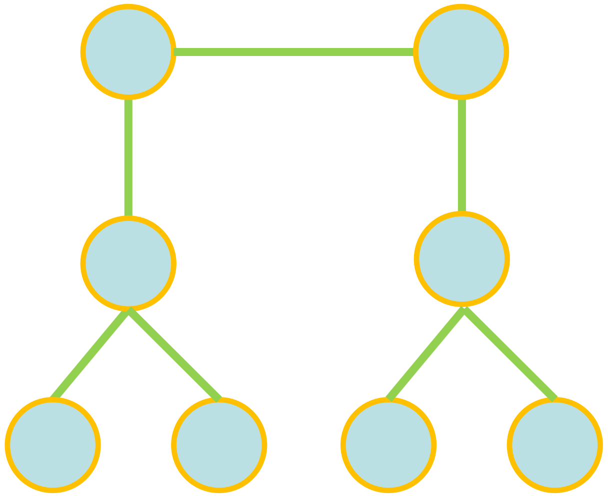
***Teorema*. Um grafo direcionado *G* tem uma raiz se e somente se ele é quase fortemente conexo.**

AGM em Digrafos

Um digrafo *G* é uma árvore se seu grafo subjacente é uma árvore.



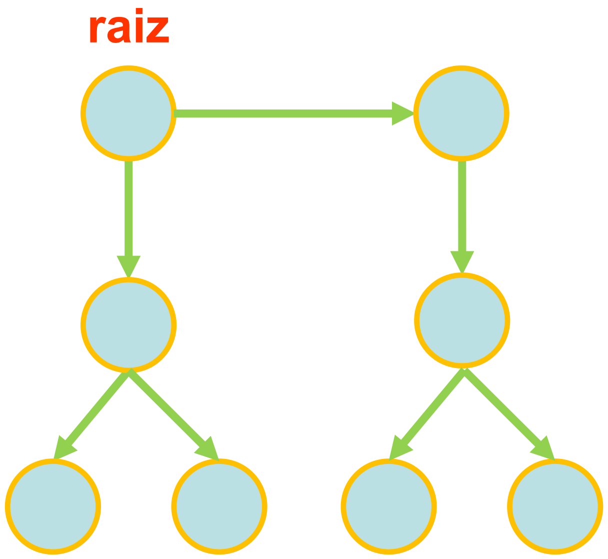
Árvore



Grafo subjacente

AGM em Digrafos

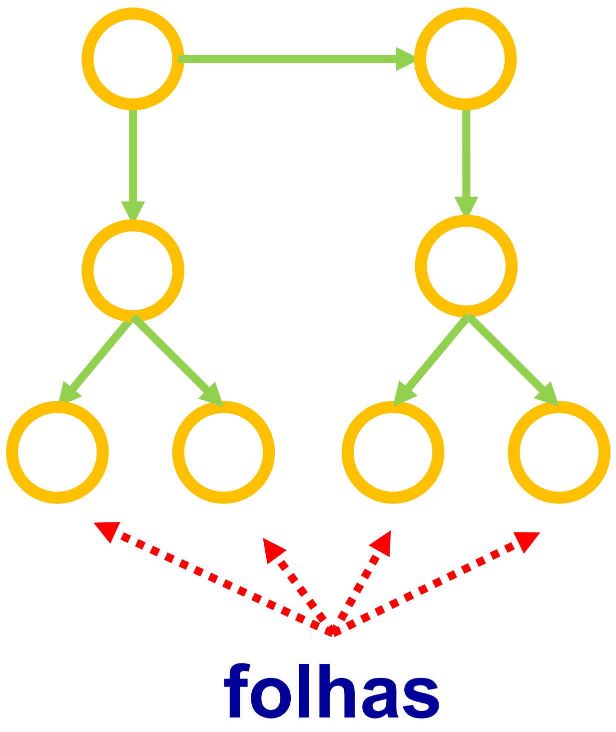
Um digrafo *G* é uma *árvore direcionada* ou *arborescência* se *G* é uma árvore e possui raiz.



Árvore Direcionada

AGM em Digrafos

Uma *folha* em uma arborescência é um vértice com grau de saída igual a 0.



AGM em Digrafos

Teorema de caracterização de arborescência

1. *G* é uma árvore direcionada

2. Existe um vértice *r* em *G* tal que existe exatamente um caminho direcionado de *r* para cada vértice de *G*.

3. *G* é quase fortemente conexo e perde tal propriedade se qualquer aresta for removida dele.

4. *G* é quase fortemente conexo e possui vértice *r* tal que $d^-(r) = 0$ e $d^-(v) = 1$, para todo $v \neq r$.

5. O grafo subjacente de *G* é acíclico e *G* possui um vértice *r* tal que que $d^-(r) = 0$ e $d^-(v) = 1$, para todo $v \neq r$.

6. *G* é quase fortemente conexo e seu grafo subjacente é acíclico.

AGM em Digrafos

Teorema. Um grafo direcionado *G* tem uma árvore geradora direcionada se e somente se *G* é quase-fortemente conexo.

Algoritmo de Chu-Liu/Edmonds

No caso de grafos dirigidos, os algoritmos clássicos não podem ser empregados, dado que a decisão gulosa iterativa não mais é capaz de fornecer a solução ótima.

~~**Kruskal**~~ ~~**Prim**~~ ~~**Borüvka**~~

Algoritmo de Chu-Liu/Edmonds

- **Chu e Liu (1965) e Edmonds (1967) desenvolveram independentemente um algoritmo $O(mn)$ para encontrar a árvore geradora mínima dirigida.**
- **Implementação de Tarjan (1977): $O(m\log n)$ para grafos esparsos e $O(n^2)$ para grafos densos.**
- **Implementação de Gabow et al. (1986): utiliza heap de Fibonacci, $O(n\log n + m)$.**

Algoritmo de Chu-Liu/Edmonds

Algoritmo

1. **Descarte os arcos que entram na raiz, se houver; Para cada nó diferente da raiz, selecione o arco de entrada com o menor custo; Defina os *n*-1 arcos selecionados como o conjunto S.**

Algoritmo de Chu-Liu/Edmonds

Algoritmo

1. Descarte os arcos que entram na raiz, se houver; Para cada nó diferente da raiz, selecione o arco de entrada com o menor custo; Defina os n-1 arcos selecionados como o conjunto S.
2. Se nenhum ciclo for formado, $G = (N, S)$ é uma árvore geradora mínima. Caso contrário, continue.

Algoritmo de Chu-Liu/Edmonds

Algoritmo

1. Descarte os arcos que entram na raiz, se houver; Para cada nó diferente da raiz, selecione o arco de entrada com o menor custo; Defina os *n*-1 arcos selecionados como o conjunto S.
2. Se nenhum ciclo for formado, *G* = (*N*, *S*) é uma árvore geradora mínima. Caso contrário, continue.
3. **Para cada ciclo formado, contraia os nós do ciclo em um pseudo-nó *k* e modifique o custo de cada arco (*i*,*j*) que chega a um nó *j* pertencente ao ciclo a partir de um nó *i* fora do ciclo de acordo com a seguinte equação.**

$$c(i,k) = c(i,j) – [c(x(j), j) - \min_{\{r\}}(c(x(r), r)]$$

onde c(x(j),j) é o custo do arco que chega a j no ciclo.

Algoritmo de Chu-Liu/Edmonds

Algoritmo

1. **Descarte os arcos que entram na raiz, se houver; Para cada nó diferente da raiz, selecione o arco de entrada com o menor custo; Defina os *n*-1 arcos selecionados como o conjunto S.**
2. **Se nenhum ciclo for formado, *G* = (*N*, *S*) é uma árvore geradora mínima. Caso contrário, continue.**
3. **Para cada ciclo formado, contraia os nós do ciclo em um pseudo-nó *k* e modifique o custo de cada arco (*i*,*j*) que chega a um nó *j* pertencente ao ciclo a partir de um nó *i* fora do ciclo de acordo com a seguinte equação.**

$$c(i,k) = c(i,j) - [c(x(j), j) - \min_{\{r\}}(c(x(r), r)]$$

onde c(x(j),j) é o custo do arco que chega a j no ciclo.

4. **Para cada pseudo-nó, selecione o arco de entrada que tem o menor custo modificado; Substitua o arco que entra no mesmo nó *real* em S pelo novo arco selecionado.**
5. **Vá para o passo 2 com o grafo contraído.**

Algoritmo de Chu-Liu/Edmonds

- **Algoritmo Chu-Liu/Edmonds**

1. **Descarte os arcos que entram na raiz, se houver; Para cada nó diferente da raiz, selecione o arco de entrada com o menor custo; Defina os *n*-1 arcos selecionados como o conjunto S.**
2. **Se nenhum ciclo for formado, *G* = (*N*, *S*) é uma árvore geradora mínima. Caso contrário, continue.**
3. **Para cada ciclo formado, contraia os nós do ciclo em um pseudo-nó *k* e modifique o custo de cada arco (*i*,*j*) que chega a um nó *j* pertencente ao ciclo a partir de um nó *i* fora do ciclo de acordo com a seguinte equação.**

   $$c(i,k) = c(i,j) – [c(x(j), j) - \min_{\{r\}}(c(x(r), r)]$$

   onde c(x(j),j) é o custo do arco que chega a j no ciclo.
4. **Para cada pseudo-nó, selecione o arco de entrada que tem o menor custo modificado; Substitua o arco que entra no mesmo nó *real* em S pelo novo arco selecionado.**
5. **Vá para o passo 2 com o grafo contraído.**

Algoritmo de Chu-Liu/Edmonds

Ideias do algoritmo de Chu-Liu/Edmonds

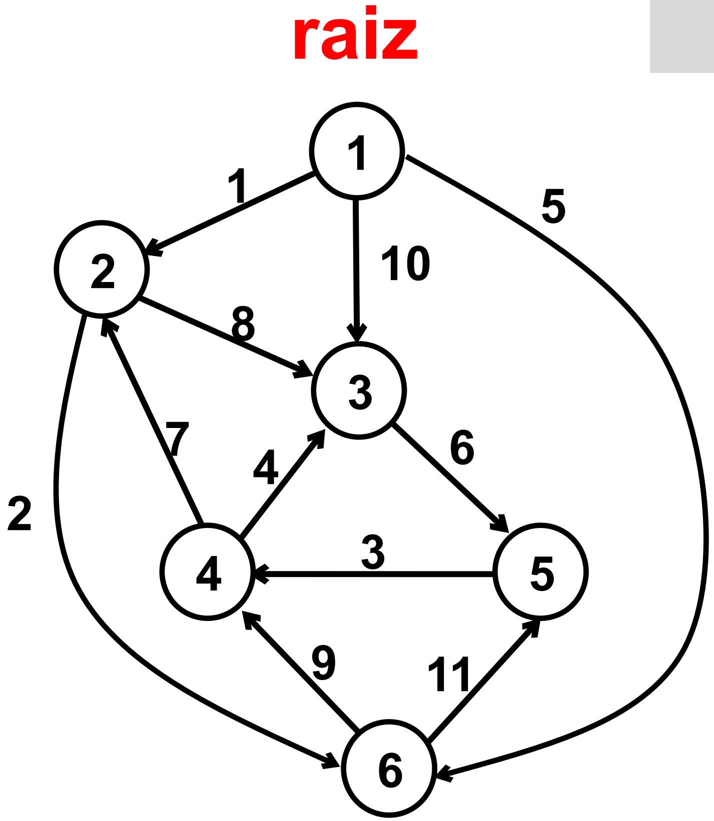
- Encontrar arcos substitutos que provocam menor custo extra e eliminam ciclos.

- A equação apresentada no algoritmo representa o cálculo do custo extra.

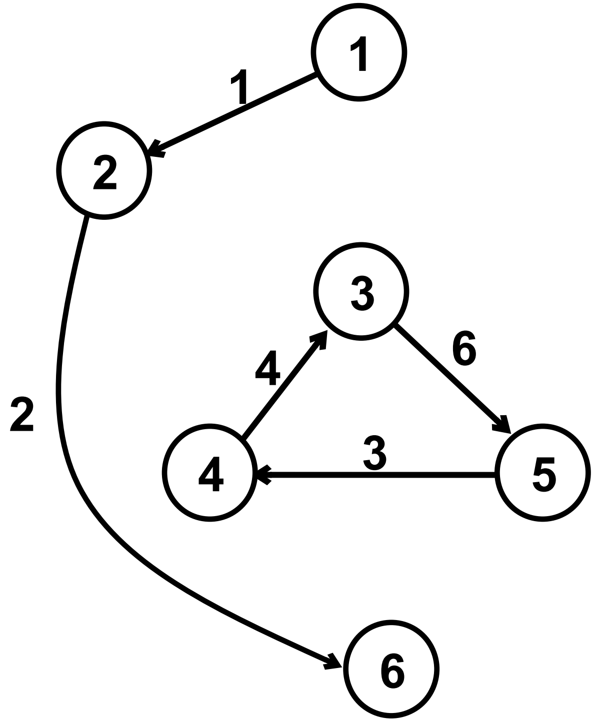
Algoritmo de Chu-Liu/Edmonds

Exemplo:

1. **Descarte os arcos que entram na raiz, se houver; Para cada nó diferente da raiz, selecione o arco de entrada com o menor custo; Defina os *n*-1 arcos selecionados como o conjunto S.**



Grafo D

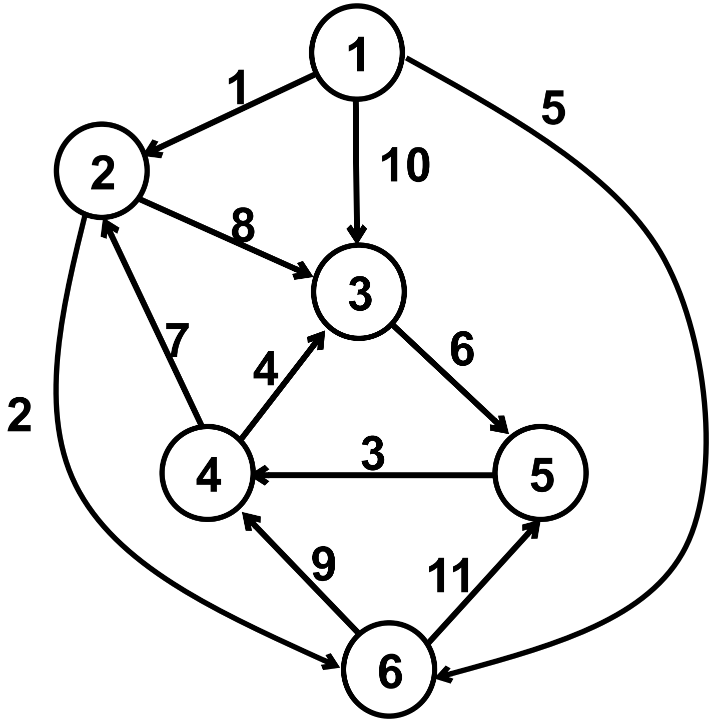


Algoritmo de Chu-Liu/Edmonds

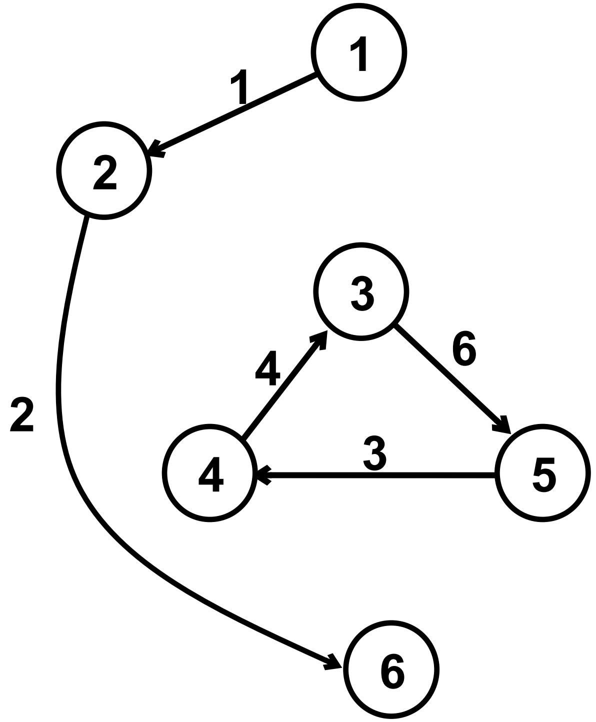
Exemplo:

2. Se nenhum ciclo for formado, *G* = (*N*, *S*) é uma árvore geradora mínima. Caso contrário, continue.

raiz

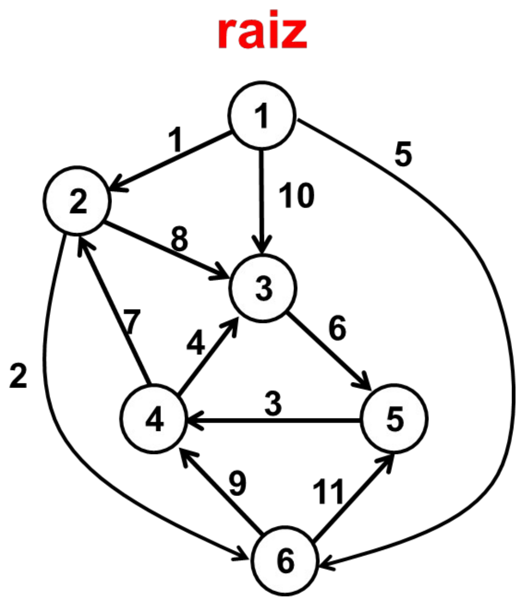


Grafo D



Algoritmo de Chu-Liu/Edmonds

Exemplo:

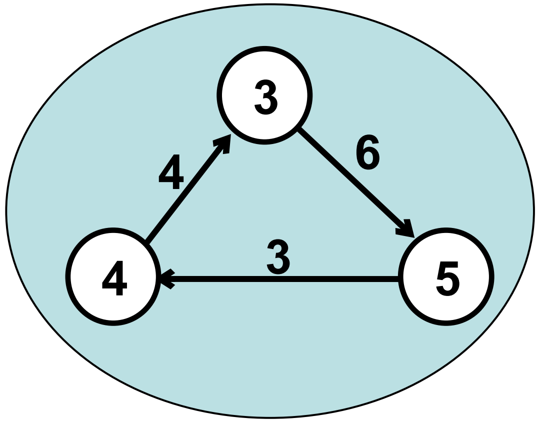


Grafo D

3. Para cada ciclo formado, contraia os nós do ciclo em um pseudo-nó *k* e modifique o custo de cada arco (*i,j*) que chega a um nó *j* pertencente ao ciclo a partir de um nó *i* fora do ciclo de acordo com a seguinte equação.

$$c(i,k) = c(i,j) - [c(x(j), j) - \min_{\{r\}}(c(x(r), r)]$$

onde c(x(j),j) é o custo do arco que chega a j no ciclo.



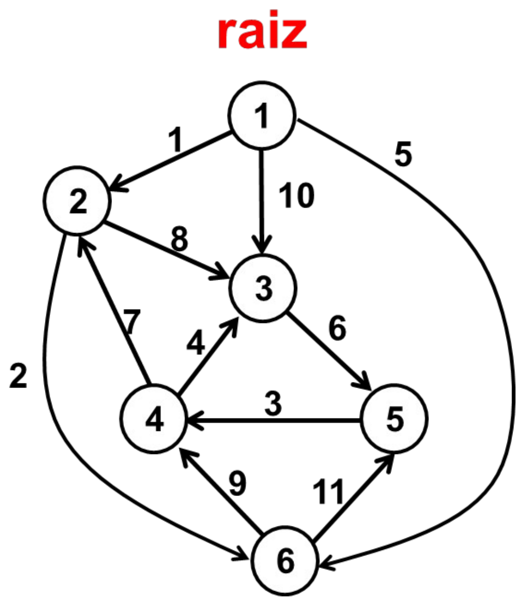
k

Arcos que chegam a um nó do ciclo

Arco	Custo modificado
(1,3)	c(1,3) = c(1,3) – [c(4,3) – c(5,4)] = = 10 - [4 – 3] = 9
(2,3)	
(6,4)	
(6,5)	

Algoritmo de Chu-Liu/Edmonds

Exemplo:

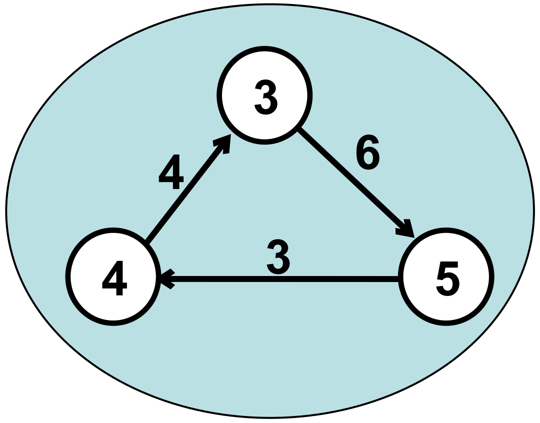


Grafo D

3. Para cada ciclo formado, contraia os nós do ciclo em um pseudo-nó *k* e modifique o custo de cada arco (*i,j*) que chega a um nó *j* pertencente ao ciclo a partir de um nó *i* fora do ciclo de acordo com a seguinte equação.

$$c(i,k) = c(i,j) - [c(x(j), j) - \min_{\{r\}}(c(x(r), r)]$$

onde c(x(j),j) é o custo do arco que chega a j no ciclo.



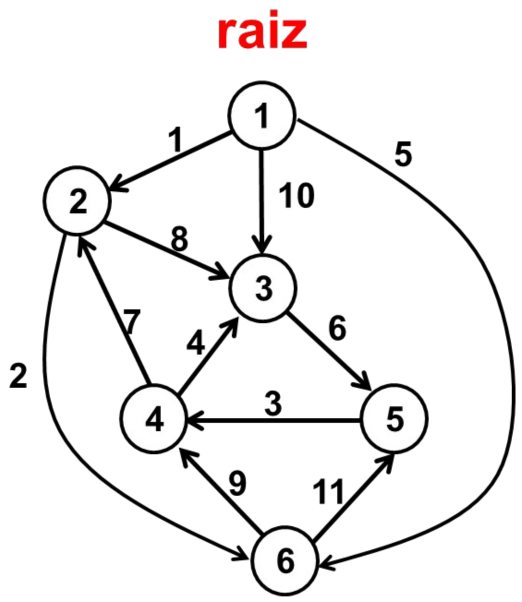
k

Arcos que chegam a um nó do ciclo

Arco	Custo modificado
(1,3)	9
(2,3)	c(2,3) = c(2,3) – [c(4,3) – c(5,4)] = = 8 - [4 – 3] = 7
(6,4)	
(6,5)	

Algoritmo de Chu-Liu/Edmonds

Exemplo:

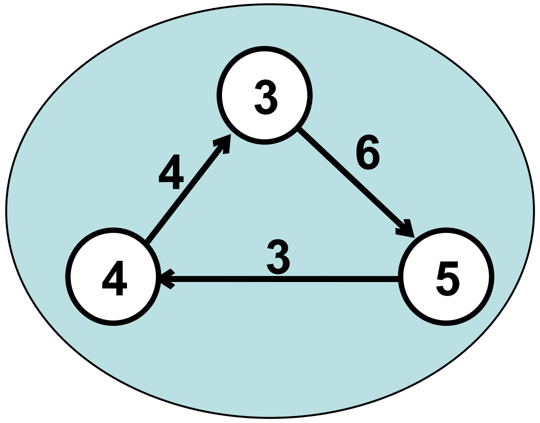


Grafo D

3. Para cada ciclo formado, contraia os nós do ciclo em um pseudo-nó *k* e modifique o custo de cada arco (*i,j*) que chega a um nó *j* pertencente ao ciclo a partir de um nó *i* fora do ciclo de acordo com a seguinte equação.

$$c(i,k) = c(i,j) – [c(x(j), j) - \min_{\{r\}}(c(x(r), r)]$$

onde c(x(j),j) é o custo do arco que chega a j no ciclo.



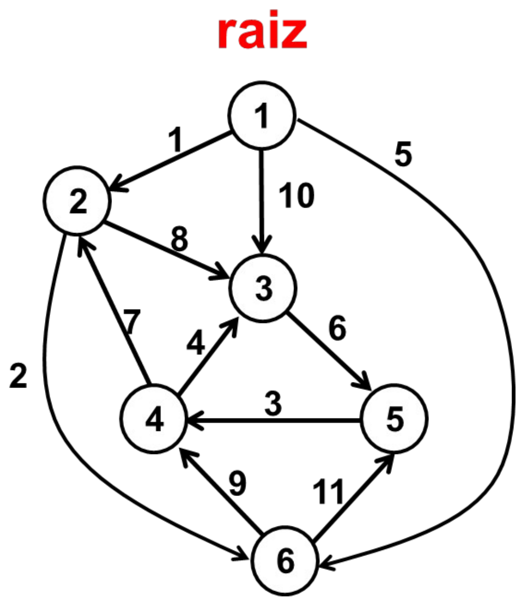
k

Arcos que chegam a um nó do ciclo

Arco	Custo modificado
(1,3)	9
(2,3)	7
(6,4)	c(6,4) = c(6,4) – [c(5,4) – c(5,4)] = = 9 - [3 – 3] = 9
(6,5)	

Algoritmo de Chu-Liu/Edmonds

Exemplo:

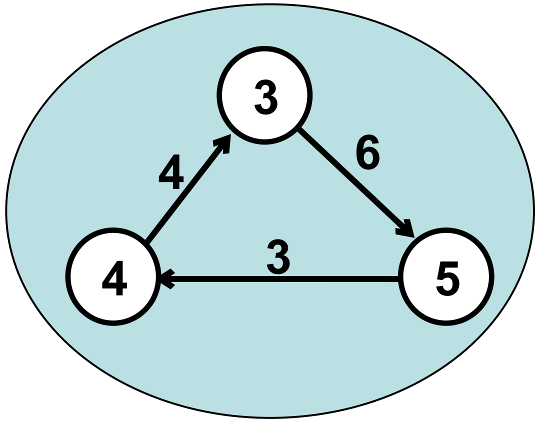


Grafo D

3. Para cada ciclo formado, contraia os nós do ciclo em um pseudo-nó *k* e modifique o custo de cada arco (*i,j*) que chega a um nó *j* pertencente ao ciclo a partir de um nó *i* fora do ciclo de acordo com a seguinte equação.

$$c(i,k) = c(i,j) - [c(x(j), j) - \min_{\{r\}}(c(x(r), r)]$$

onde c(x(j),j) é o custo do arco que chega a j no ciclo.



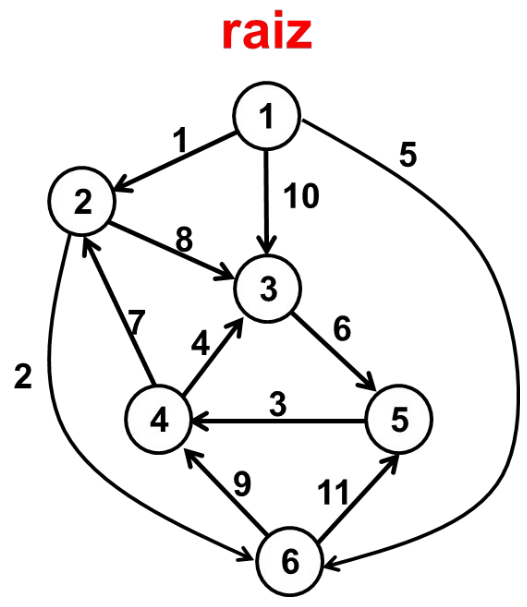
k

Arcos que chegam a um nó do ciclo

Arco	Custo modificado
(1,3)	9
(2,3)	7
(6,4)	9
(6,5)	c(6,5) = c(6,5) – [c(3,5) – c(5,4)] = = 11 - [6 – 3] = 8

Algoritmo de Chu-Liu/Edmonds

Exemplo:

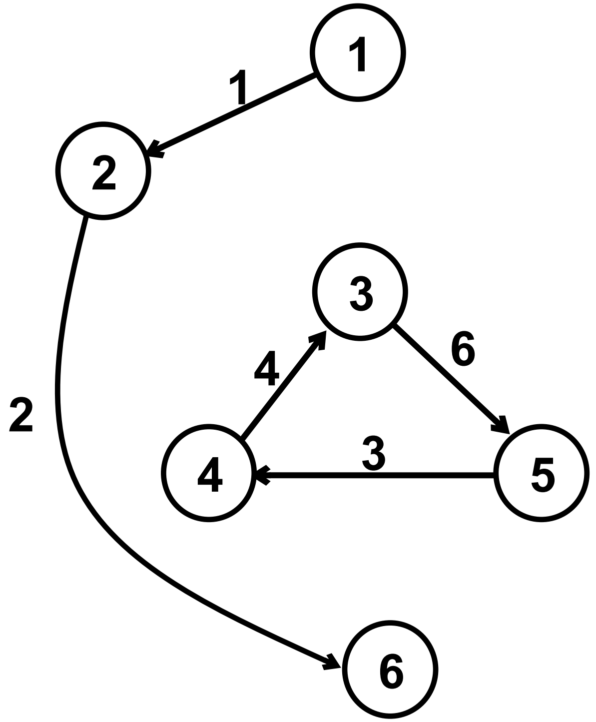


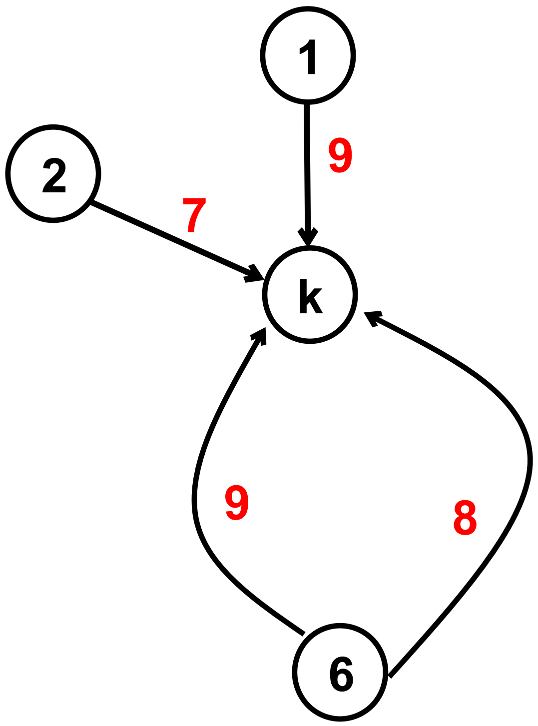
Grafo D

3. Para cada ciclo formado, contraia os nós do ciclo em um pseudo-nó *k* e modifique o custo de cada arco (*i,j*) que chega a um nó *j* pertencente ao ciclo a partir de um nó *i* fora do ciclo de acordo com a seguinte equação.

$$c(i,k) = c(i,j) – [c(x(j), j) - \min_{\{r\}}(c(x(r), r)]$$

onde c(x(j),j) é o custo do arco que chega a j no ciclo.

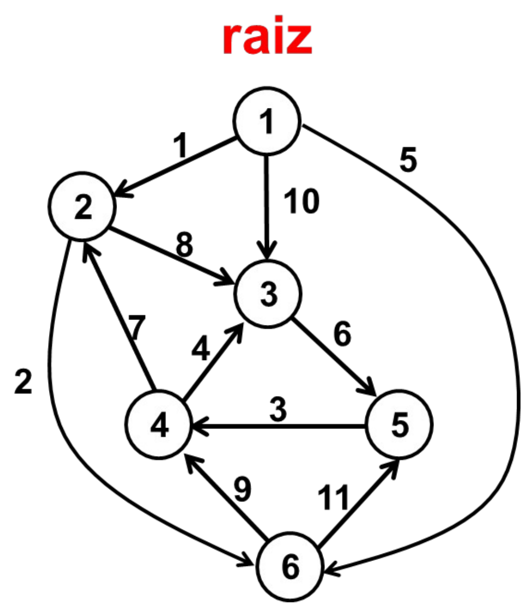




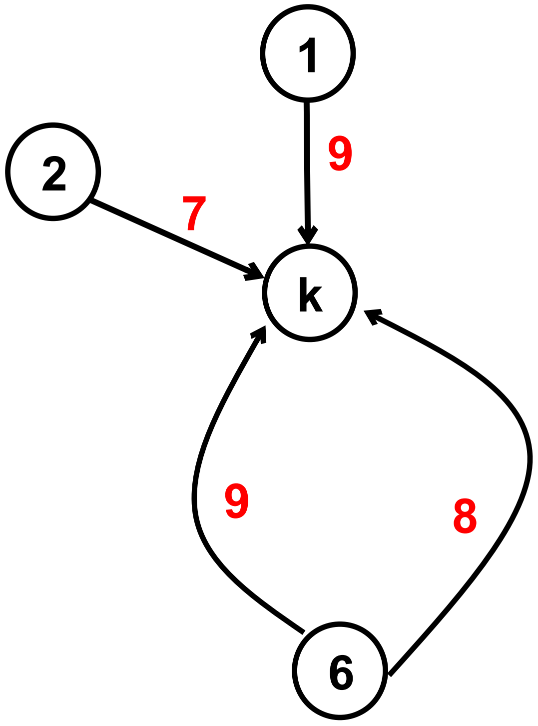
Algoritmo de Chu-Liu/Edmonds

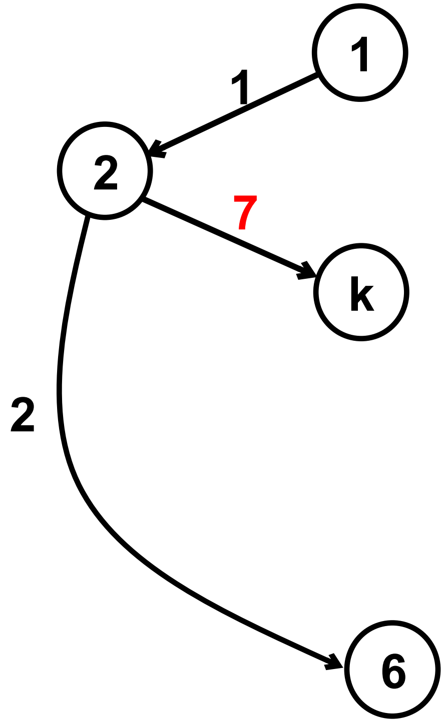
Exemplo:

4. Para cada pseudo-nó, selecione o arco de entrada que tem o menor custo modificado; Substitua o arco que entra no mesmo nó *real* em S pelo novo arco selecionado.



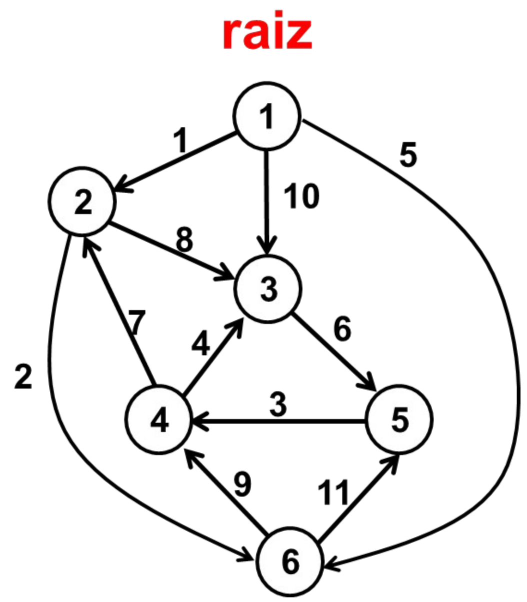
Grafo D





Algoritmo de Chu-Liu/Edmonds

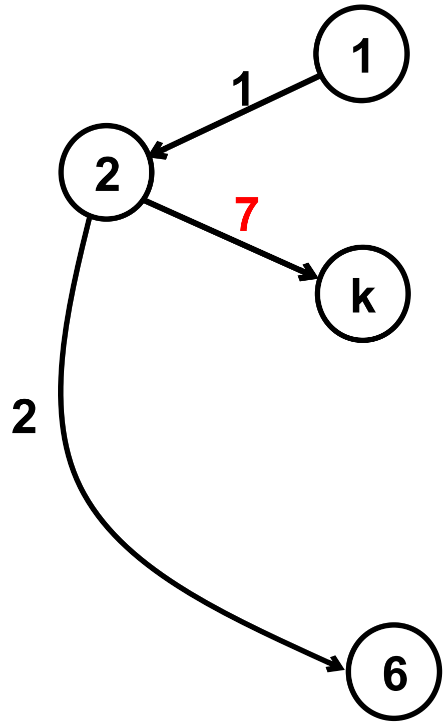
Exemplo:

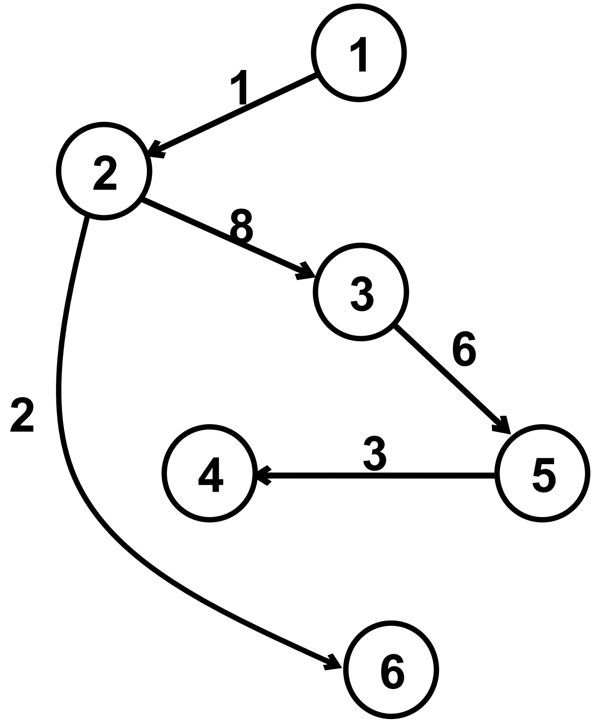


Grafo D

5. Vá para o passo 2 com o grafo contraído.

2. Se nenhum ciclo for formado, $G = (N, S)$ é uma árvore geradora mínima. Caso contrário, continue.





Fim

Bibliografia e Referências

- CHU, Y.J.; LIU, T.H. On the shortest arborescence of a directed graph, *Science Sinica*, v.14, 1965, pp.1396-1400.
- EDMONDS, J. Optimum branchings, *J. Research of the National Bureau of Standards*, 71B, 1967, pp.233-240.
- GABOW, H.N., GALIL, Z.,SPENCER, T., TARJAN, R.E. Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. *Combinatorica v.* 6, pp. 109-122.
- TARJAN, R.E. Finding Optimum Branchings, *Networks*, v.7, 1977, pp.25-35.