Algoritmo Floyd-Warshall

S 21.3

Problema dos caminhos mínimos entre todos os pares

Problema: Dado um digrafo com custo nos arcos, determinar, para cada par de vértices `s`, `t` o custo de um caminho mínimo de `s` a `t`

Esse problema pode ser resolvido aplicando-se V vezes o algoritmo Bellman-Ford

O consumo de tempo dessa solução é $O(V^2 A)$.

Um algoritmo mais eficiente foi descrito por Floyd, baseado em uma idéia de Warshall.

O algoritmo supõe que o digrafo não tem ciclo negativo

Problema dos caminhos mínimos entre todos os pares

Problema: Dado um digrafo com custo nos arcos, determinar, para cada par de vértices s, t o custo de um caminho mínimo de s a t

Esse problema pode ser resolvido aplicando-se V vezes o algoritmo Bellman-Ford

O consumo de tempo dessa solução é $O(V^2 A)$.

Um algoritmo mais eficiente foi descrito por Floyd, baseado em uma idéia de Warshall.

O algoritmo supõe que o digrafo não tem ciclo negativo

Problema dos caminhos mínimos entre todos os pares

Problema: Dado um digrafo com custo nos arcos, determinar, para cada par de vértices s, t o custo de um caminho mínimo de s a t

Esse problema pode ser resolvido aplicando-se V vezes o algoritmo Bellman-Ford

O consumo de tempo dessa solução é $O(V^2 A)$.

Um algoritmo mais eficiente foi descrito por Floyd, baseado em uma idéia de Warshall.

O algoritmo supõe que o digrafo não tem ciclo negativo

Programação dinâmica

$0, 1, 2, \ldots, \texttt{V}-1$ = lista dos vértices do digrafo

`custo[k][s][t]` = menor custo de um caminho de `s` a `t` usando vértices internos em $\{0, 1, \ldots, \texttt{k}-1\}$

Recorrência:

```
custo[0][s][t]  = G->adj[s][t]
custo[k][s][t]  = min{custo[k-1][s][t],
                      custo[k-1][s][k-1] + custo[k-1][k-1][t]}
```

Se o digrafo não tem ciclo negativo acessível a partir de `s`, então `custo[V][s][t]` é o menor custo de um **caminho simples** de `s` a `t`

Programação dinâmica

$0, 1, 2, \ldots,$ `V`-1 = lista dos vértices do digrafo

`custo[k][s][t]` = menor custo de um caminho de `s` a `t` usando vértices internos em $\{0, 1, \ldots,$ `k`$-1\}$

Recorrência:

```
custo[0][s][t]  = G->adj[s][t]
custo[k][s][t]  = min{custo[k-1][s][t],
                      custo[k-1][s][k-1] + custo[k-1][k-1][t]}
```

Se o digrafo não tem ciclo negativo acessível a partir de `s`, então `custo[V][s][t]` é o menor custo de um **caminho simples** de `s` a `t`

```
void floyd_warshall (Digraph G){
 1   Vertex s, t; double d;
 2   for (s=0; s < G->V; s++)
 3       for (t=0; t < G->V; t++)
 4           custo[0][s][t] = G->adj[s][t];
 5   for (k=1; k <=G->V; k++)
 6       for (s=0; s < G->V; s++)
 7           for (t=0; t < G->V; t++){
 8               custo[k][s][t]=custo[k-1][s][t];
 9               d=custo[k-1][s][k-1]
                          +custo[k-1][k-1][t];
10               if (custo[k][s][t] > d)
11                   custo[k][s][t] = d;
             }
         }
}
```

Consumo de tempo

O consumo de tempo da função `floyd_warshall1` é $O(V^3)$.

```
void floyd_warshall (Digraph G){
 1    Vertex s, t; double d;
 2    for (s=0; s < G->V; s++)
 3         for (t=0; t < G->V; t++)
 4              cst[s][t] = G->adj[s][t];
 5    for (k=1; k <= G->V; k++)
 6         for (s=0; s < G->V; s++)
 7              for (t=0; t < G->V; t++){
 8                   d=cst[s][k-1]+cst[k-1][t];
10                   if (cst[s][t] > d)
11                        cst[s][t] = d;
                }
          }
}
```

Relação invariante

No início de cada iteração da linha 5 vale que

$\texttt{cst}[\texttt{s}][\texttt{t}] = \texttt{custo}[\texttt{k}][\texttt{s}][\texttt{t}]$ = o menor custo de um caminho de $\texttt{s}$ a $\texttt{t}$ usando vértices internos em $\{0, 1, \ldots, \texttt{k}-1\}$

Novo resumo

função	consumo de tempo	observação
DAGmin	$O(V + A)$	digrafos acíclicos custos arbitrários
dijkstra	$O(A \lg V)$	custos ≥ 0, min-heap
	$O(V^2)$	custos ≥ 0, fila
bellman-ford	$O(V^3)$ $O(VA)$	digrafos densos digrafos esparsos
floyd-warshall	$O(V^3)$	digrafos sem ciclos negativos

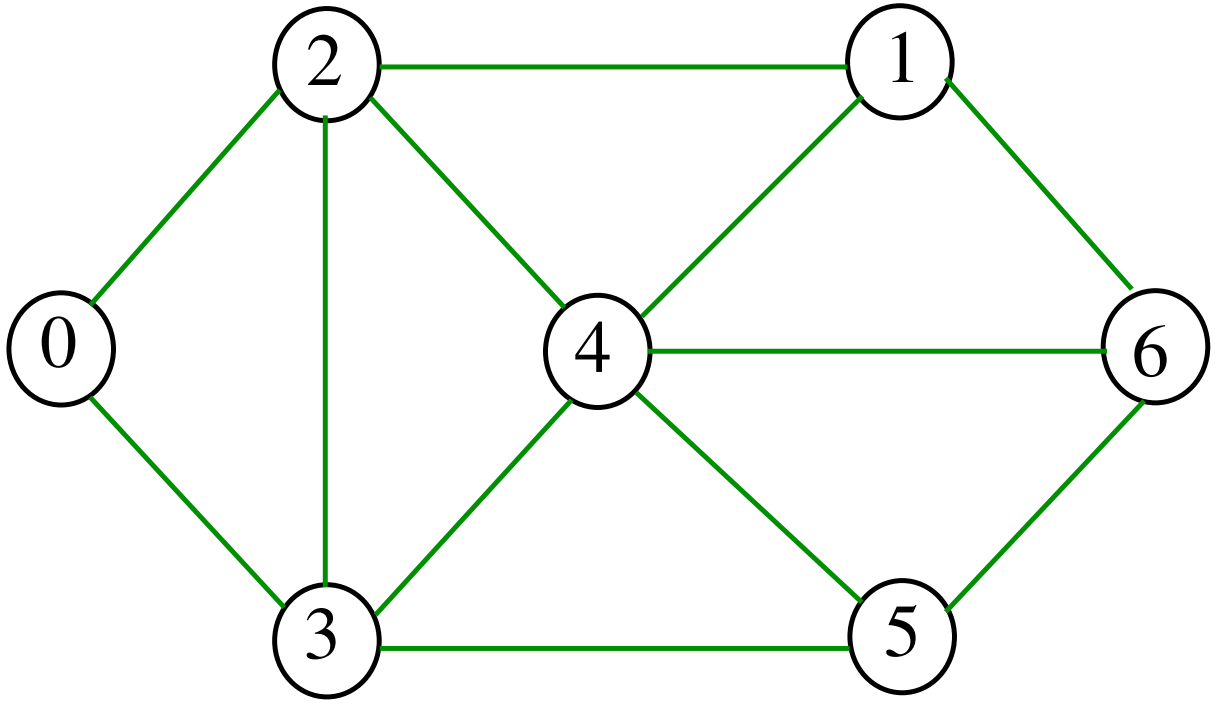
O problema SPT em digrafos com ciclos negativos é NP-difícil.

Árvores geradoras de grafos

Subárvores

Uma **subárvore** de um grafo `G` é qualquer árvore `T` que seja subgrafo de `G`

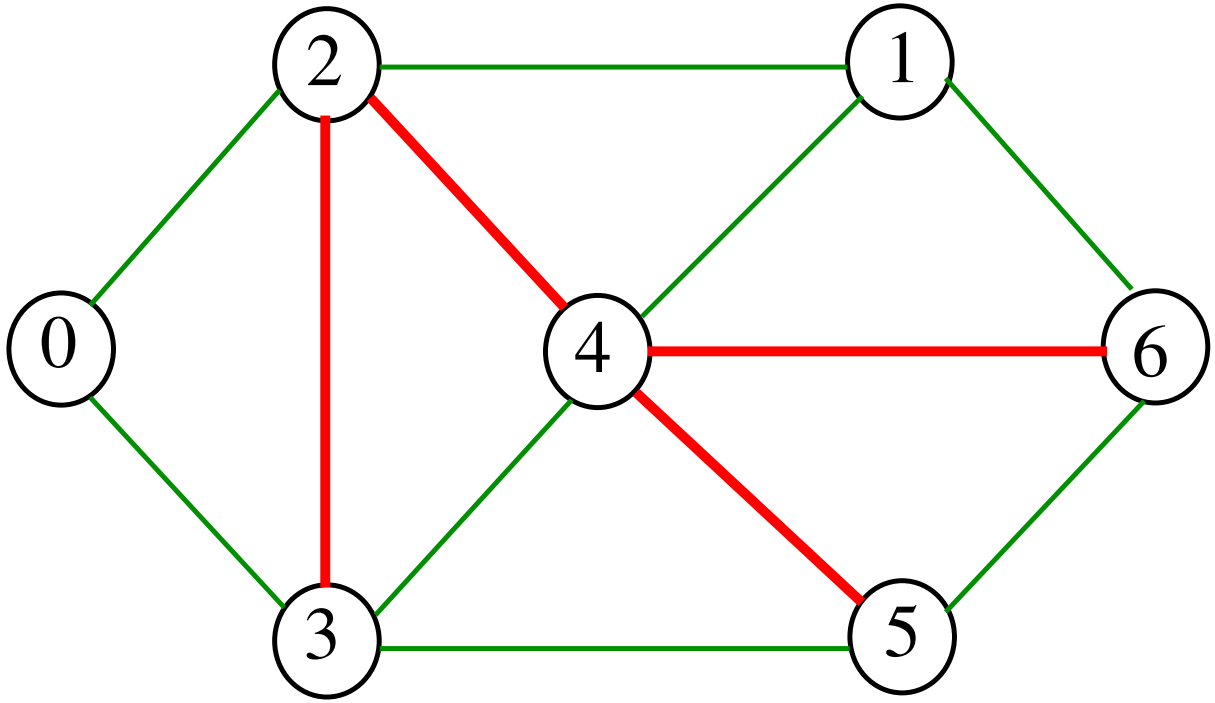
Exemplo:



Subárvores

Uma **subárvore** de um grafo `G` é qualquer árvore `T` que seja subgrafo de `G`

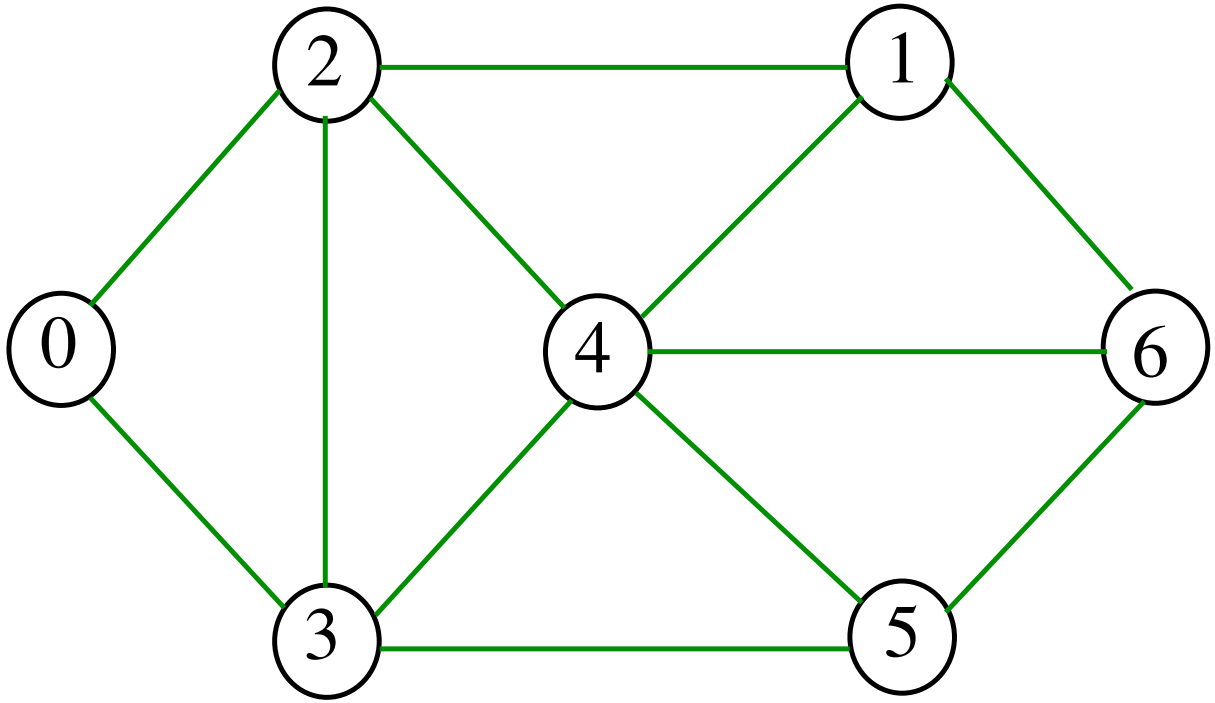
Exemplo: as arestas em vermelho formam uma subárvore



Árvores geradoras

Uma **árvore geradora** (= *spanning tree*) de um grafo é qualquer subárvore que contenha **todos** os vértices

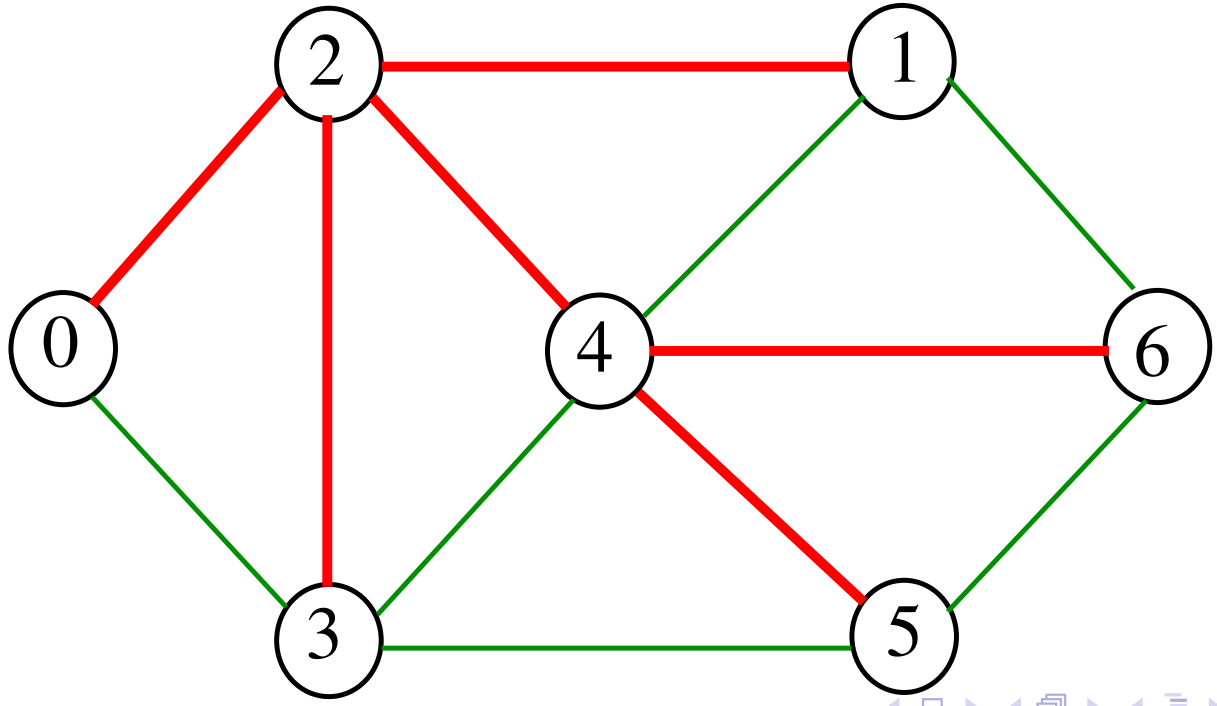
Exemplo:



Árvores geradoras

Uma **árvore geradora** (= *spanning tree*) de um grafo é qualquer subárvore que contenha **todos** os vértices

Exemplo: as arestas em vermelho formam uma árvore geradora

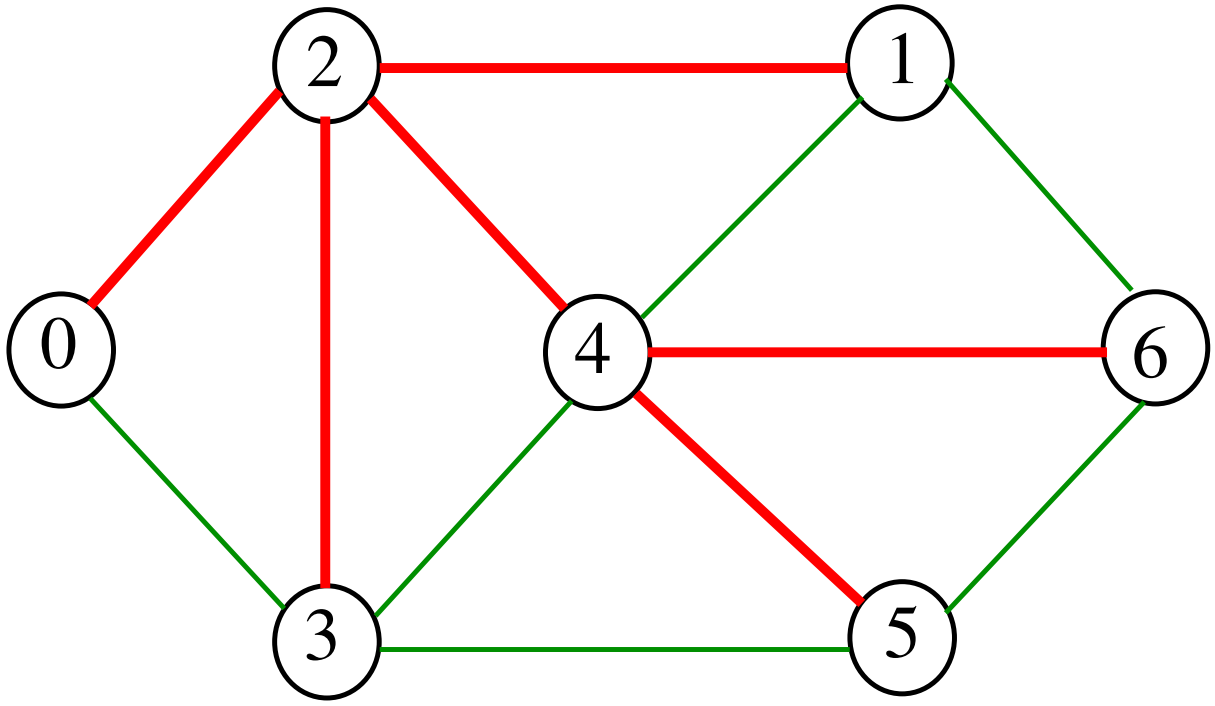


Árvores geradoras

Somente grafos conexos têm árvores geradoras

Todo grafo conexo tem uma árvore geradora

Exemplo:



Algoritmos que calculam árvores geradoras

É fácil calcular uma árvore geradora de um grafo conexo:

- a busca em profundidade e
- a busca em largura

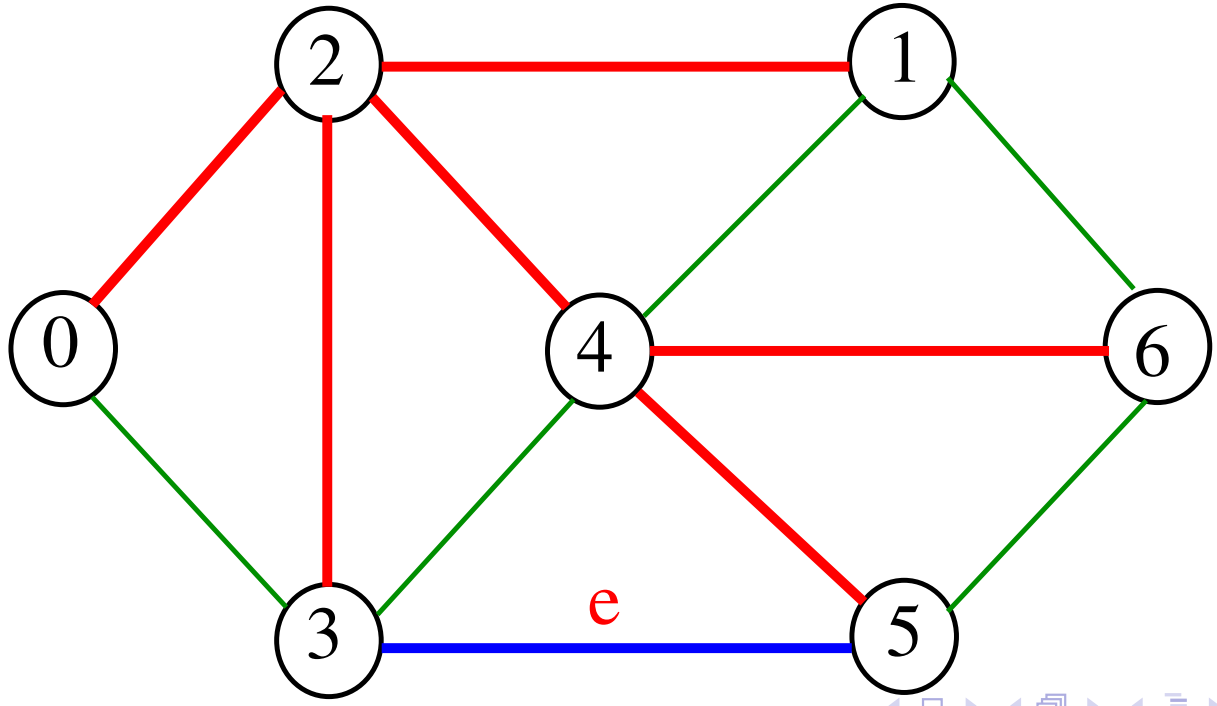
fazem isso.

Qualquer das duas buscas calcula uma arborescência que contém um dos arcos de cada aresta de uma árvore geradora do grafo

Primeira propriedade da troca de arestas

Seja T uma árvore geradora de um grafo G Para qualquer aresta e de G que não esteja em T, T+e tem um **único ciclo** não-trivial, o **ciclo fundamental** $C(T, e)$.

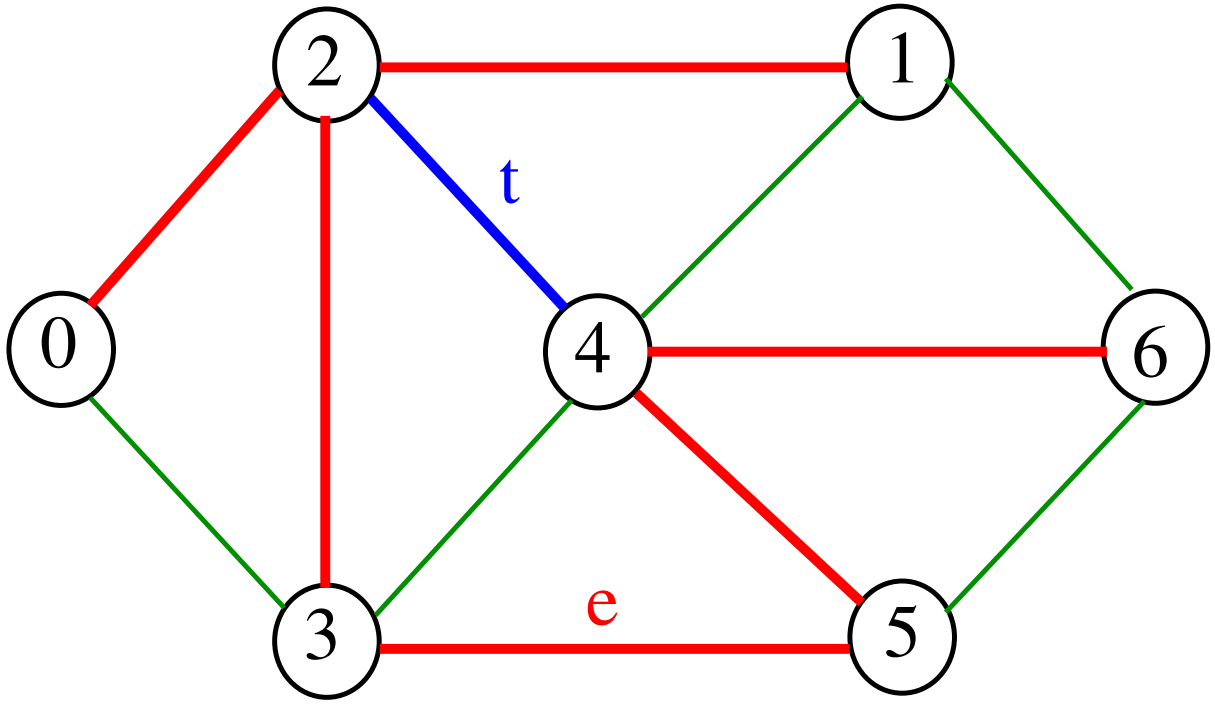
Exemplo: T+e



Primeira propriedade da troca de arestas

Seja T uma árvore geradora de um grafo G Para qualquer aresta t $\in C(T, e)$, T+e-t é uma **árvore geradora**

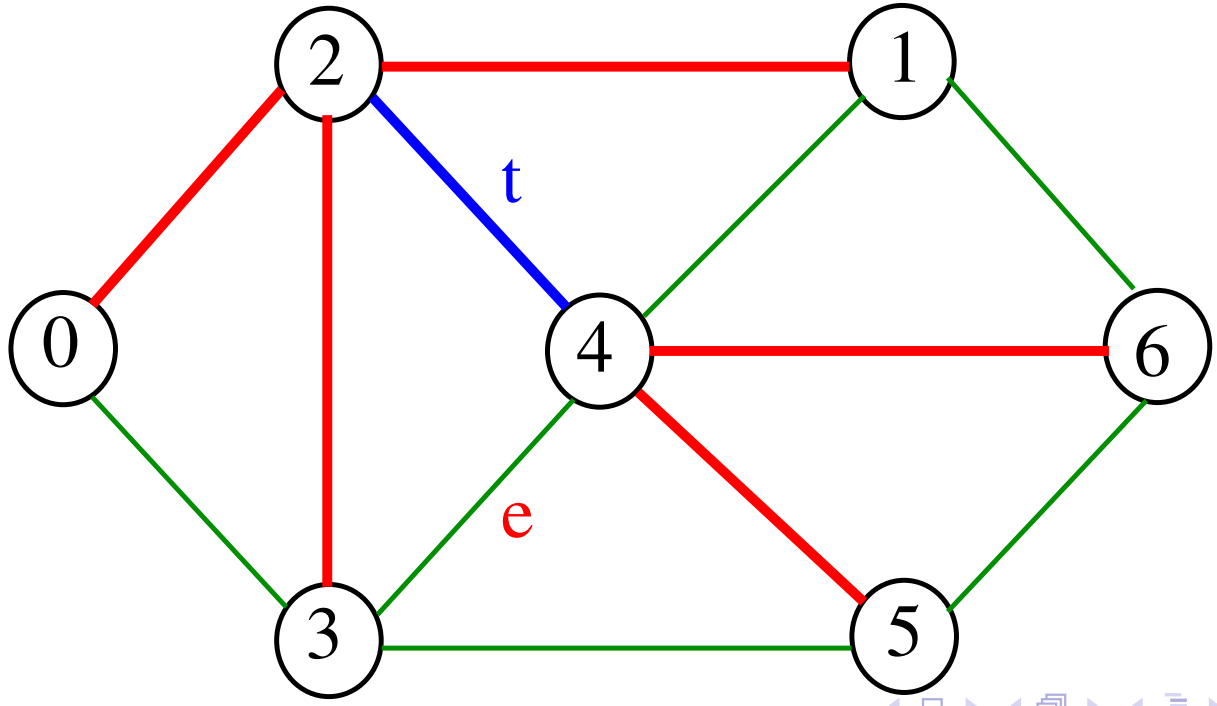
Exemplo: T+e-t



Segunda propriedade da troca de arestas

Seja T uma árvore geradora de um grafo G
Para qualquer aresta t de T e qualquer aresta e que atravesse o corte determinado por T-t, o grafo T-t+e é uma árvore geradora

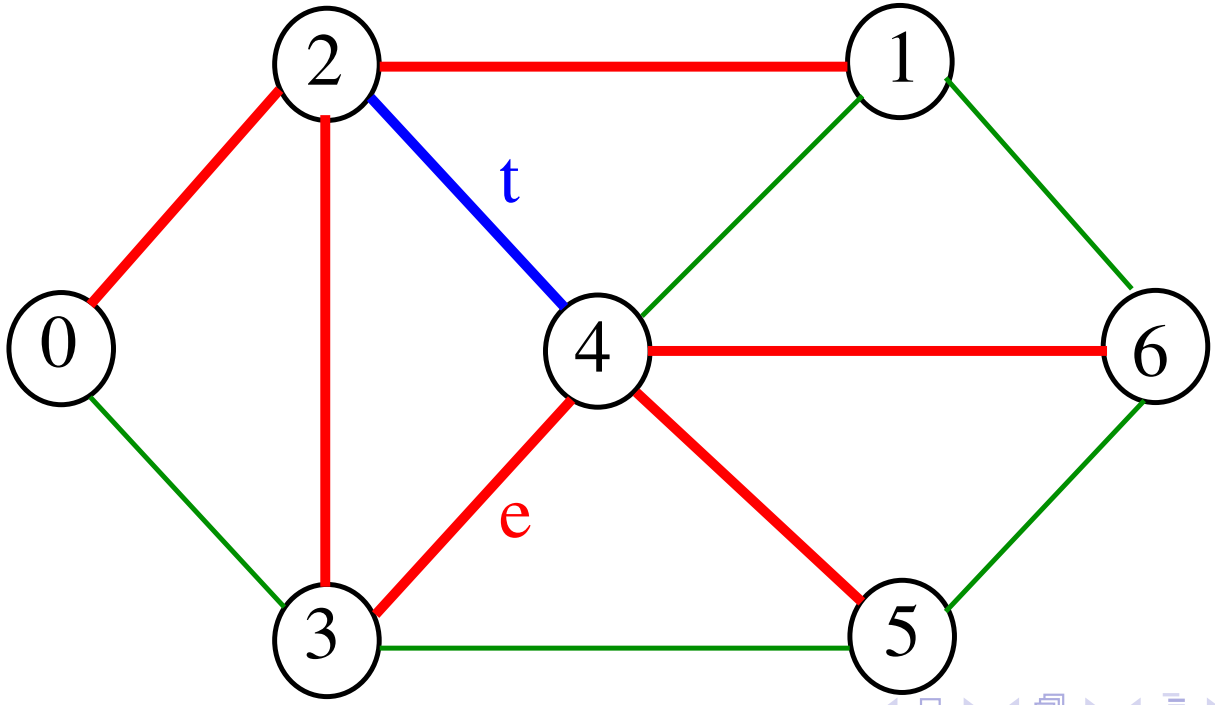
Exemplo: T-t



Segunda propriedade da troca de arestas

Seja T uma árvore geradora de um grafo G
Para qualquer aresta t de T e qualquer aresta e que atravesse o corte determinado por T-t, o grafo T-t+e é uma árvore geradora

Exemplo: T-t+e



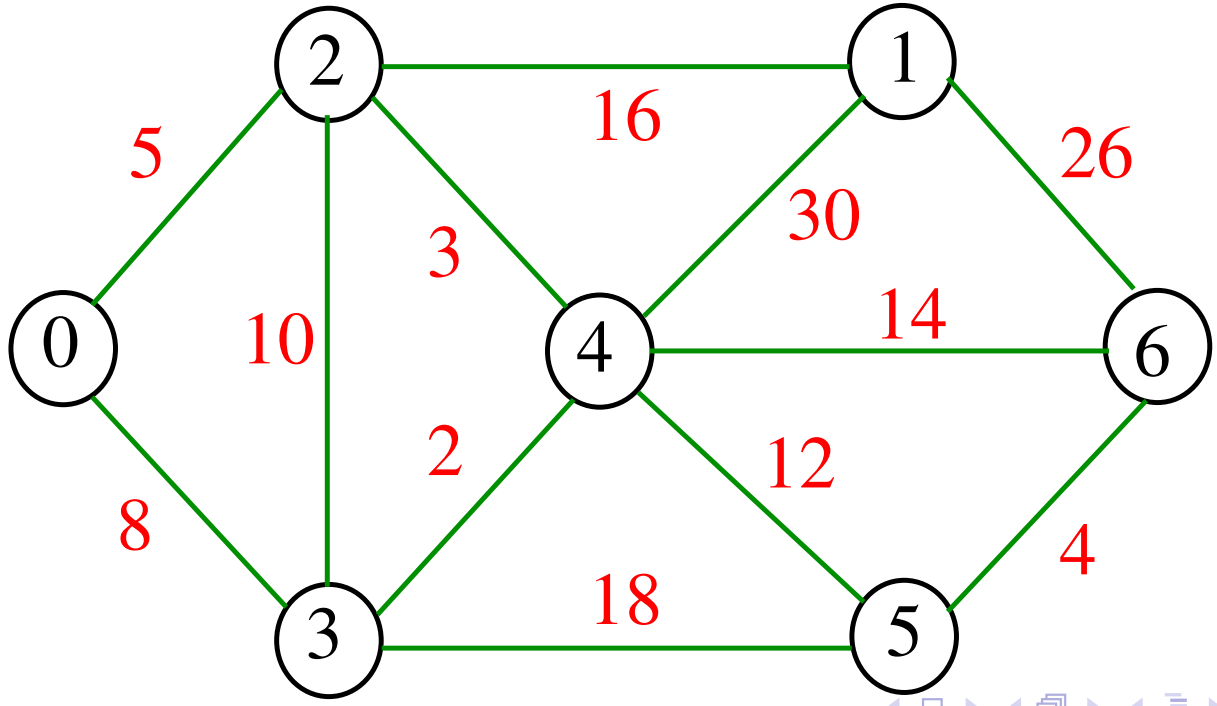
Árvores geradoras de custo mínimo

S 20.1 e 20.2

Árvores geradoras mínimas

Uma **árvore geradora mínima** (= *minimum spanning tree*), ou MST, de um grafo com custos nas arestas é qualquer árvore geradora do grafo que tenha custo mínimo

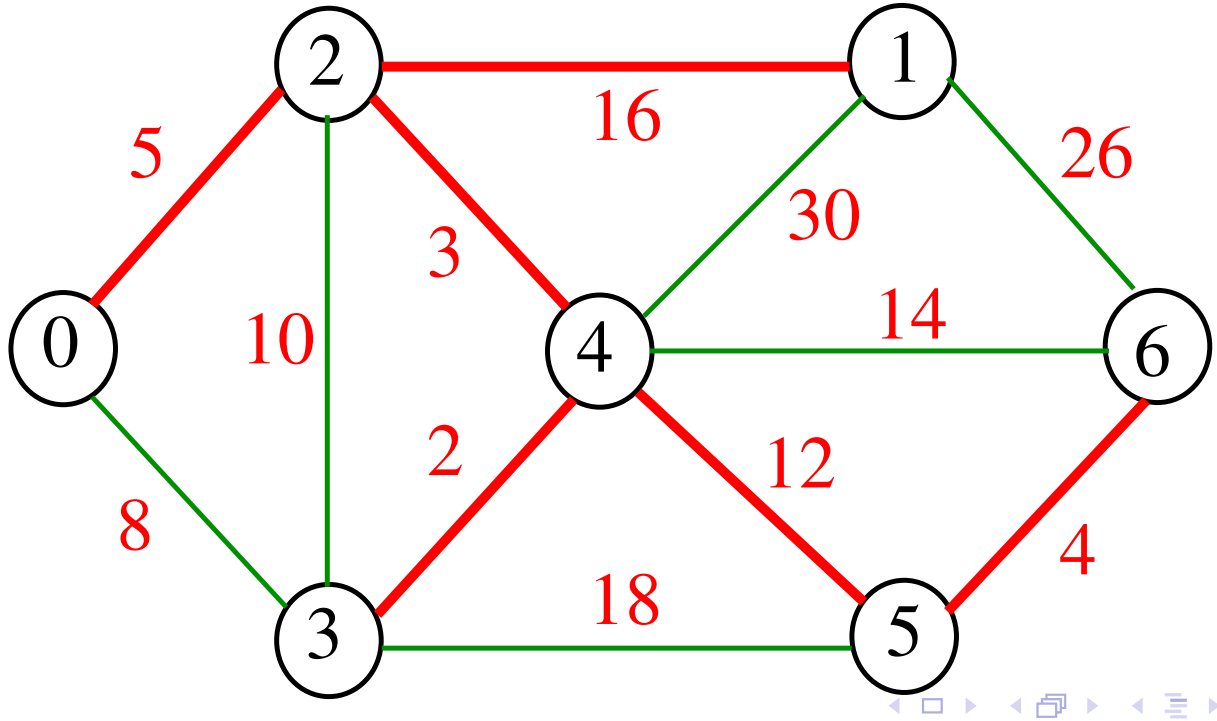
Exemplo: um grafo com custos nas arestas



Árvores geradoras mínimas

Uma **árvore geradora mínima** (= *minimum spanning tree*), ou MST, de um grafo com custos nas arestas é qualquer árvore geradora do grafo que tenha custo mínimo

Exemplo: MST de custo 42

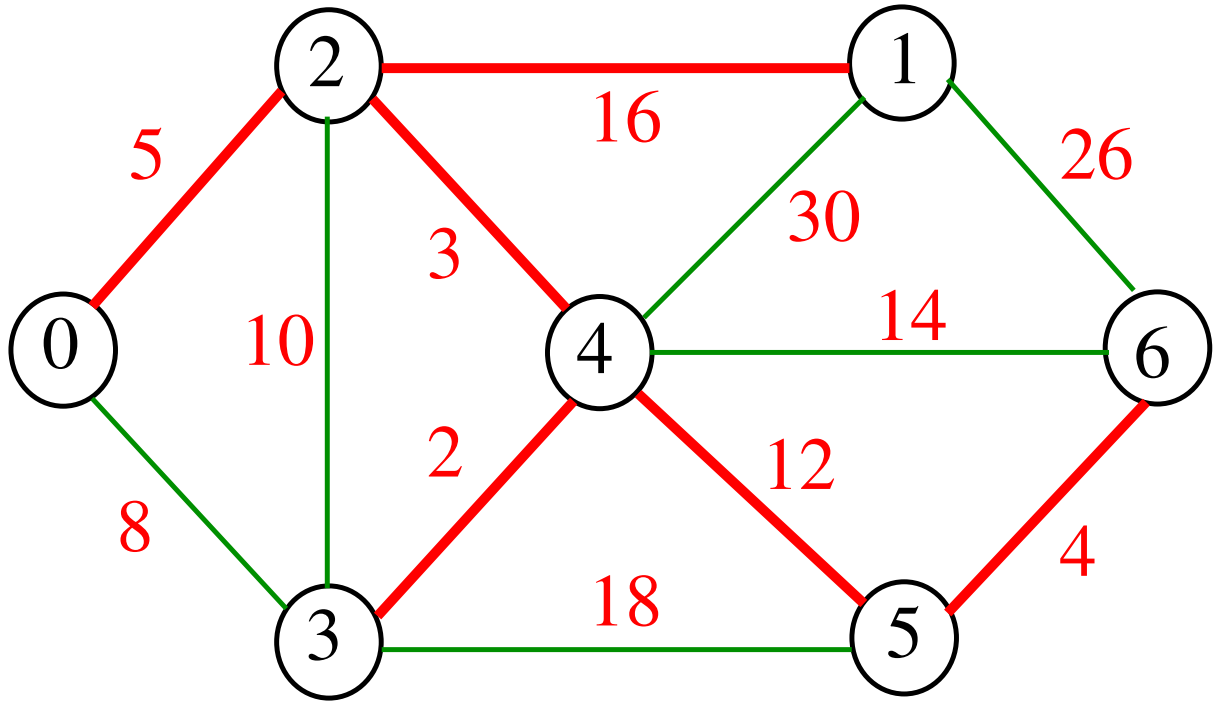


Problema MST

Problema: Encontrar uma MST de um grafo G com custos nas arestas

O problema tem solução se e somente se o grafo G é conexo

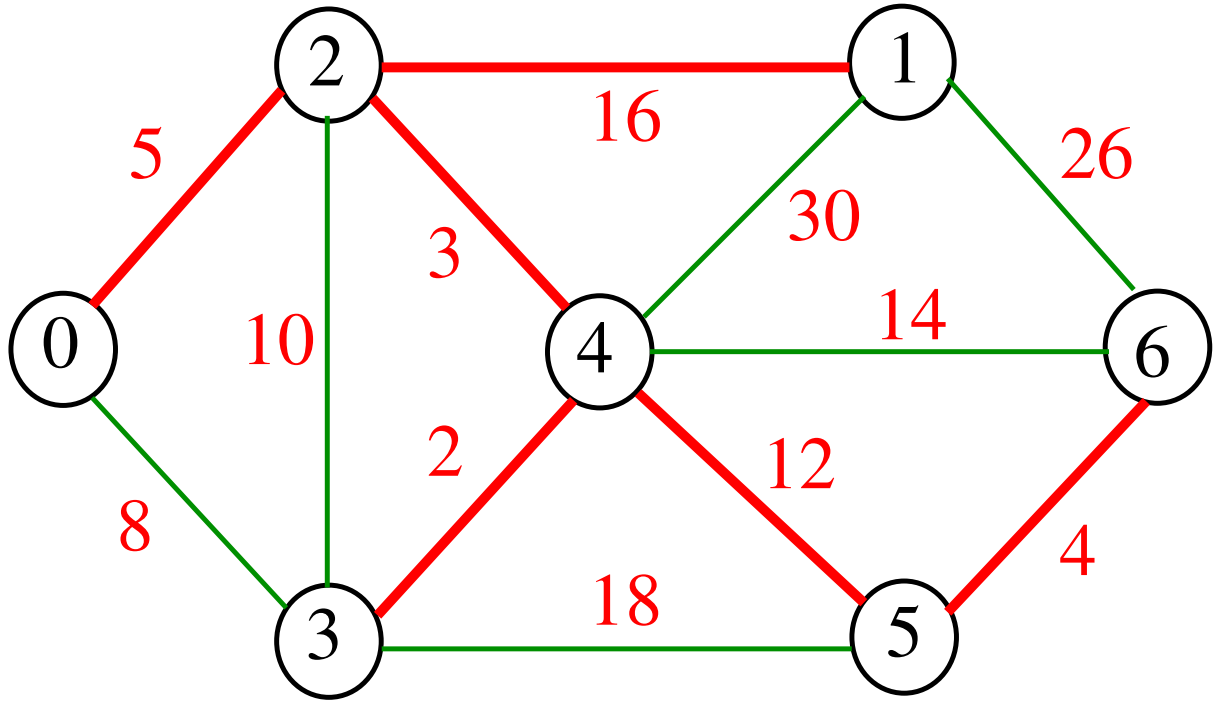
Exemplo: MST de custo 42



Propriedade dos ciclos

Condição de Otimalidade: Se T é uma MST então toda aresta e fora de T tem custo máximo dentre as arestas do único ciclo não-trivial em T+e

Exemplo: MST de custo 42



Demonstração da recíproca

Seja T uma árvore geradora satisfazendo a condição de otimalidade.

Vamos mostrar que T é uma MST.

Seja M uma MST tal que o número de arestas comuns entre T e M seja máximo.

Se $T = M$ não há o que demonstrar.

Suponha que $T \neq M$ e seja e uma aresta de custo mínimo dentre as arestas que estão em M mas não estão em T.

Seja d uma aresta qualquer que **não está** em M mas **está** no ciclo fundamental $C(T, e)$.

Demonstração da recíproca

Seja T uma árvore geradora satisfazendo a condição de otimalidade.

Vamos mostrar que T é uma MST.

Seja M uma MST tal que o número de arestas comuns entre T e M seja máximo.

Se $T = M$ não há o que demonstrar.

Suponha que $T \neq M$ e seja e uma aresta de custo mínimo dentre as arestas que estão em M mas não estão em T.

Seja d uma aresta qualquer que **não está** em M mas **está** no ciclo fundamental $C(T, e)$.

Continuação

Logo, $\texttt{custo}(\texttt{d}) \leq \texttt{custo}(\texttt{e})$ (1).
Seja f uma aresta qualquer em $C(\texttt{M}, \texttt{d}) - \texttt{T}$.
Como M é uma MST, $\texttt{custo}(\texttt{f}) \leq \texttt{custo}(\texttt{d})$ (2).
Pela escolha de e, $\texttt{custo}(\texttt{e}) \leq \texttt{custo}(\texttt{f})$ (3).
Juntando (1), (2) e (3), vem que

$$\texttt{custo}(\texttt{d}) = \texttt{custo}(\texttt{f}) = \texttt{custo}(\texttt{e})$$

Mas então, $\texttt{M}-\texttt{f}+\texttt{d}$ é uma MST que tem o mesmo custo que M, logo é mínima. Por outro lado, tem uma aresta a mais em comum com T do que M. Isso contradiz a escolha de M.

Portanto, $\texttt{T} = \texttt{M}$, o que mostra que T é uma MST.

Propriedade dos cortes

Condição de Otimalidade: T é uma MST se e somente se cada aresta t de T é uma aresta mínima dentre as que atravessam o corte determinado por T-t

Exemplo: MST de custo 42