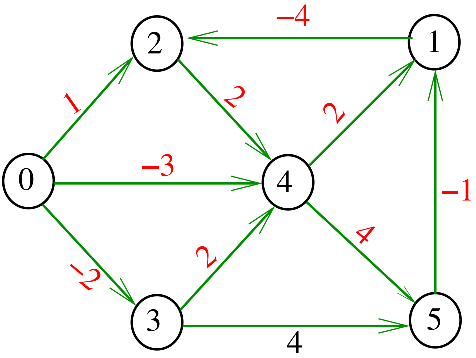
Melhores momentos

AULA 18

Problema da SPT

Problema: Dado um vértice s de um digrafo com custos (possivelmente negativos) nos arcos, encontrar uma SPT com raiz s

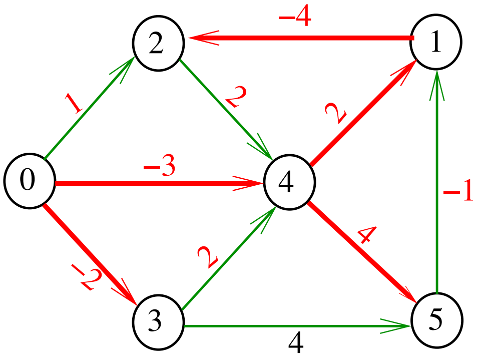
Entra:



Problema da SPT

Problema: Dado um vértice s de um digrafo com custos (possivelmente negativos) nos arcos, encontrar uma SPT com raiz s

Sai:



Fato

O algoritmo de Dijkstra não funciona para digrafos com custos negativos, mesmo que o digrafo seja acíclico.

Programação dinâmica

Propriedade (da subestrutura ótima)
Se G é um digrafo com custo (possivelmente negativos) nos arcos, sem **ciclos negativos** e v_0-v_1-v_2-$\ldots$-v_k é um caminho mínimo então v_i-v_{i+1}-$\ldots$-v_j é um caminho mínimo para $0 \leq i \leq j \leq k$

Bellman-Ford

custo[k][w] = menor custo de um caminho de
s a w com $\leq$ k arcos.

Recorrência:

$$
\begin{aligned}
\texttt{custo}[0][\texttt{s}] &= 0 \\
\texttt{custo}[0][\texttt{w}] &= \texttt{maxCST},\ \texttt{w} \neq \texttt{s} \\
\texttt{custo}[\texttt{k}][\texttt{w}] &= \min\{\texttt{custo}[\texttt{k}-1][\texttt{w}], \\
&\quad \min\{\texttt{custo}[\texttt{k}-1][\texttt{v}] + \texttt{G->adj}[\texttt{v}][\texttt{w}]\}\}
\end{aligned}
$$

Se o digrafo não tem ciclo negativo acessível a partir de s, então custo[V-1][w] é o menor custo de um **caminho simples** de s a w

Ciclos negativos

Se custo[V][v] $\neq$ custo[V-1][v], então G tem um ciclo negativo alcançável a partir de s.

Se G tem um ciclo negativo alcançável a partir de s, então custo[V][v] $\neq$ custo[V−1][v] para algum vértice v.

Consumo de tempo

O consumo de tempo do algoritmo de Bellman-Ford é $O(V^3)$.

Se o grafo for representado através de vetor de listas de adjacência e implementarmos uma fila de tal forma que cada operação consuma tempo constante teremos:

O consumo de tempo do algoritmo de Bellman-Ford é $O(VA)$.

Ciclos negativos

Problema: Dado um digrafo com custos nos arcos, decidir se o digrafo possui algum ciclo negativo.

Uma adaptação da função `bellman_ford` decide se um dado digrafo com custos nos arcos possui algum ciclo negativo. O consumo de tempo dessa função adaptada é $O(VA)$.

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 esparços

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

AULA 19

Algoritmo de 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`

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 de Bellman-Ford

O consumo de tempo dessa solução é $\mathrm{O}(\mathtt{V}^2\mathtt{A})$.

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 de Bellman-Ford

O consumo de tempo dessa solução é $\mathrm{O}(V^2 A)$.

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

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

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

Programação dinâmica

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

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

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){
    Vertex s, t; double d;
    for (s=0; s < G->V; s++)
        for (t=0; t < G->V; t++)
            custo[0][s][t] = G->adj[s][t];
    for (k=1; k <=G->V; k++)
        for (s=0; s < G->V; s++)
            for (t=0; t < G->V; t++){
                custo[k][s][t]=custo[k-1][s][t];
                d=custo[k-1][s][k-1]
                        +custo[k-1][k-1][t];
                if (custo[k][s][t] > d)
                    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){
    Vertex s, t; double d;
    for (s=0; s < G->V; s++)
        for (t=0; t < G->V; t++)
            cst[s][t] = G->adj[s][t];
    for (k=1; k <= G->V; k++)
        for (s=0; s < G->V; s++)
            for (t=0; t < G->V; t++){
                d=cst[s][k-1]+cst[k-1][t];
                if (cst[s][t] > d)
                    cst[s][t] = d;
            }
        }
}
```

Relação invariante

No início de cada iteração da linha 5 vale que

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

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 esparços
floyd-warshall	$O(V^3)$	digrafos sem ciclos negativos

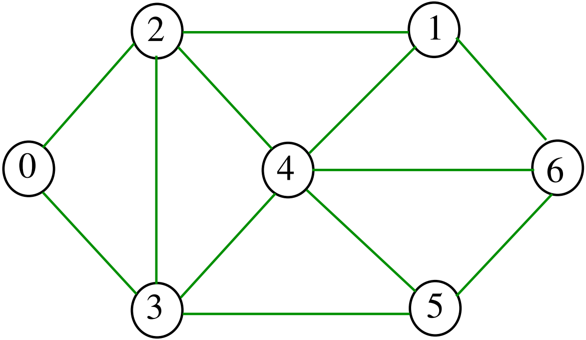
O problema SPT em digrafos com ciclos negativos é NP-difícil.

Árvores geradoras de grafos

Subárvores

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

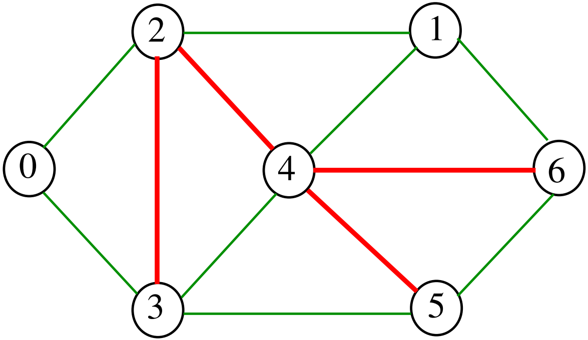
Exemplo:



Subárvores

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

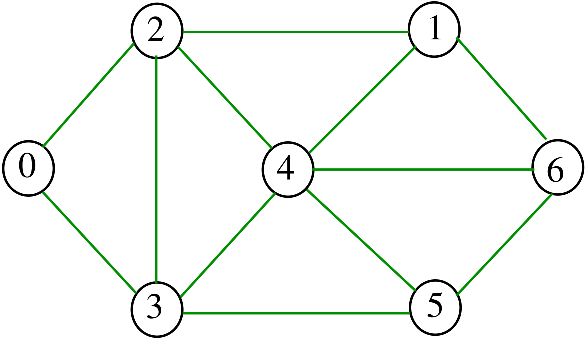
Exemplo: as aretas em vermelho formam uma subárvore



Árvores geradoras

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

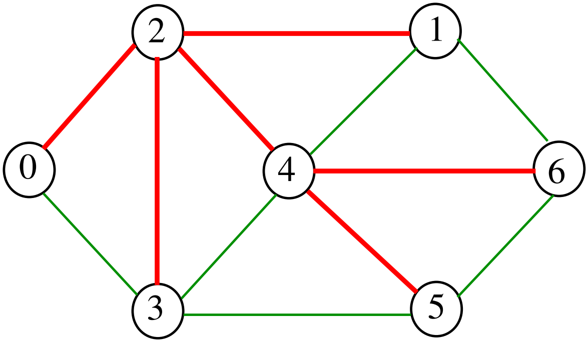
Exemplo:



Árvores geradoras

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

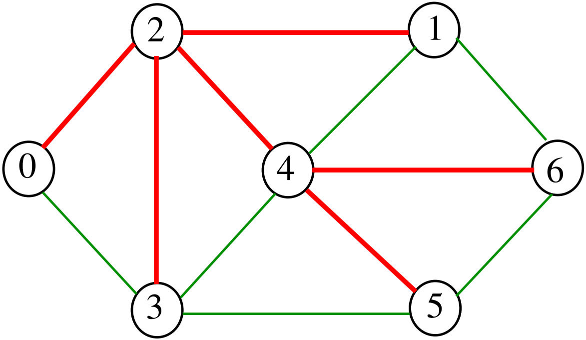
Exemplo: as aretas em vermelho formam uma árvore geradora



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

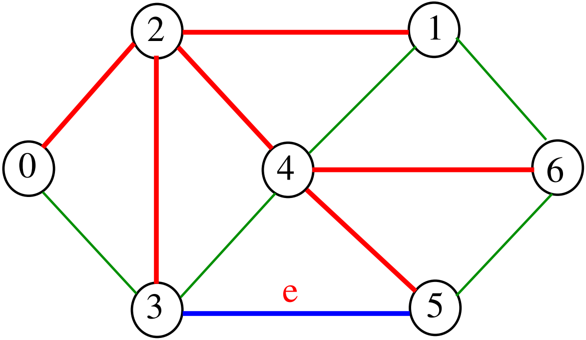
fazem isso.

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

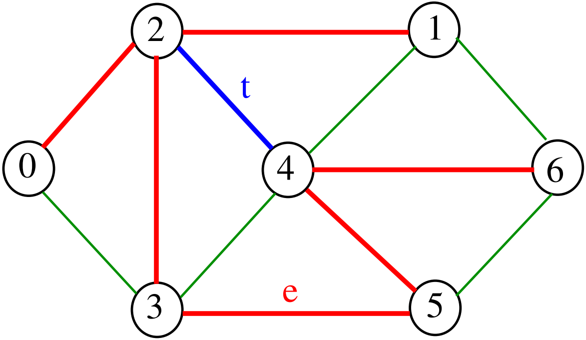
Exemplo: T+e



Primeira propriedade da troca de arestas

Seja T uma árvore geradora de um grafo G Para qualquer aresta t desse ciclo, T+e-t uma **árvore geradora**

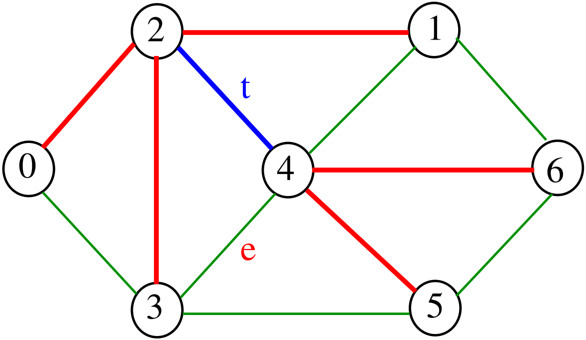
Exemplo: T+e-t



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

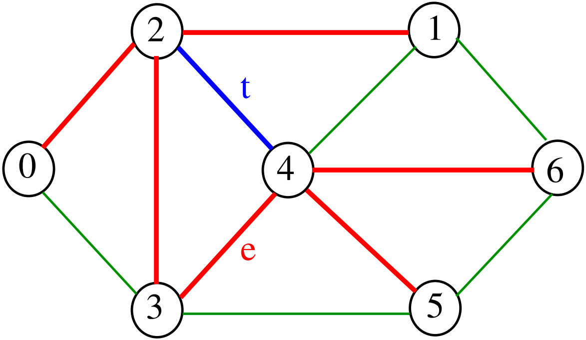
Exemplo: T-t



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



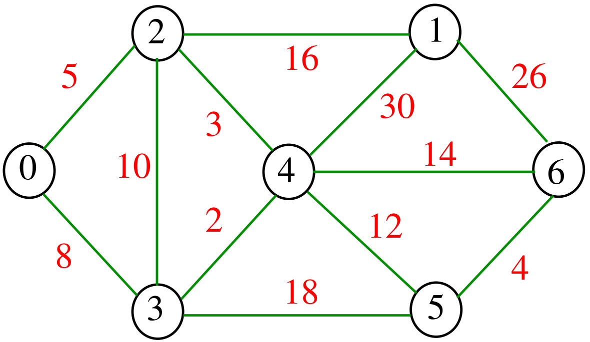
Árvores geradoras de custo mínimo

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

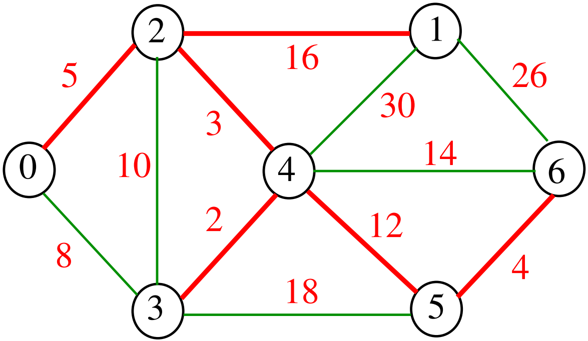
Exemplo: um grafo com custos nas aretas



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

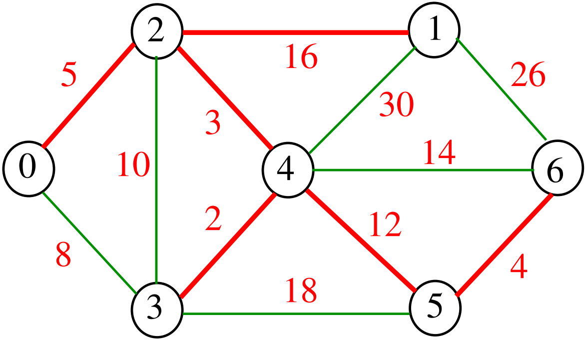
Exemplo: MST de custo 42



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