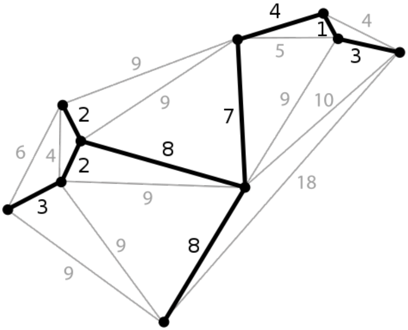
Aula 26: Grafos: árvores geradoras mínimas

Árvores de extensão mínima

David Déharbe
Programa de Pós-graduação em Sistemas e Computação
Universidade Federal do Rio Grande do Norte
Centro de Ciências Exatas e da Terra
Departamento de Informática e Matemática Aplicada

Plano



Introdução

Um algoritmo abstrato

Algoritmo de Kruskal

Algoritmo de Prim

Referência: Cormen, cap 24.

Introdução

- Exemplo
  - rede ótica interligando diferentes institutos
  - C_{AB} custo para conectar instituto A ao instituto B
  - identificar qual a rede que minimiza os custos
- otimização combinatória: árvore geradora mínima
  minimum spanning tree

Formalização

- entrada: grafo não dirigido com pesos $G = (V, E, w)$
- saída: T subconjunto acíclico de E tal que
    1. $w(T) = \sum_{e \in T} w(e)$ é o menor possível
    2. T connecta todos os elementos de V
- T acíclico $\Rightarrow$ T é uma árvore
- T connecta todos os vértices: é uma cobertura de G
- a soma dos pesos das arestas é o mínimo possível

Abordagem algorítmica

- abordagem gulosa
  - solução parcial pode ser estendida a uma solução completa
  - a solução é construída incrementalmente
  - uma sequência de decisões localmente ótimas gera uma solução global ótima
- dois algoritmos
  - Vojtech Jarník (1930), Robert Prim (1957), Edsger Dijkstra (1959)
  - Joseph Kruskal (1956)
  - ambos são casos particulares de um algoritmo abstrato

PPgSC

Crescendo uma árvore geradora mínima

Algoritmo abstrato

- A: sub-conjunto de uma árvore geradora mínima
  - A não pode ter ciclos
  - o grafo induzido por A não precisa ser conectado
  - A forma uma floresta de árvores
- iterativamente uma aresta (u, v) é adicionada a A
  - (u, v) é segura se $A \cup \{(u, v)\}$ é um sub-conjunto de uma árvore geradora mínima.
- A continua um sub-conjunto de uma árvore geradora mínima
- término: A cobre todo o grafo

MST-ABSTRACT(G)

$A = \emptyset$
// invariante: $A \subseteq$ uma árvore geradora mínima
while A não forma uma árvore geradora mínima
    Escolher uma aresta (u, v) segura para A
    $A = A \cup \{(u, v)\}$
return A

MST-ABSTRACT(G)

$A = \emptyset$

// invariante: $A \subseteq$ uma árvore geradora mínima

while A não forma uma árvore geradora mínima

Escolher uma aresta (u, v) segura para A

$A = A \cup \{(u, v)\}$

return A

Questão: Como identificar uma aresta segura para A?

Identificação de aresta segura

Roteiro

- Definições:
  - corte de um grafo não dirigido
  - aresta leve
- Propriedades
  - teorema: arestas leves e subconjunto de árvores geradoras mínimas
  - corolário (fundamentação dos algoritmos de Prim e Kruskal): componentes conectados e arestas leves.

Definições

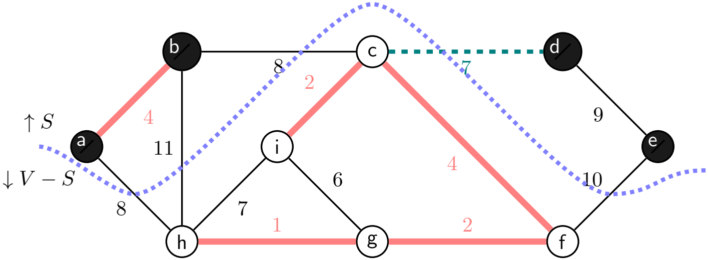
Definição (Corte)

Seja $G = (V, E)$ um grafo não dirigido. Um corte de G é uma partição $(S, V - S)$ dos vértices.

Definição

Seja $G = (V, E)$ um grafo não dirigido e $(S, V - S)$ um corte de G.

- A aresta (u, v) cruza o corte se $u \in S$ e $v \in V - S$.
- O conjunto de arestas $A \subseteq E$ é compatível com o corte se nenhuma aresta de A cruza o corte.
- Se G é um grafo com pesos, a aresta $(u, v) \in E$ é uma aresta leve se atravessa o corte e tem o menor peso entre todas as arestas cruzando o corte.

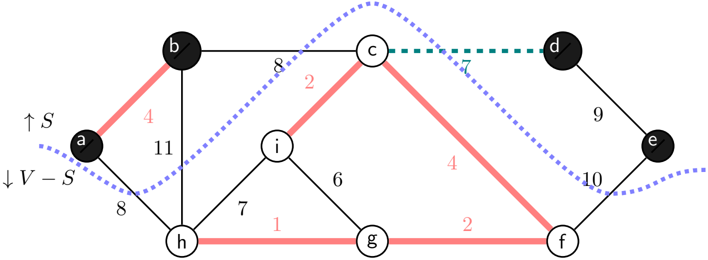


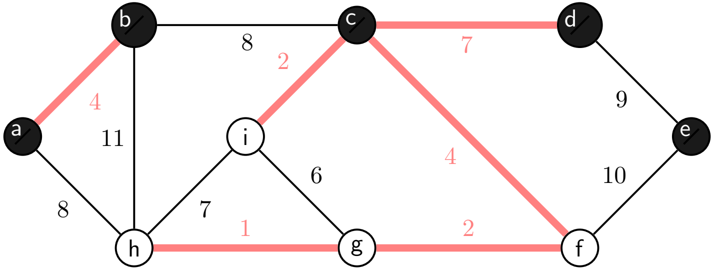
- corte, aresta cruzando corte, aresta leve, conjunto de arestas compatível com o corte

O teorema seguinte fornece um critério para escolher arestas no algoritmo abstrato:

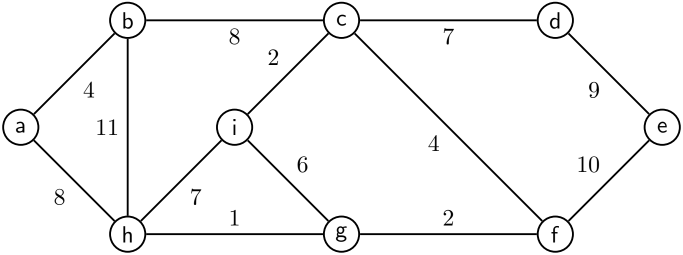
Teorema (Critério da arestas leve)

Seja $G = (V, E, w)$ um grafo não dirigido conectado com pesos. Seja $A \subseteq E$, tal que A é um sub-conjunto de uma árvore geradora mínima. Seja $(S, V - S)$ um corte compatível com A, e (u, v) uma aresta leve cruzando $(S, V - S)$. Então (u, v) é seguro para A.


↑ S
↓ $V - S$
a
b
c
d
e
f
g
h
i
4
8
11
8
2
7
7
9
10
6
4
1
2


a
b
c
d
e
f
g
h
i
4
8
11
8
2
7
7
9
10
6
4
1
2

Pratique



$A = \emptyset$
while A não forma uma árvore geradora mínima
    Escolher uma aresta (u, v) segura para A
    $A = A \cup \{(u, v)\}$

(u, v) é seguro para A:

- A é um sub-conjunto de uma árvore geradora mínima.
- A é compatível com $(S, V - S)$,
- (u, v) uma aresta leve cruzando $(S, V - S)$.

Teorema (Critério da arestas leve)

Seja $G = (V, E, w)$ um grafo não dirigido conectado com pesos. Seja $A \subseteq E$, tal que A é um sub-conjunto de uma árvore geradora mínima. Seja $(S, V - S)$ um corte com o qual A é compatível, e (u, v) uma aresta leve cruzando $(S, V - S)$. Então (u, v) é seguro para A.

Teorema (Critério da arestas leve)

Seja $G = (V, E, w)$ um grafo não dirigido conectado com pesos. Seja $A \subseteq E$, tal que A é um sub-conjunto de uma árvore geradora mínima. Seja $(S, V - S)$ um corte com o qual A é compatível, e (u, v) uma aresta leve cruzando $(S, V - S)$. Então (u, v) é seguro para A.

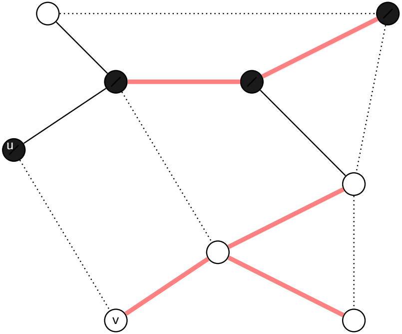
Demonstração.

- hipótese 1: $A \subseteq T$, T é uma árvore geradora mínima
- conclusão: $\exists T' \cdot A \cup \{(u, v)\} \subseteq T'$, T' árvore geradora mínima
- Duas possibilidades
    1. Se $(u, v) \in T$, então $T' = T$
    2. Se $(u, v) \notin T$, vamos construir T'

PPgSC

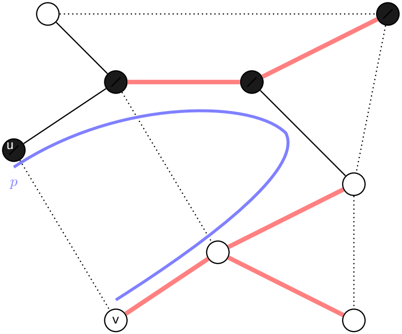
Demonstração.

- $A \subseteq T$, T é uma árvore geradora mínima, $(u, v) \notin T$,
- hipótese 2: (u, v) cruza $(S, V - S)$.



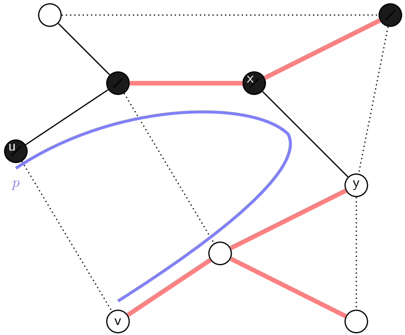
Demonstração.

- em T, $u \rightsquigarrow v$, seja p o caminho: $p, (u, v)$ formam um ciclo



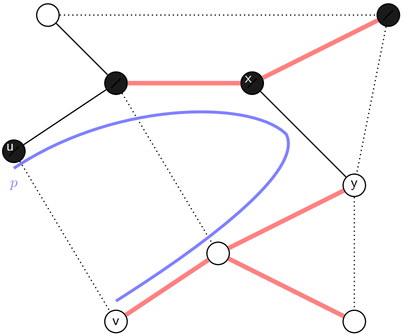
Demonstração.

- existe $(x, y) \in p$ tal que (x, y) cruza $(S, V - S)$.
- hipótese 3: A é compatível com $(S, V - S)$, logo $(x, y) \notin A$.



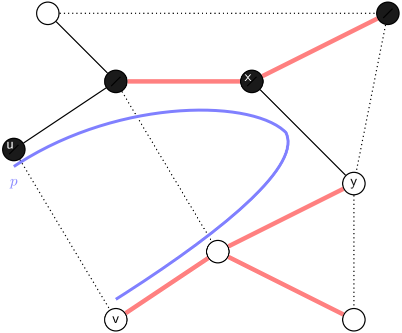
Demonstração.

- $T - \{(x, y)\}$ resulta em duas árvores não conexas
- $T' = (T - \{(x, y)\}) \cup \{(u, v)\}$ é uma árvore geradora



Demonstração.

- hipótese 4: (u, v) é uma aresta leve
- logo $w(u, v) \leq w(x, y)$, e $w(T') \leq w(T)$: T' é árvore geradora mínima.



Corolário

Seja $G = (V, E, w)$ um grafo não dirigido conectado com pesos. Seja $A \subseteq E$, tal que A é um sub-conjunto de uma árvore geradora mínima. Seja C uma árvore na floresta $G_A = (V, A)$, Se (u, v) é uma aresta leve conectando C a uma outra árvore de G_A, então (u, v) é segura para A.

Demonstração.

- Por definição, A é compatível com o corte $(C, V - C)$.
- A aresta (u, v) é uma aresta leve cruzando $(C, V - C)$
- O teorema diz que (u, v) é uma aresta segura para A.

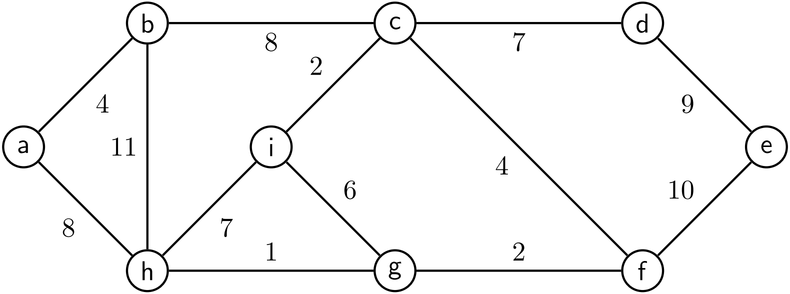
- instancia o algoritmo abstrato
- mantem uma floresta de sub-conjuntos de árvore geradora mínima
  - inicialmente cada vértice forma uma árvore individualmente
- abordagem gulosa: escolha a menor aresta que connecta duas árvores da floresta
  - decrementa o número de árvores na floresta
- término: a floresta tem uma árvore, geradora mínima
- abordagem gulosa (*greedy*)

Algoritmo de Kruskal

MST-KRUSKAL(G)

$A = \emptyset$
for $v \in G.V$
    MAKE-SET(v)
Ordena $G.E$ por peso crescente
for $(u, v) \in G.E$, em ordem crescente de peso
    if FIND-SET$(u) \neq$ FIND-SET(v)
        $A = A \cup \{(u, v)\}$
        UNION(u, v)

Algoritmo de Kruskal
Pratique



- Estrutura de dados:
  - conjuntos disjuntos com união por tamanho e compressão de caminho
  - melhor complexidade assintótica conhecida
- Inicialização: $\Theta(V)$
- Ordenação das arestas: $O(E \lg E)$
- $\Theta(E)$ operações sobre florestas de árvore: $O(E \lg E)$
- Grafo conectado $E \in \Omega(V)$, $E \in O(V^2)$
- Total: $O(E \lg E)$

Algoritmo de Prim

Princípios

- A forma uma (única) árvore
- a cada iteração uma aresta segura é adicionada a A
- a aresta incluída é aquela que soma o menor peso
- abordagem gulosa
- ponto principal: como identificar eficazmente a aresta que soma o menor peso
  - fila de prioridade dos vértices que não estão em A
  - prioridade de v: $v.key$ menor distância do vértice até um vértice de A
  - $v.up$ vértice de A ao qual o v é conectado quando $(v, v.up)$ é acrescentado a A.

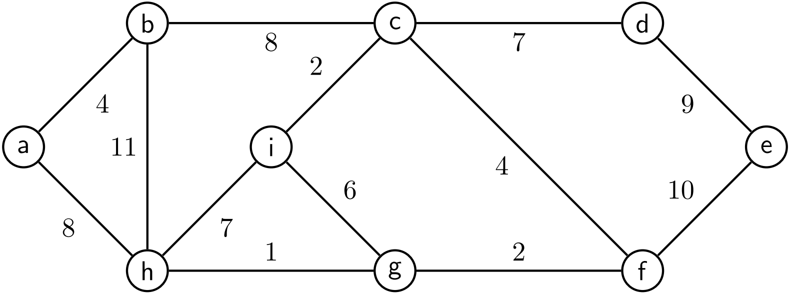
PPgSC

Algoritmo de Prim

```
MST-PRIM(G)
selecionar r ∈ V qualquer
for u ∈ G.V
    u.key = ∞
r.key = 0
r.up = NIL
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    for v ∈ u.adj
        if v ∈ Q e w(u, v) < v.key
            v.up = u
            v.key = w(u, v)
```

Ilustração

Algoritmo de Prim



Complexidade

Algoritmo de Prim

```
MST-PRIM(G)
selecionar r ∈ V qualquer
for u ∈ G.V // Θ(V)
    u.key = ∞
r.key = 0
r.up = NIL
Q = G.V // O(V)
while Q ≠ ∅ // Θ(V) vezes
    u = EXTRACT-MIN(Q) // O(lg V)
    for v ∈ u.adj // total: Θ(E)
        if v ∈ Q e w(u, v) < v.key
            v.up = u // Θ(1)
            v.key = w(u, v) // O(lg v)
```

MST-PRIM(G)

selecionar $r \in V$ qualquer
for $u \in G.V$ // $\Theta(V)$
    $u.key = \infty$
$r.key = 0$
$r.up = \text{NIL}$
$Q = G.V$ // $O(V)$
while $Q \neq \emptyset$ // $\Theta(V)$ vezes
    $u = \text{EXTRACT-MIN}(Q)$ // $O(\lg V)$
    for $v \in u.adj$ // total: $\Theta(E)$
        if $v \in Q$ e $w(u,v) < v.key$
            $v.up = u$ // $\Theta(1)$
            $v.key = w(u,v)$ // $O(\lg v)$

$$O(E \lg V + V \lg V) = O(E \lg V)$$