# GRAFOS – ÁRVORES DE CUSTO MÍNIMO

Gustavo Carvalho
(ghpc@cin.ufpe.br)

Universidade Federal de Pernambuco
Centro de Informática, 50740-560, Brazil

# Agenda

# Introdução

Como conectar *n* pontos da melhor forma possível, de forma que exista um caminho entre dois pontos quaisquer?

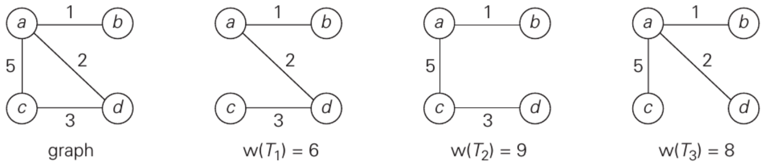
Aplicações

- Projeto de diferentes tipos de redes
- Fins de classificação
- Construção de soluções aproximadas

# Introdução

Árvore geradora de custo mínimo (*mininum spanning tree*)

- Seja $G$ um grafo conectado e não-dirigido, uma árvore geradora é um subgrafo acíclico e conectado de $G$ com todos os vértices de $G$. Se $G$ for ponderado, a árvore geradora de custo mínimo é aquela com o menor custo (soma dos pesos das arestas).



[1]

[1] Fonte: A. Levitin. Introduction to the Design and Analysis of Algorithms. 2011.

# Algoritmo de Prim

Outro exemplo de algoritmo guloso

- Inicialmente, a árvore $T$ é formada por um vértice arbitrário $v$
- Em cada passo, escolhe um vértice $v'$ que não está em $T$, mas que se liga a algum vértice de $T$, cujo peso da aresta $(v, v')$ é o menor dentre as opções possíveis

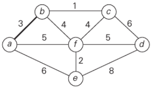
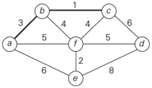
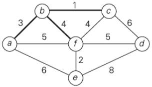
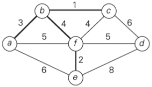
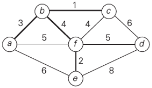
Similar ao algoritmo de Dijkstra

- Busca o próximo vértice mais próximo de qualquer vértice já em $T$
- Permite pesos negativos

Eficiência temporal: igual à do Dijkstra

- Matriz s/ heap: $\Theta(\mid V \mid^2)$
- Lista c/ heap: $\Theta((\mid V \mid + \mid E \mid) \, log \mid V \mid)$

# Algoritmo de Prim

| | | |
|---|---|---|
| a(−, −) | **b(a, 3)** c(−, ∞) d(−, ∞) e(a, 6) f(a, 5) |  |
| b(a, 3) | **c(b, 1)** d(−, ∞) e(a, 6) f(b, 4) |  |
| c(b, 1) | d(c, 6) e(a, 6) **f(b, 4)** |  |
| f(b, 4) | d(f, 5) **e(f, 2)** |  |
| e(f, 2) | **d(f, 5)** |  [2] |
| d(f, 5) | | |

[2] Fonte: A. Levitin. Introduction to the Design and Analysis of Algorithms. 2011.

# Algoritmo de Prim

**Algoritmo:** void Prim(Graph G, int[] D, int[] V)

```
H[0] ← (0, 0);
for i ← 0 to n(G) − 1 do
    D[i], V[i] ← ∞, −1;
    setMark(G, i, UNVISITED);
D[0] ← 0;
for i ← 0 to n(G) − 1 do
    repeat
        v ← vertex(removemin(H));
        if v = NULL then return;
    until getMark(G, v) = UNVISITED;
    setMark(G, v, VISITED);
    w ← first(G, v);
    while w < n(G) do
        if getMark(G, w) ≠ VISITED ∧ D[w] > weight(G, v, w) then
            D[w], V[w] ← weight(G, v, w), v;
            insert(H, (w, D[w]));
        w ← next(G, v, w);
```

# Agenda

# Algoritmo de Kruskal

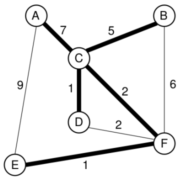
Outro exemplo de algoritmo guloso

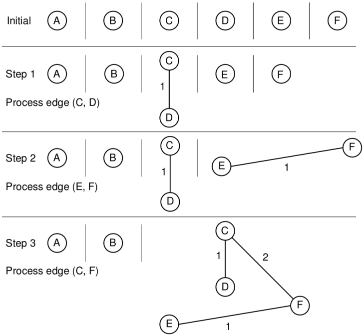
- Inicialmente, $n = |V|$ árvores mínimas
- Em cada passo, escolhe a menor aresta de $G$ que une duas árvores mínimas distintas (i.e., sem introduzir um ciclo)

Eficiência temporal: similar à de Prim

- Prim: melhor em grafos densos
- Kruskal: melhor em grafos esparsos

# Algoritmo de Kruskal[3]




[3] Fonte: C. Shaffer. Data Structures and Algorithm Analysis. 2013.

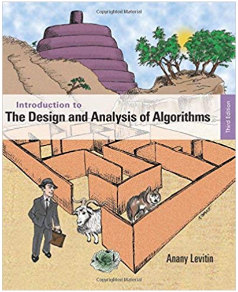
# Algoritmo de Kruskal

**Algoritmo:** void Kruskal(Graph G)

```
edgecnt ← 0;
for i ← 0 to n(G) − 1 do
    w ← first(G, v);
    while w < n(G) do
        E[edgecnt++] ← (weight(G, i, w), i, w);
        w ← next(G, i, w);
HeapBottomUp(E);                          // heap criada de forma bottom-up
ds ← create_disjointSubset(n(G));         // manipular conjuntos disjuntos
numMST ← n(G);
while numMST > 1 do
    temp ← removemin(H);
    v, u ← firstNode(temp), secondNode(temp);
    if find(ds, v) ≠ find(ds, u) then
        union(ds, v, u);
        numMST--;
```

# Agenda

# Bibliografia + leitura recomendada




**Capítulo 9 (pp. 318–322)**
**Capítulo 9 (pp. 325–327)**
**Anany Levitin.**
*Introduction to the Design and Analysis of Algorithms*.
3a edição. Pearson. 2011.


**Capítulo 11 (pp. 393–399)**
**Clifford Shaffer.**
*Data Structures and Algorithm Analysis*.
Dover, 2013.

# GRAFOS – ÁRVORES DE CUSTO MÍNIMO

Gustavo Carvalho
(ghpc@cin.ufpe.br)

Universidade Federal de Pernambuco
Centro de Informática, 50740-560, Brazil