Governo Federal

Ministério da Educação

deINF departamentodeInformática

Universidade Federal do Maranhão

A Universidade que Cresce com Inovação e Inclusão Social

Estrutura de Dados II

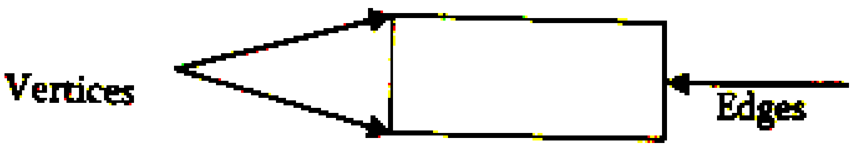
Árvore Geradora Mínima
Algoritmo de Prim e

Problema

- Projeto de redes de comunicações conectando n localidades.
- Arranjo de n – 1 conexões, conectando duas localidades cada.
- Conexões: cabos de transmissão
- Objetivo: dentre as possibilidades de conexões, achar a que usa menor quantidade de cabos.

Árvore Geradora Mínima (MST)

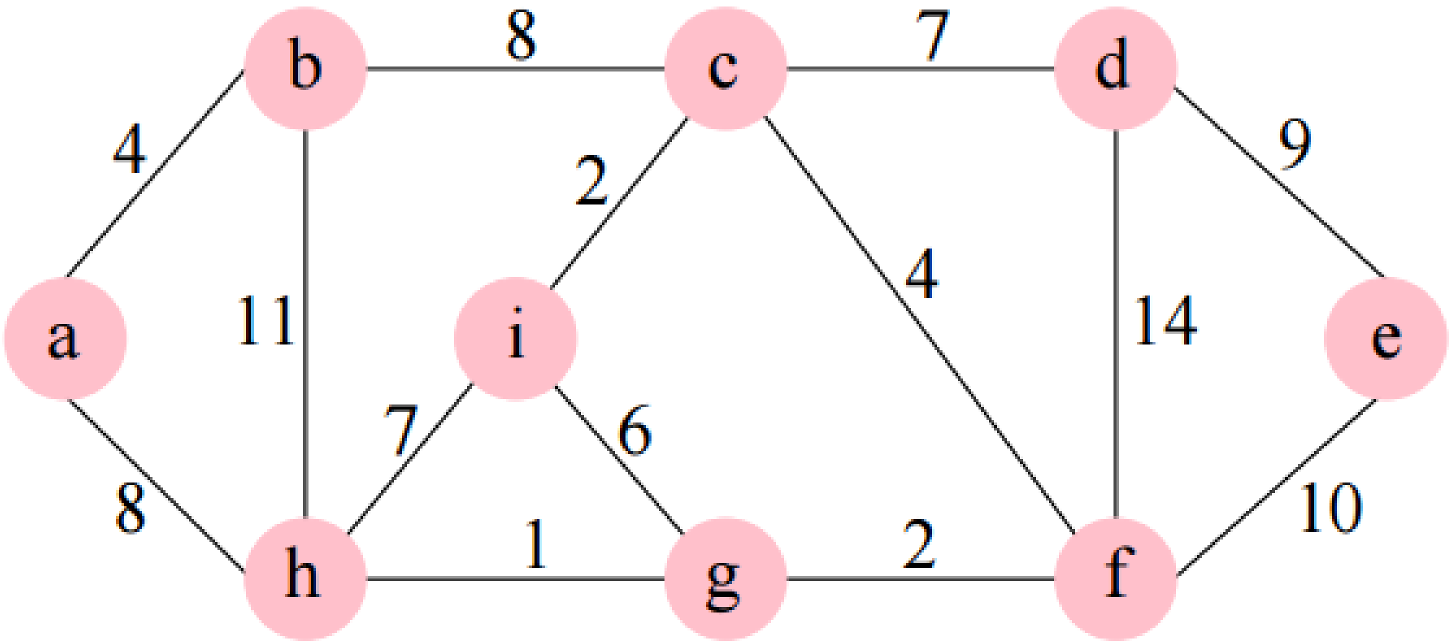
- **Árvore geradora de um grafo:**
  - É um subgrafo que contém todos os vértices e é também uma árvore
  - Um grafo pode ter muitas MST



Modelagem

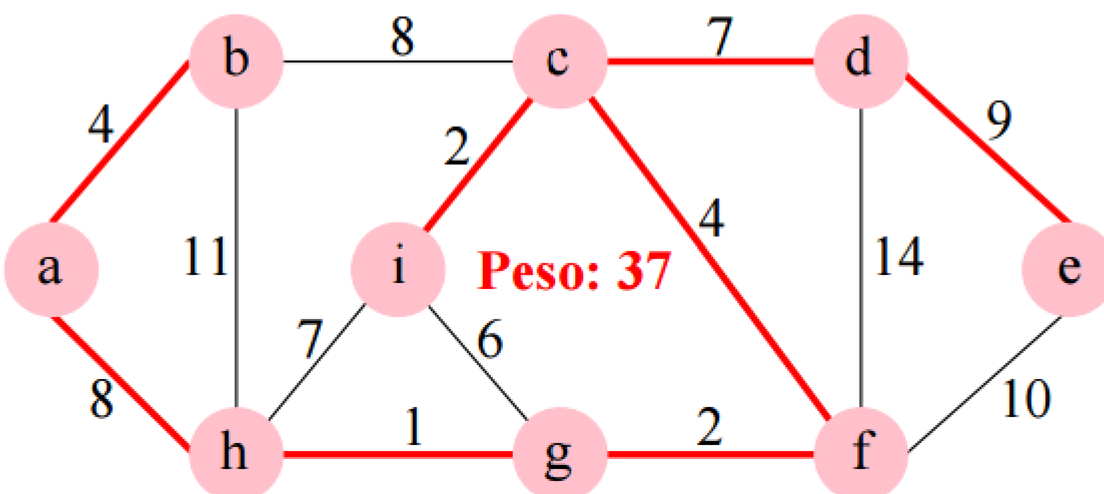
- Modelagem:
  - $G = (V,A)$: grafo conectado, não direcionado.
    - V : conjunto de cidades.
    - A: conjunto de possíveis conexões
    - $p(u, v)$: peso da aresta $(u, v) \in A$, custo total de cabo para conectar u a v.
- Solução: encontrar um subconjunto $T \subseteq A$ que conecta todos os vértices de G e cujo peso total $p(T) = P(u,v) \in T\ p(u, v)$ é minimizado.

Modelagem



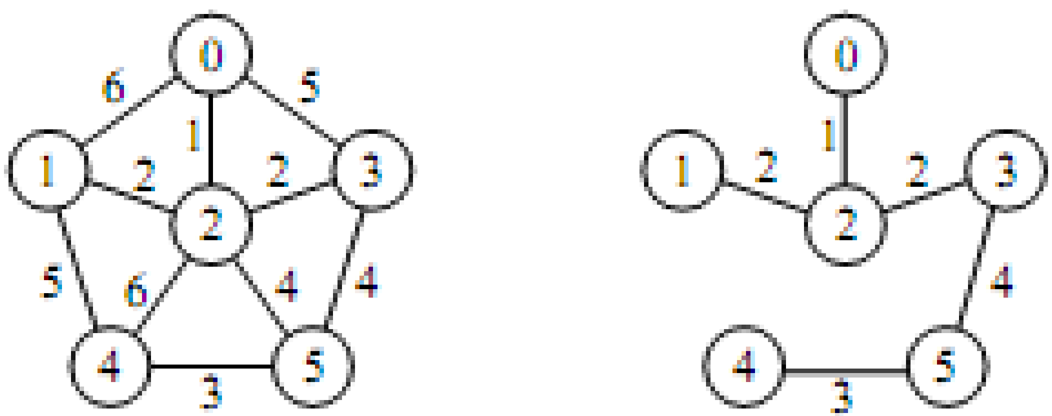
Modelagem

- Como G' = (V, T) é acíclico e conecta todos os vértices, T forma uma árvore chamada árvore geradora de G.
- O problema de obter a árvore T é conhecido como **árvore geradora mínima (AGM) (ou Árvore de Espalhamento Mínimo, Minimum Spanning Tree - MST)**



Exemplo

- Ex.: Árvore geradora mínima T cujo peso total é 12. T não é única, pode-se substituir a aresta (3, 5) pela aresta (2, 5) obtendo outra árvore geradora de custo 12.



(a) (b)

Algoritmo Genérico

- Uma estratégia gulosa permite obter a AGM adicionando uma aresta de cada vez.
- Invariante: Antes de cada iteração, S é um subconjunto de uma árvore geradora mínima.

Algoritmo Genérico

- A cada passo adicionamos a S uma aresta (u, v) que não viola o invariante. (u, v) é chamada de uma aresta segura.

void GenericoAGM

    $S = \emptyset$;

    while (S não constitui uma árvore geradora mínima)

        (u, v) = seleciona (A);

        if (aresta (u, v) é segura para S) $S = S + \{(u, v)\}$

    return S;

Algoritmo Genérico

- Dentro do while, S tem que ser um subconjunto próprio da AGM T, e assim tem que existir uma aresta (u, v) ∈ T tal que (u, v) ∈ S e (u, v) é seguro para S

```
GENERIC-MST(G, w)
A ← 0
while A não formar uma árvore espalhada
    do encontrar uma aresta (u, v) que seja segura para A
        A ← A ∪ {(u, v)}
return A
```

Como reconhecer arestas seguras?

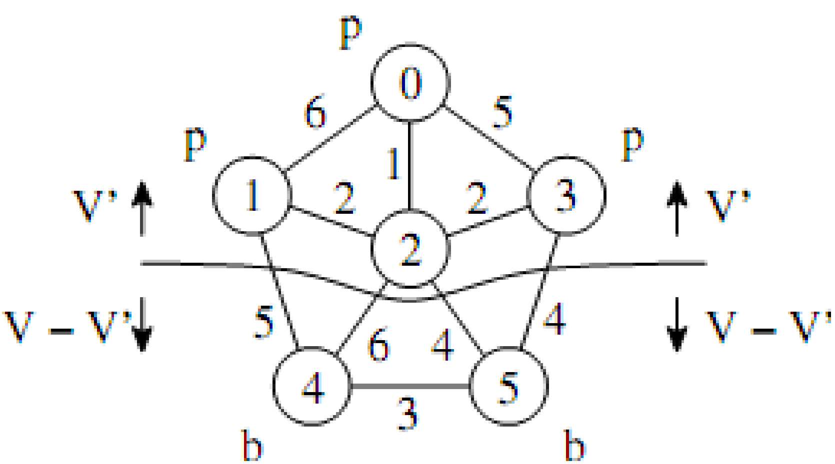
- **Definição de Corte.**
- **Aresta Leve.**

Definição do Corte

- Um corte (V’, V – V’) de um grafo não direcionado G = (V,A) é uma partição de V.
- Uma aresta (u, v) ∈ A cruza o corte (V’, V – V’) se um de seus vértices pertence a V’ e o outro vértice pertence a V – V’
- Um corte respeita um conjunto S de arestas se não existirem arestas em S que o cruzem.

Aresta Leve

- Uma aresta cruzando o corte que tenha custo mínimo sobre todas as arestas cruzando o corte é uma aresta leve



Aresta Segura

- Seja G = (V,A) um grafo conectado, não direcionado, com pesos p sobre as arestas V

- seja S um subconjunto de V que está incluído em alguma AGM para G.

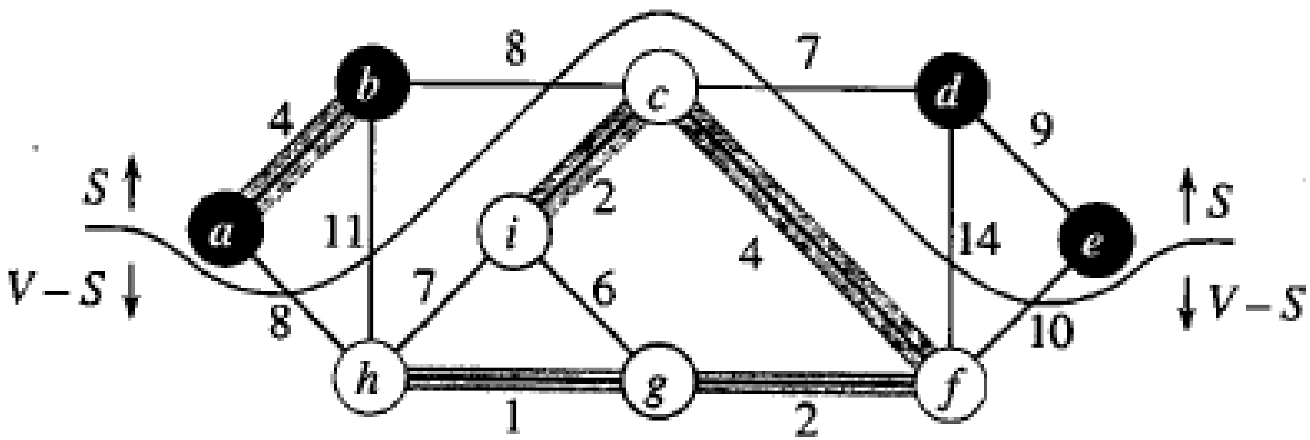
- Seja (V’, V – V’) um corte qualquer que respeita S.

- Seja (u, v) uma aresta leve cruzando (V’, V – V’).

- Satisfeitas essas condições, a aresta (u, v) é uma aresta segura para S.

Exemplo corte (S, V-S)

- Vértices do conjuno S em preto
- Vértices do conjunto S-V em branco
- As arestas que cruzam o corte são as que conectam os vértices brancos com vértices pretos
- A aresta (d,c) é a única aresta leve que cruza o corte
- O subconjunto A de aresta está sombreado; observe que o corte (S,V-S) respeita A, pois nenuma aresta de A cruza o Corte



Árvore Geradora Mínima

- **Principais algoritmos para resolver o problema (ambos utilizam estratégias gulosas): os algoritmos de Kruskal e Prim**
- **Os algoritmos diferem basicamente em como aplicar as técnicas**
  - O algoritmo de Kruskal executa em tempo $O(|E| \lg |E|)$ usando algoritmo de ordenação merge-sort e a estrutura de dados union-find
    - Union-Find(uma estrutura de dados que considera um conjunto de elementos particionados em vários subconjuntos disjuntos)
  - O algoritmo de Prim executa em tempo $O(|E| + |V| \lg |V|)$ se for utilizado um heap Fibonacci

Árvore Geradora Mínima

- **Prim x Kruskal**
  - Cada um utiliza regra especifica para determinar uma aresta segura
  - Kruskal:
    - o conjunto A é uma floresta
    - A aresta segura adicionada a A é sempre uma aresta de peso mínimo no grafo que conecta dois componentes distintos
  - Prim:
    - O cojunto A forma uma única árvore
    - A aresta segura adicionada a A é sempre uma aresta de peso mínimo que conecta a árvore a um vértice não presente na árvore

Árvore Geradora Mínima

- **O que significa dizer que os algoritmos são “gulosos”?**
- **A cada passo, uma de muitas possíveis opções deve ser escolhida**
  - A estratégia faz a escolha que dá o maior ganho imediato Essa estratégia não garante, em geral, encontrar a solução ótima
  - No entanto, para o problema da MST, a estratégia gulosa produz a solução ótima!

Algoritmo Guloso Genérico para cálculo da MST

- **Grande questão relativa ao algoritmo genérico: como encontrar uma aresta segura?**
- **Observações:**
  - À medida que o algoritmo progride, o conjunto A é sempre acíclico
  - Em qualquer estágio do algoritmo, o grafo $G_A = (V, A)$ é uma floresta e cada componente conexo de G_A é uma árvore
    - No início do algoritmo, cada árvore contém apenas um vértice
  - Qualquer aresta segura (u, v) para A conecta apenas componentes distintos de G_A, uma vez que $A \cup \{(u, v)\}$ deve ser acíclico

Algoritmo de Prim

- **Assim como o algoritmo de Kruskal, Prim é um caso especial do algoritmo genérico de MST**
  – Este algoritmo opera de forma similar ao algoritmo de Dijkstra para encontrar os menores caminhos em um grafo
- **Este algoritmo tem a propriedade de que as arestas no conjunto A sempre formam uma única árvore**
  – Esta árvore inicia-se (raiz) a partir de um vértice arbitrário e cresce até que a árvore estenda-se a todos os vértices em V

Algoritmo de Prim

- **Durante a execução, os vértices que não estão em GA residem em uma fila de prioridade Q**
  - Para cada vértice v, key[v] é o peso da aresta mais leve (menor peso) conectando v a algum vértice de V[GA]
  - key[v] = [] se tal aresta não existe
  - O parâmetro [][v] corresponde ao pai de v na MST

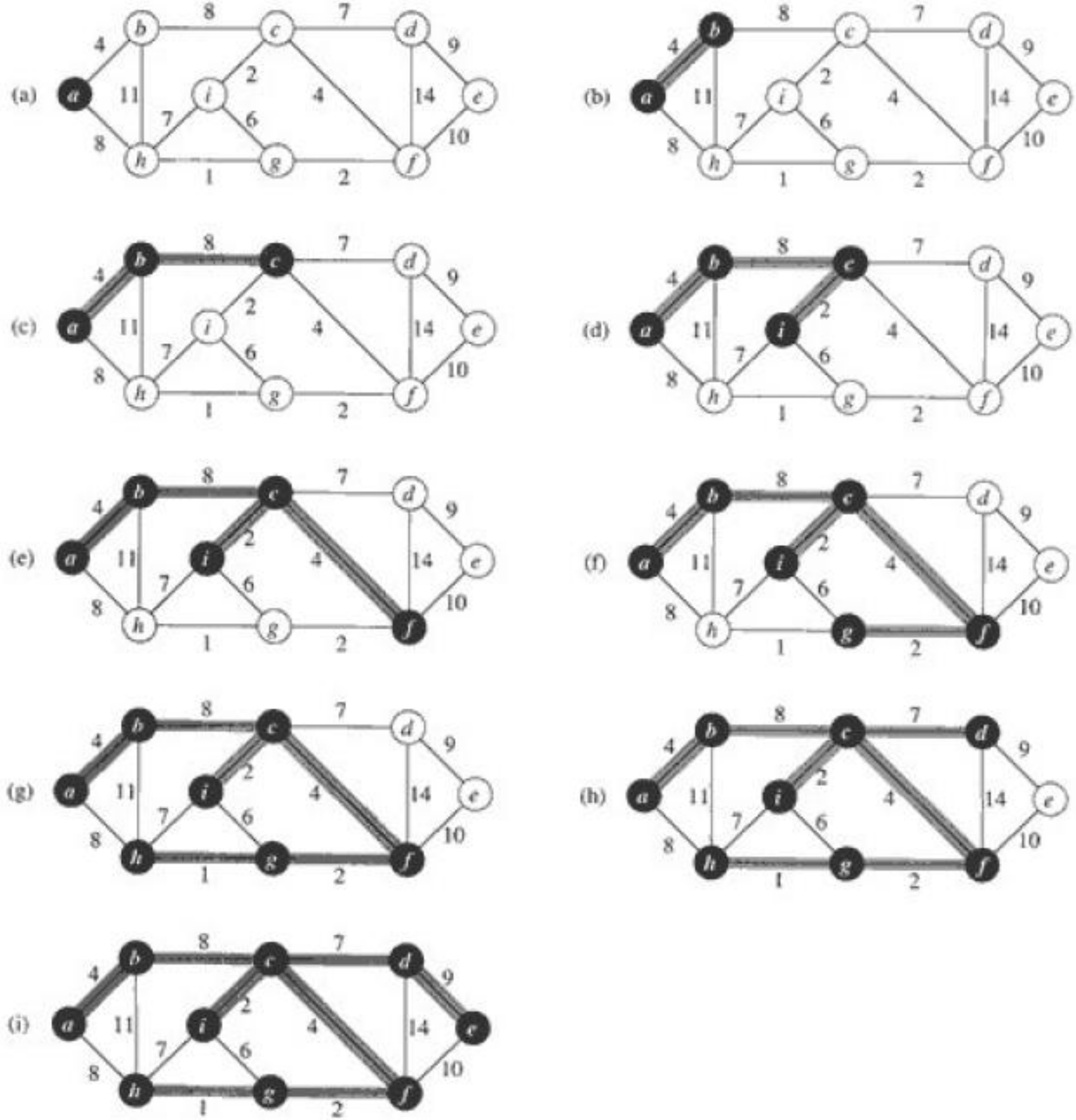
- **Quando o algoritmo termina, a fila de prioridades Q está vazia, e a MST A para G é dada por**
- **A = {(v, [][v]) : v ∈ V – {r}}**

Algoritmo de Prim

```
MST-PRIM(G, w, r)
for each u ∈ V[G]
    do key[u] ← ∞
       π[u] ← NIL
key[r] ← 0
Q ← V[G]
while Q ≠ Ø
    do u ← EXTRACT-MIN(Q)
       for each v ∈ Adj[u]
           do if v ∈ Q and w(u, v) < key[v]
                 then π[v] ← u
                      key[v] ← w(u, v)
```

```
Prim(G,r):
  para cada v ∈ V(G) faça
      p[v] = ∞
      pai[v] = −1
  p[r] = 0
  Constrói heap mínimo A com V(G) (com base em p)
  S = ∅
  enquanto |A| > 1 faça
      u = RetiraMin(A); Refaz heap
      S = S ∪ {u}
      para v ∈ adj(u) faça
          se (v ∈ A) e (p[v] > p(u, v)) então
              p[v] = p(u, v)
              pai[v] = u; Refaz heap
```

Exemplo



Algoritmo de Prim - Análise

- O desempenho do algoritmo de Prim depende de como implementamos a fila de prioridades Q
- Se for utilizado um heap binário, então
- Os passos de inicialização de 1 a 5 são executados em tempo O(|V|)
- O laço while é executado |V| vezes
  - Uma vez que EXTRACT MIN custa O(lg |V|) as chamadas EXTRACT MIN vão custar O(|V| lg |V|)
- O laço for, 8-11, é executado |E| vezes, uma vez que o comprimento das listas de adjacências é no máximo 2|E|
  - A atribuição na linha 11 envolve operações no heap da ordem O(lg|V|)
  - Assim, o custo total de redução de chaves é O(|E| lg |V|)

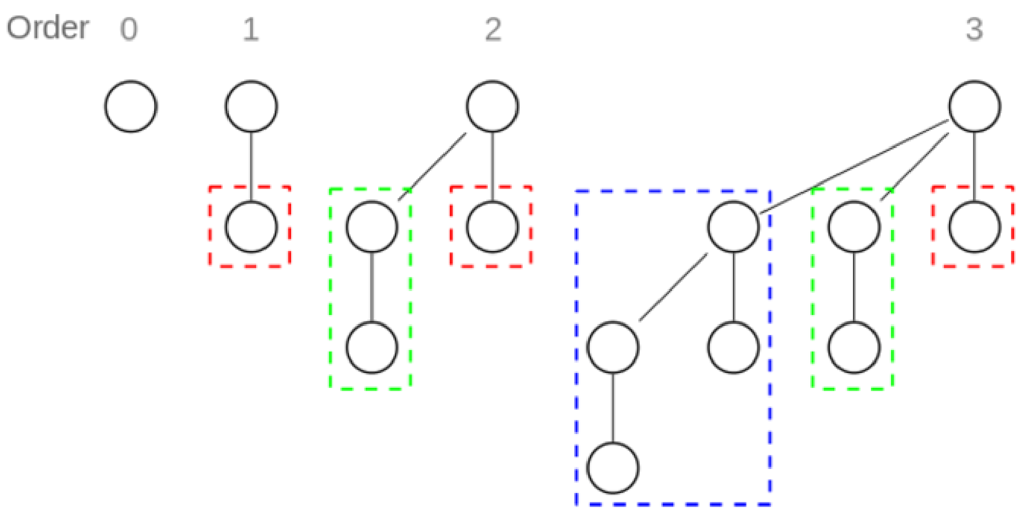
Algoritmo de Prim - Análise

- Portanto, o algoritmo de Prim tem tempo de execução
  $O(|V| \lg |V| + |E| \lg |V|) \in O(|E| \lg |V|)$

- Usando heaps Fibonacci, o algoritmo de Prim pode ter complexidade reduzida para
  $O(|V| \lg |V| + |E|)$

Heap de Fibonacci

- **Coleção de árvores ordenados como heaps mínimos. Expande os Termos:**
  – NC(0) = 1
  – NC(1) = 2
  – NC(2) = NC(0) + NC(0) + 1 =
  – 3 NC(3) = NC(1) + NC(0) + NC(0) + 1 = 5 NC(4) = NC(2) + NC(1) + NC(0) + NC(0)



Referencias

- CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. Algoritmos: Teoria e Prática. Editora Campus, 2002
- Ziviani, N. Projeto de Algoritmos Com Implementações em Pascal e C, Cengage Learning, 2004.
- Notas de aula. Prof. Rafael Fernandes DAI/IFMA
- Notas de aula. Profa. Leticia Bueno UFABC
- **http://www.facom.ufu.br/~madriana/EBD/Didatica.pdf**