① 14ª Aula

Árvore geradora mínima

Situação hipotética:

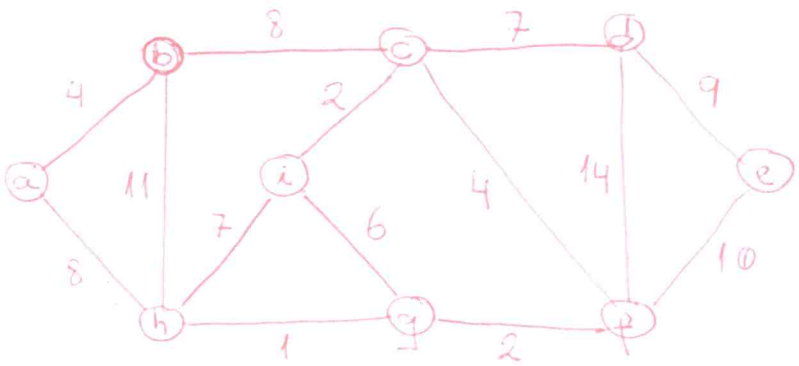
Conjunto de computadores, onde cada par de computadores pode ser ligado usando uma quantidade de cabo/fibra. Qual será a rede interconectada que utiliza a menor quantidade de cabo/fibra?

② Formalização do problema

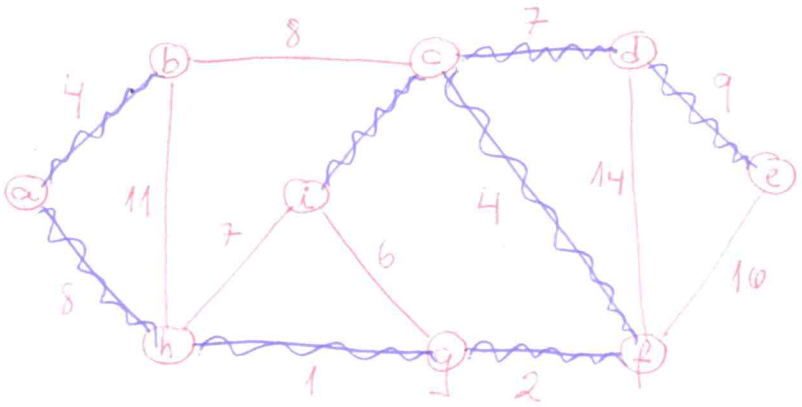
Entrada: Grafo conexo $G = (V, E)$ com pesos $w(u,v)$ para cada aresta (u,v)

Saída: Subgrafo gerador conexo T de G cujo peso total $w(T) = \sum_{(u,v) \in T} w(u,v)$ seja o menor possível

③ Exemplo



G: Grafo



T: Árvore geradora mínima

④ Algoritmo de Kruskal

```
MST-Kruskal(G, w){
  T ← ∅;
O(m)→ Construa uma fila de prioridade Q com as arestas
      de E usando We como chave;
O(n)→ VS ← {{v} : v ∈ V} /* inicialização do union-find */
      [enquanto |VS| > 1 faça]
O(log m)→ [remova e = xy de custo mínimo de Q /* extrai-mínimo(Q) */
          se x e y estão em conjuntos distintos X e Y de VS /* find(x) ≠ find(y) */
            [então]
              [troque X e Y por X ∪ Y em VS; /* union(x,y) */
               adicione e = xy a T;
              ]
          ]
}
```

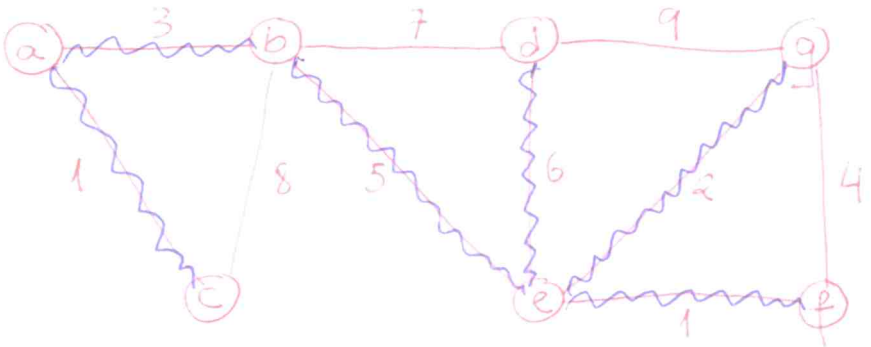
⑤ Análise do algoritmo

→ Complexidade: $O(m \cdot \log n) = O(|E| \log |V|)$

→ Note que o algoritmo utiliza uma estratégia gulosa

→ Maiores detalhes de union-find no Capítulo 21 (Estrutuas de dados para conjuntos disjuntos)

⑥ Exemplo



$w(T) = 1 + 3 + 5 + 6 + 2 + 1 = 18$

⑦ Algoritmo de Prim

→ Também é um algoritmo guloso

→ Complexidade: $O(m \log n)$

→ Aplique o algoritmo de Prim no mesmo exemplo do algoritmo de Kruskal (Exercício)

⑧ Problema do(s) caminho(s) mínimo(s)

→ $G = (V, E)$ é um grafo orientado com pesos (custo/distância nas arestas ($w(u,v)$).

→ Problema do caminho mínimo entre dois vértices: dados vértices s e t em (G, w), encontrar um caminho mínimo de s a t.

→ Problema dos caminhos mínimos de mesma origem: dados (G, w) e $s \in V$, encontrar para cada v em G, um caminho mínimo de s a v.

(9) Algoritmo de Dijkstra

Recebe um grafo orientado (G, w) sem arestas de peso negativo e um vértice s de G, e devolve:

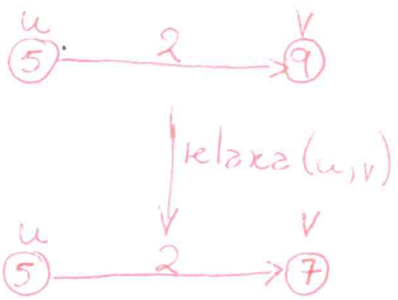
- para cada $v \in V[G]$, o peso de um caminho mínimo de s a v
- uma árvore de caminhos mínimos com raiz s (um caminho de s a v nesta árvore é um caminho mínimo de s a v em (G, w)).

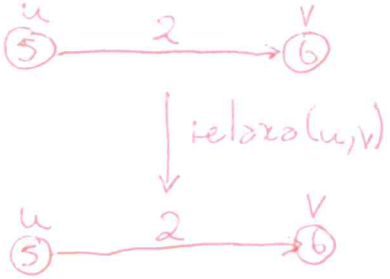
(10)

```
Dijkstra(G, s){
    para u em V[G] faça
        d(u) ← ∞
        π(u) ← null
    d(s) ← 0
    π(s) ← null
    Q ← heap(V[G]) /* fila de prioridade */
    S ← ∅
    enquanto Q ≠ ∅ faça
        u ← extrai-minimo(Q)
        S ← S ∪ {u}
        para cada vértice v ∈ Adj(u) faça
            relaxa(u, v)
            atualiza-heap(Q)
}
```

(11)

```
relaxa(u,v){
    se d(v) > d(u) + w(u,v) então
        d(v) ← d(u) + w(u,v)
        π(v) ← u
}
```

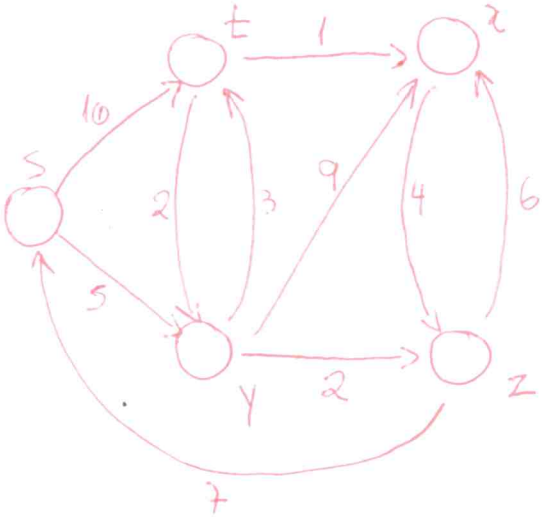


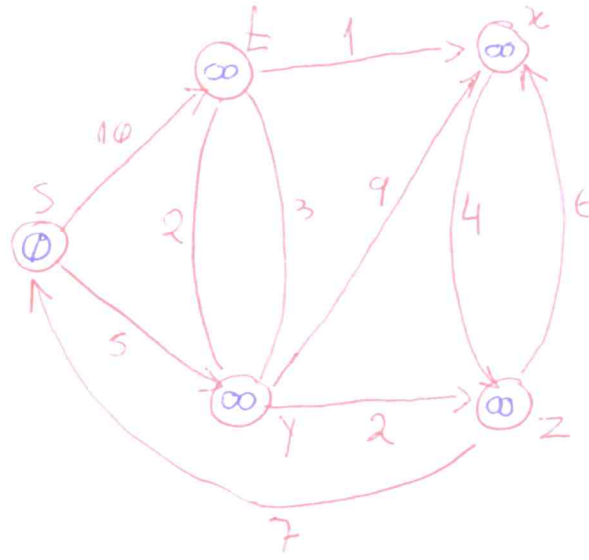


(12) Análise de complexidade

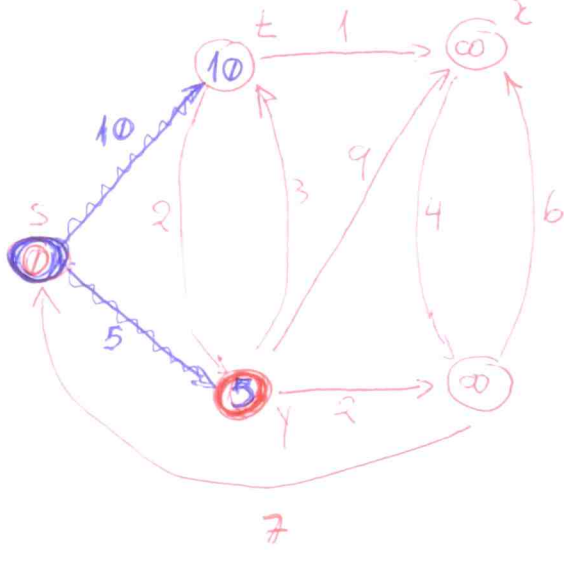
- Construção do heap: $O(n)$
- extrai-minimo() tem complexidade $\log n$ e é executado n vezes: $O(n \log n)$
- atualiza-heap() tem complexidade $\log n$ e é executado m vezes: $O(m \log n)$
- Complexidade total: $O((n+m)\log n)$

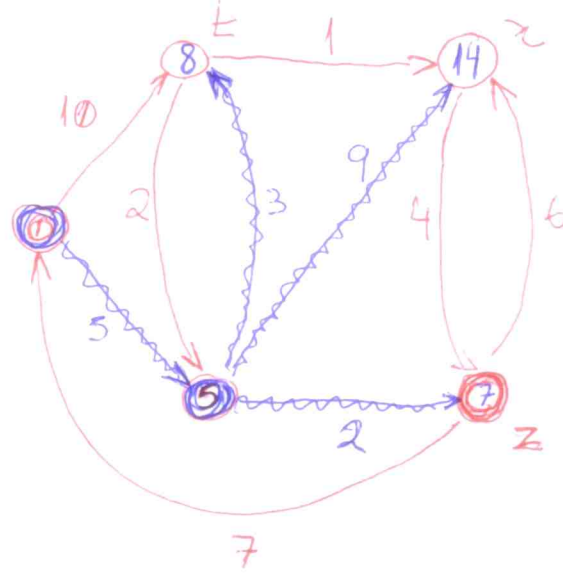
(13) Exemplo



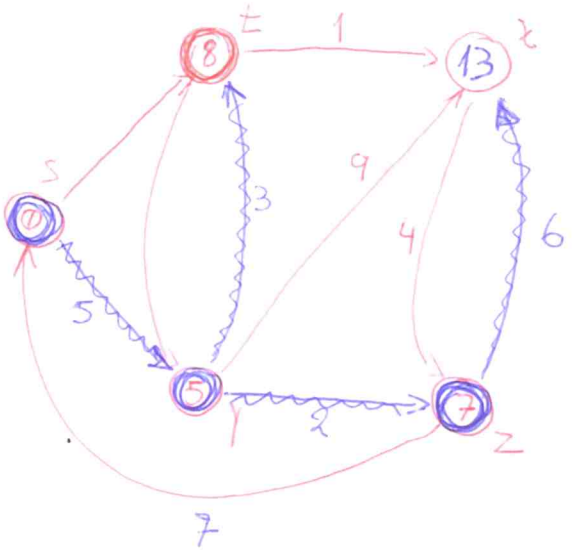


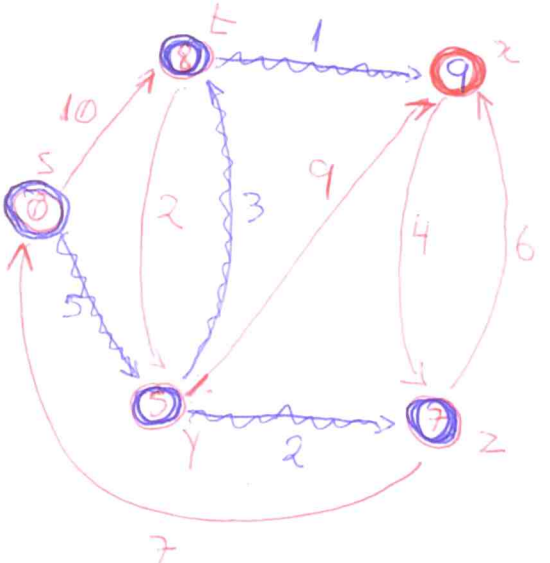
(14)



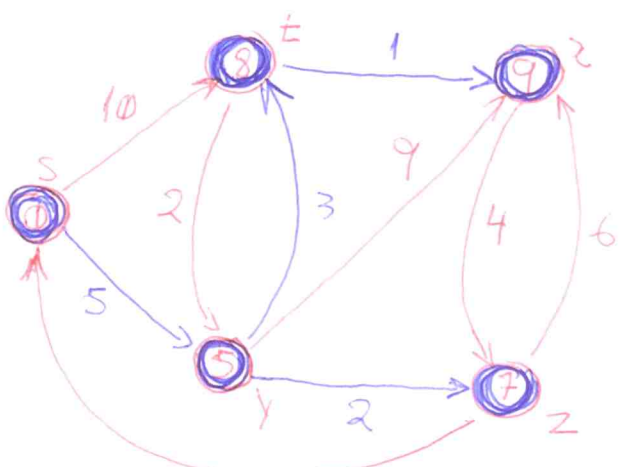


15





16



Limitação: Funciona apenas para arestas com pesos positivos

(17) # Algoritmo Bellman-Ford

→ Aceita arestas com pesos negativos

→ Complexidade: $O(n.m)$

→ Aplique o algoritmo de Bellman-Ford no mesmo exemplo do algoritmo de Dijkstra (Exercício)