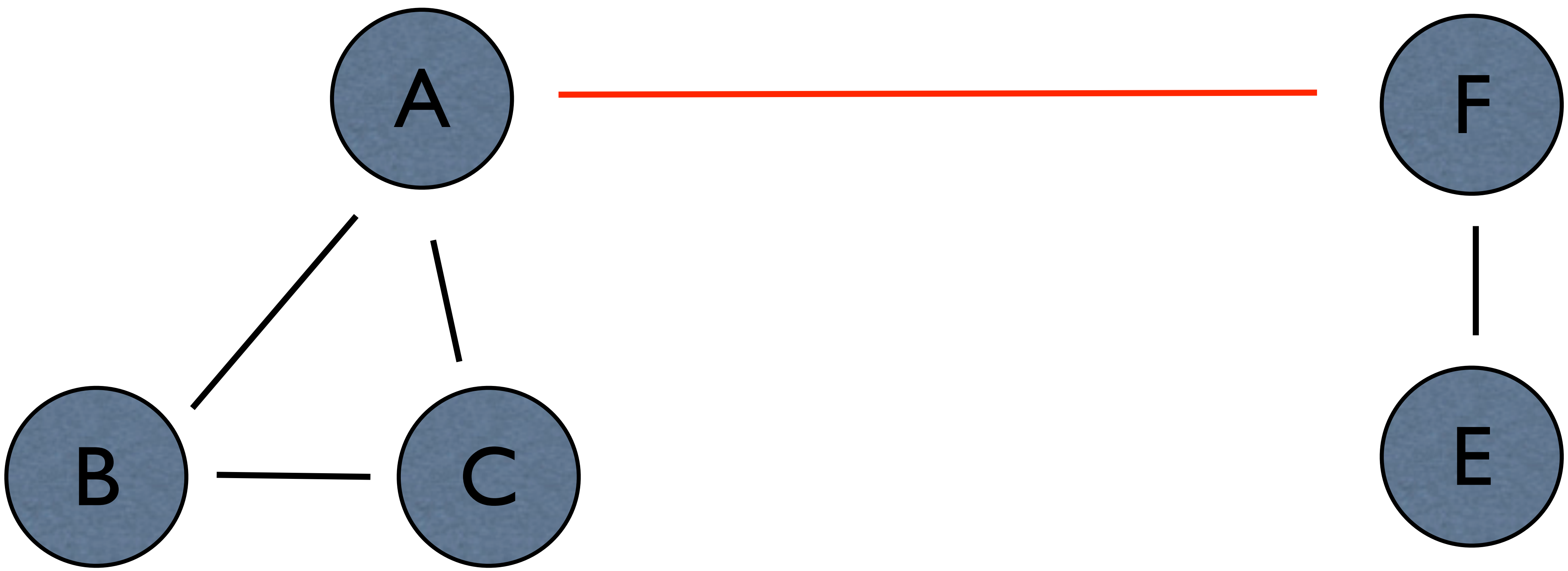
Detecção de Pontes

Prof. Leandro G. M. Alvim

Agenda

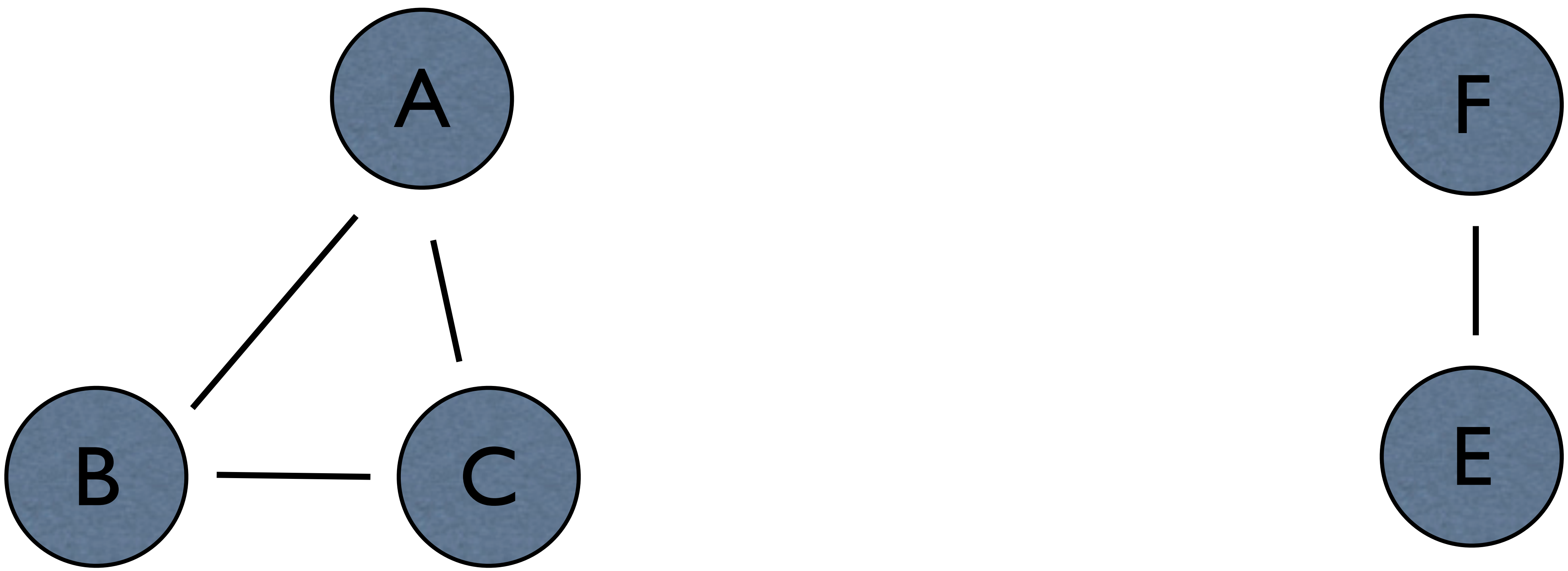
- Ponte
- Aplicação
- Algoritmo Simples
- Algoritmo de Tarjan

Ponte



Componentes
Conexas = 1

Ponte



Componentes
Conexas = 2

Definição

- Uma ponte ou aresta de corte é uma aresta (u,v) e E tal que quando removida do grafo, aumenta o número de componentes conexas

Banda Larga

- Desejamos identificar cabos críticos em uma rede de telefonia
  - Cabos que se falharem, interrompem a comunicação entre grande parte dos clientes. Ex. Conexão entre bairros
- Adicionar cabos extras
- Fazer manutenção

Algoritmo Simples

- c=componentesConexas(G)
- Para cada aresta (u,v) e E:
  - G'= G, remova (u,v) de G'
  - k=componentesConexas(G')
  - Se k != c:
    - "ponte: " , u,v

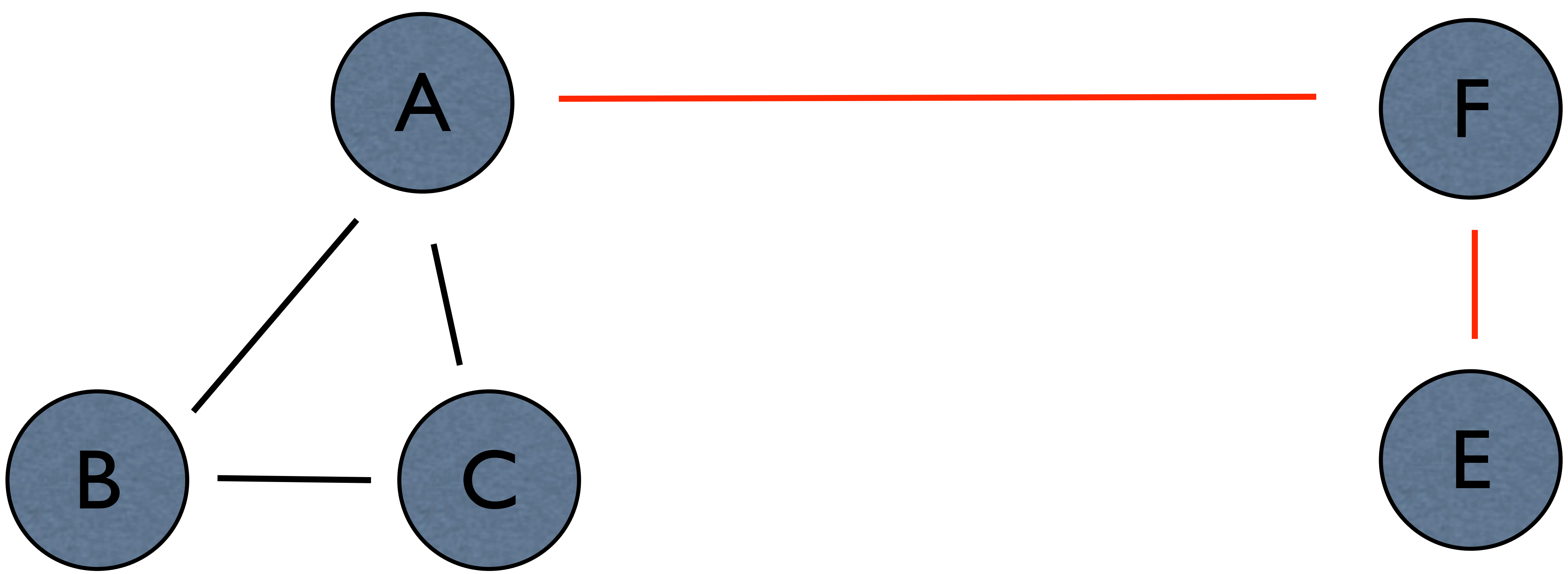
Algoritmo Simples

- Complexidade
  - $O(E*(V+E)) = O(EV + E^2)$

Quadrático em E!!

Algoritmo de Tarjan

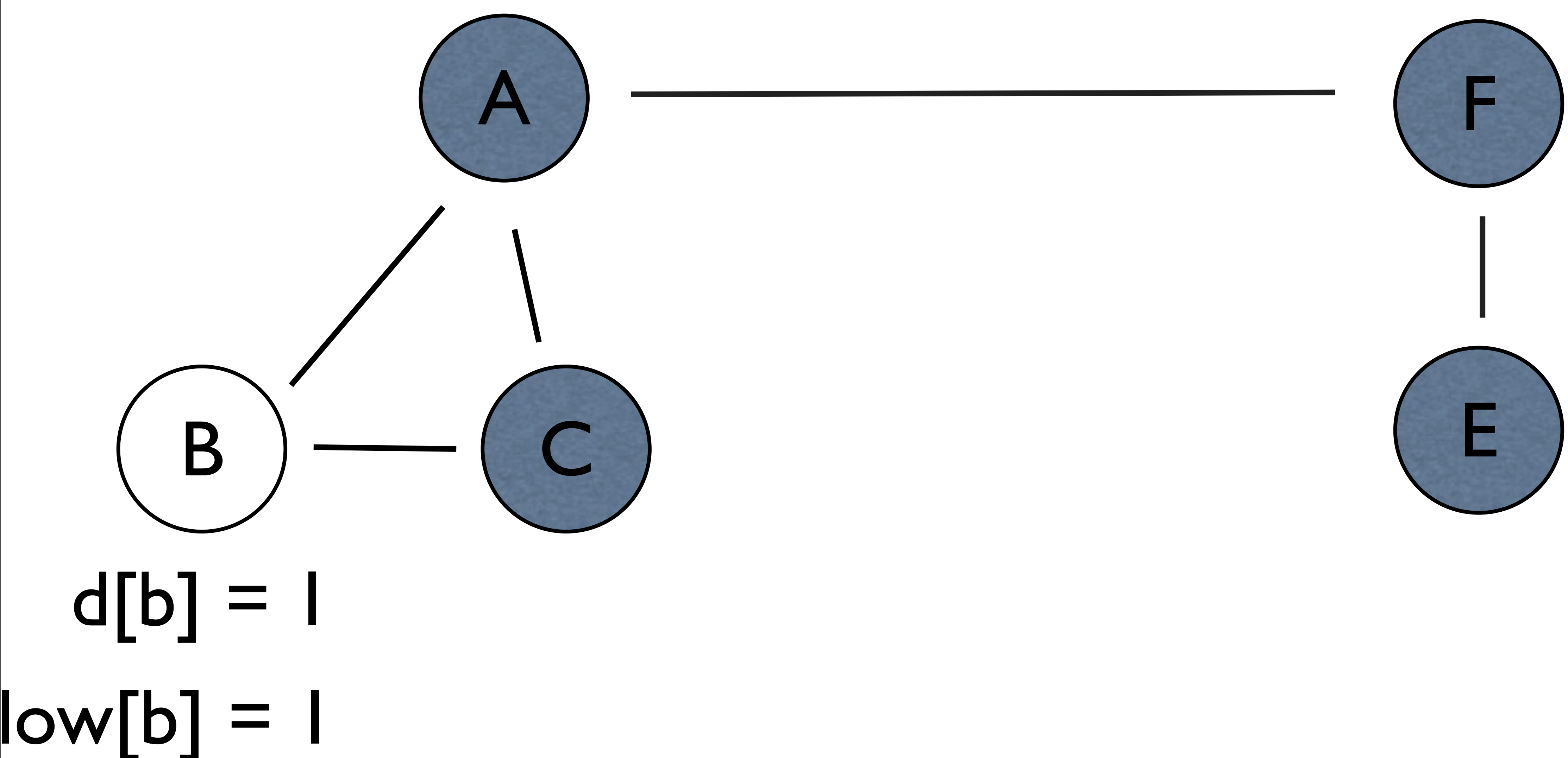
- Arestas que fazem parte de uma árvore são as pontes ou arestas que fazem parte de um ciclo não são pontes



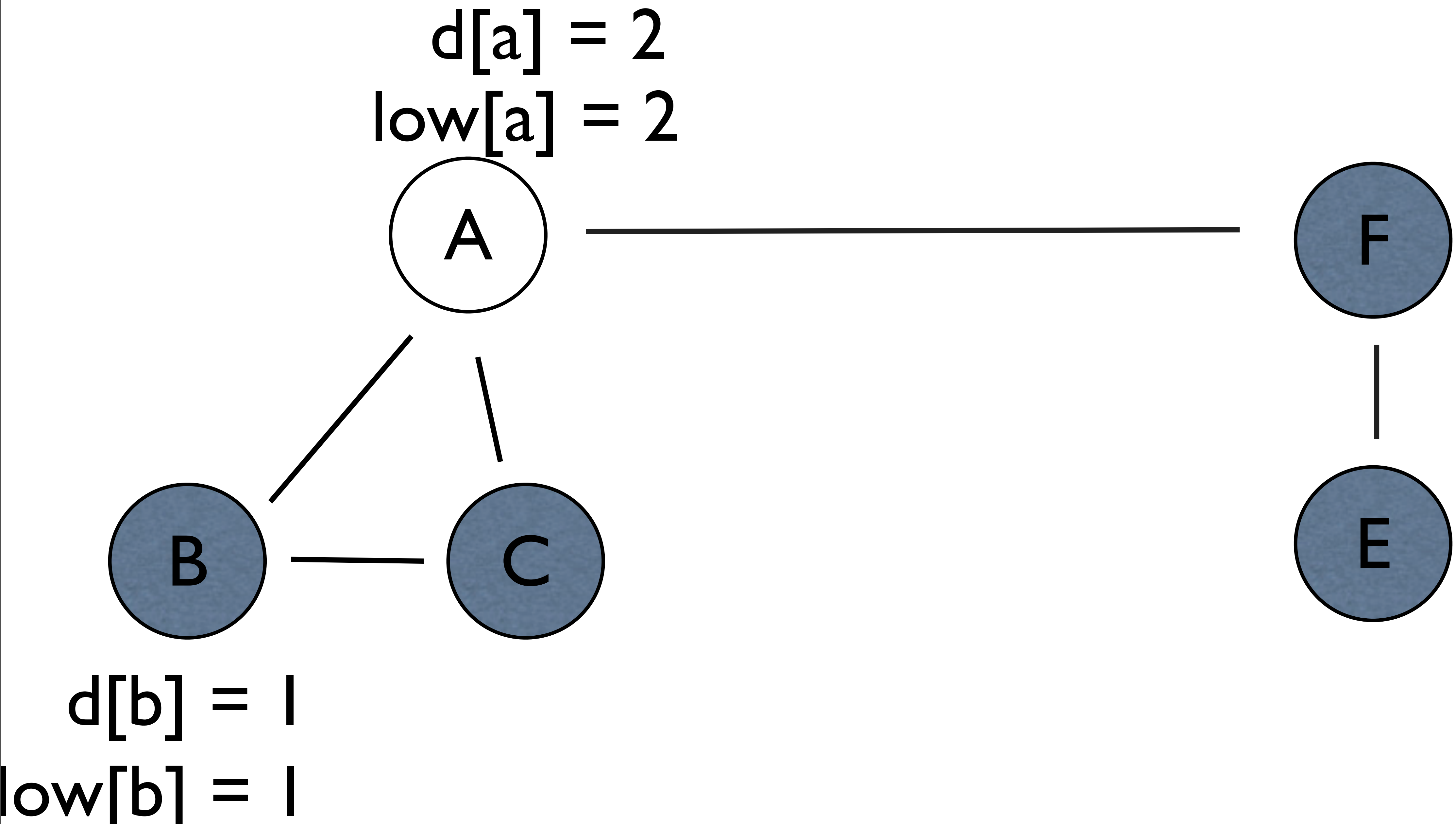
Algoritmo de Tarjan

- Busca em profundidade
  - Toda aresta dentro de um ciclo não será uma ponte
    - Tempo de visitação (d[u])
    - Menor tempo de visitação encontrado para um “filho” (low[v])

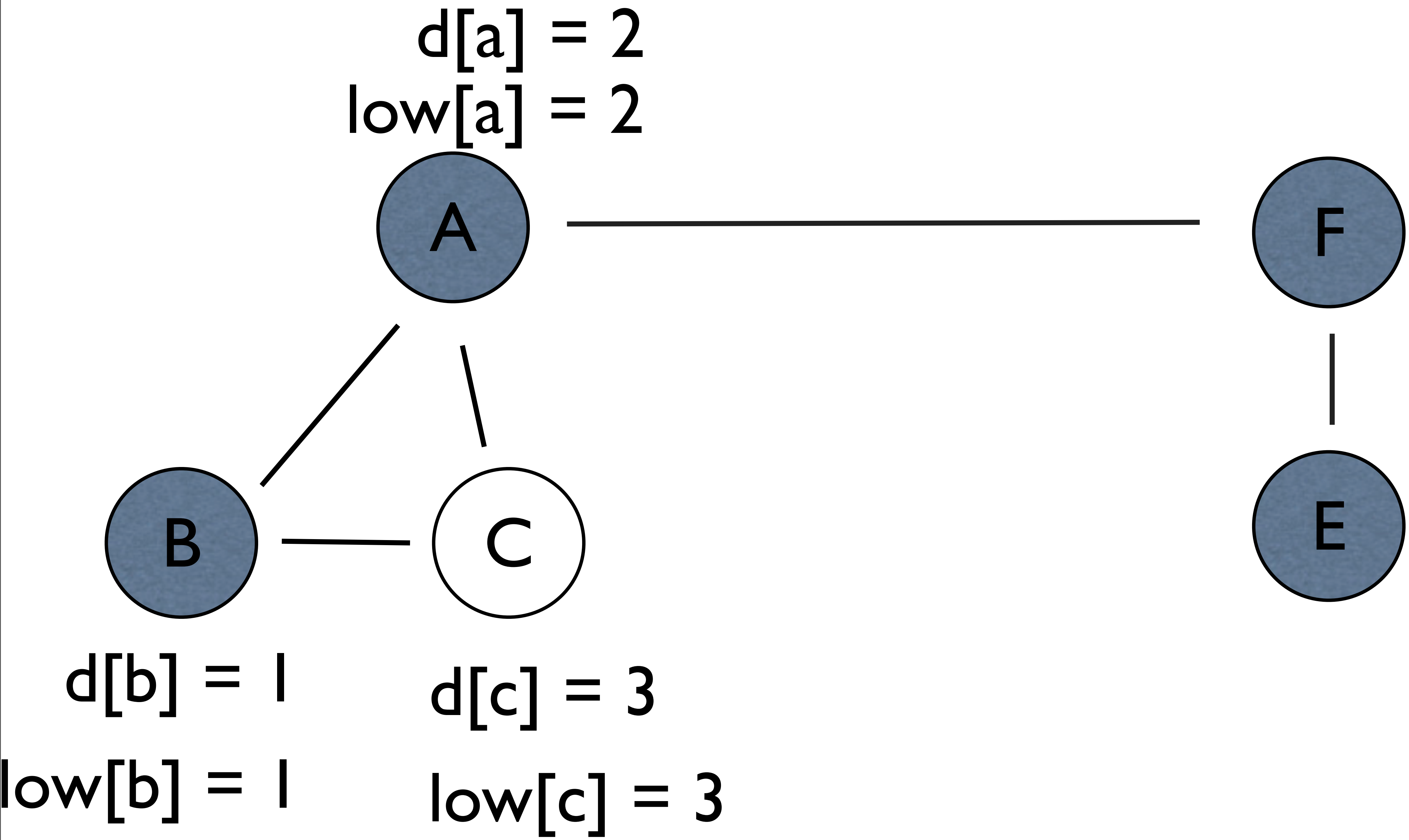
Algoritmo de Tarjan



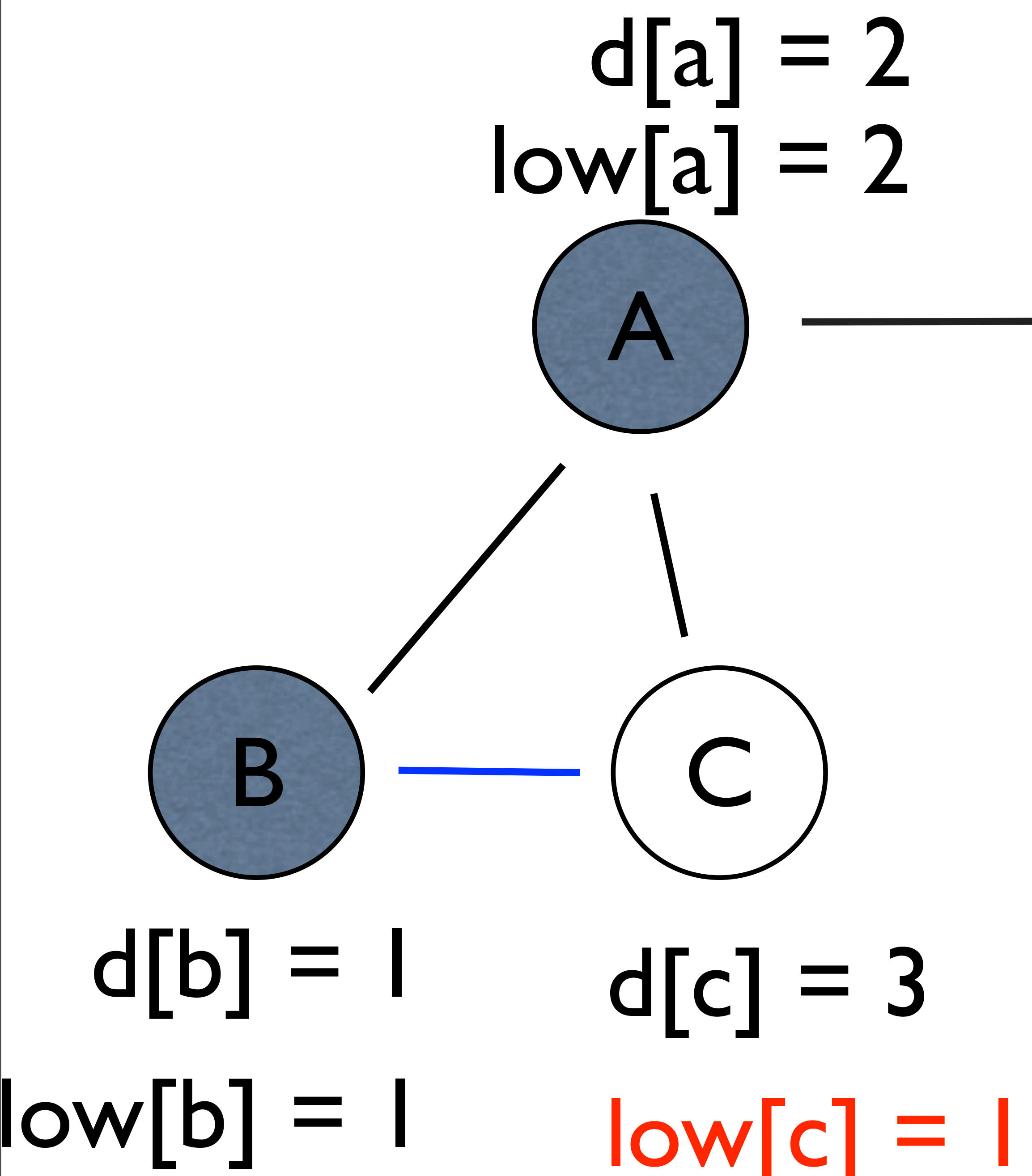
Algoritmo de Tarjan



Algoritmo de Tarjan



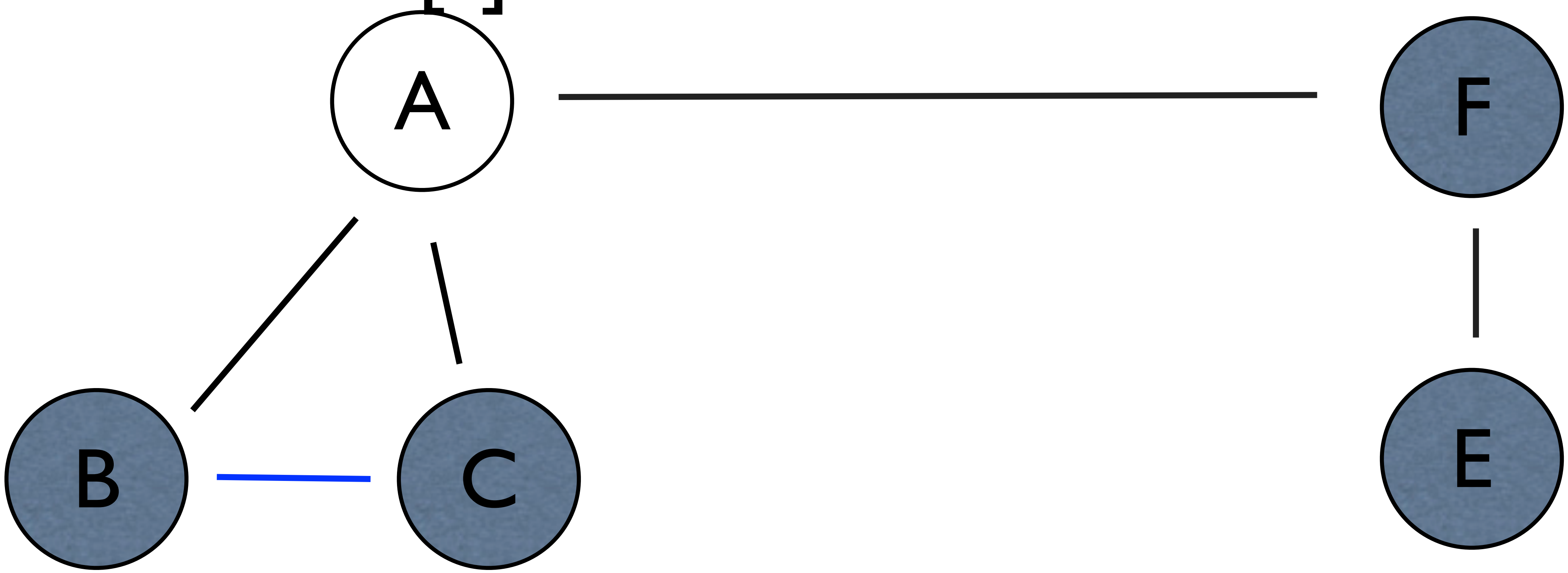
Algoritmo de Tarjan



(b,c) não é ponte pois encontra um nó na pilha (cinza)

Algoritmo de Tarjan

d[a] = 2
low[a] = 2

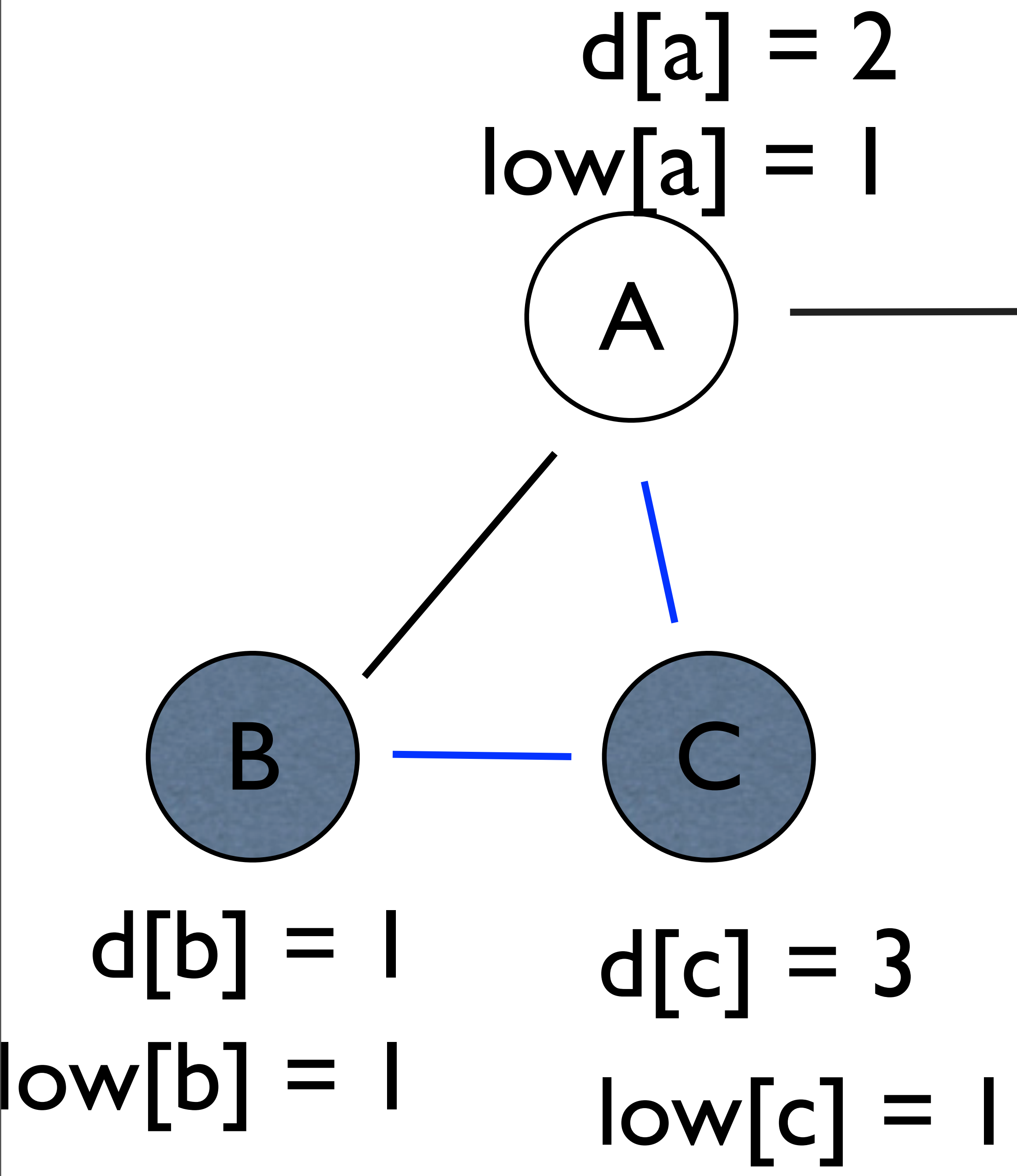


d[b] = 1
low[b] = 1

d[c] = 3
low[c] = 1

Voltamos
para (a,c)

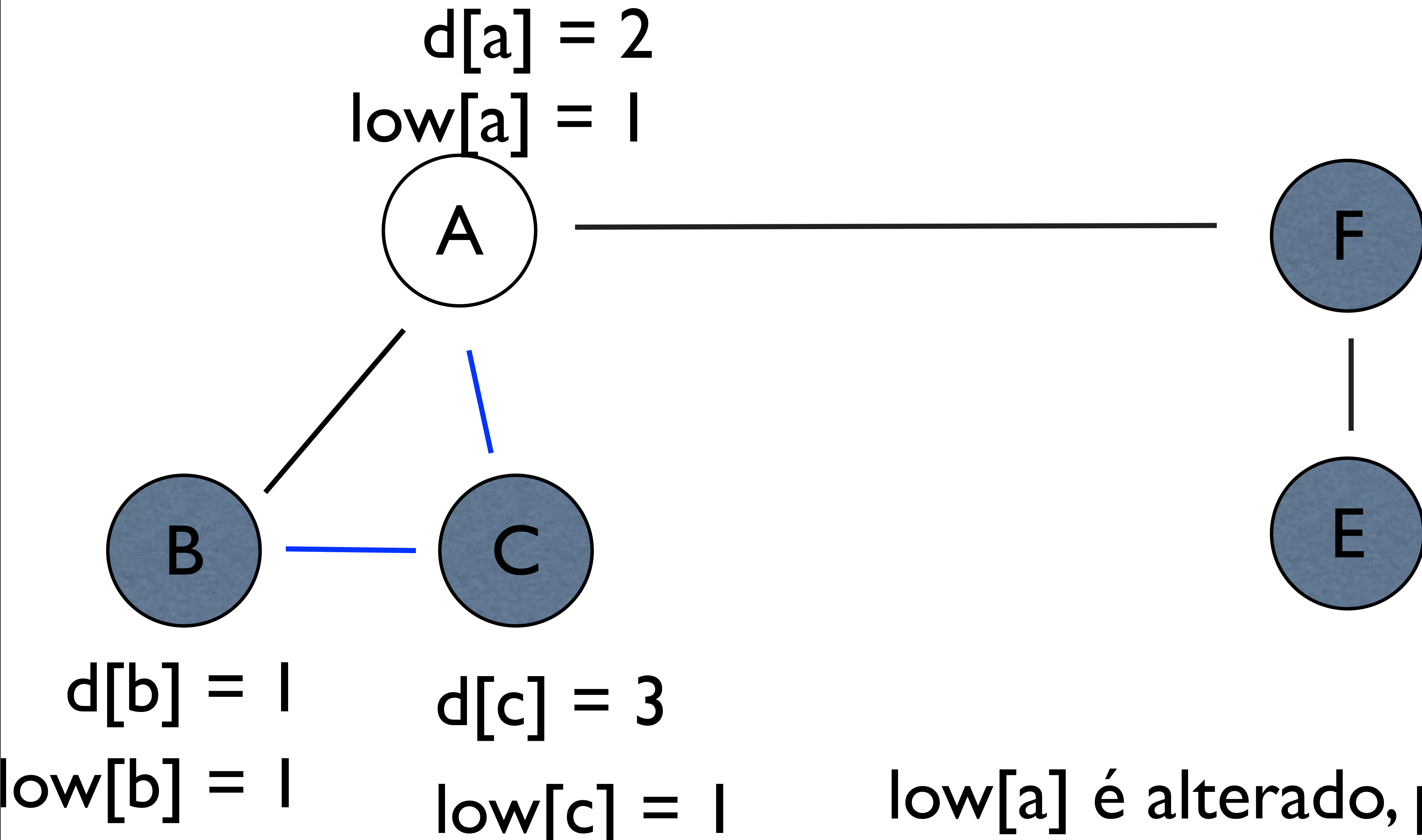
Algoritmo de Tarjan



Em (a,c), d[a] > low[c]

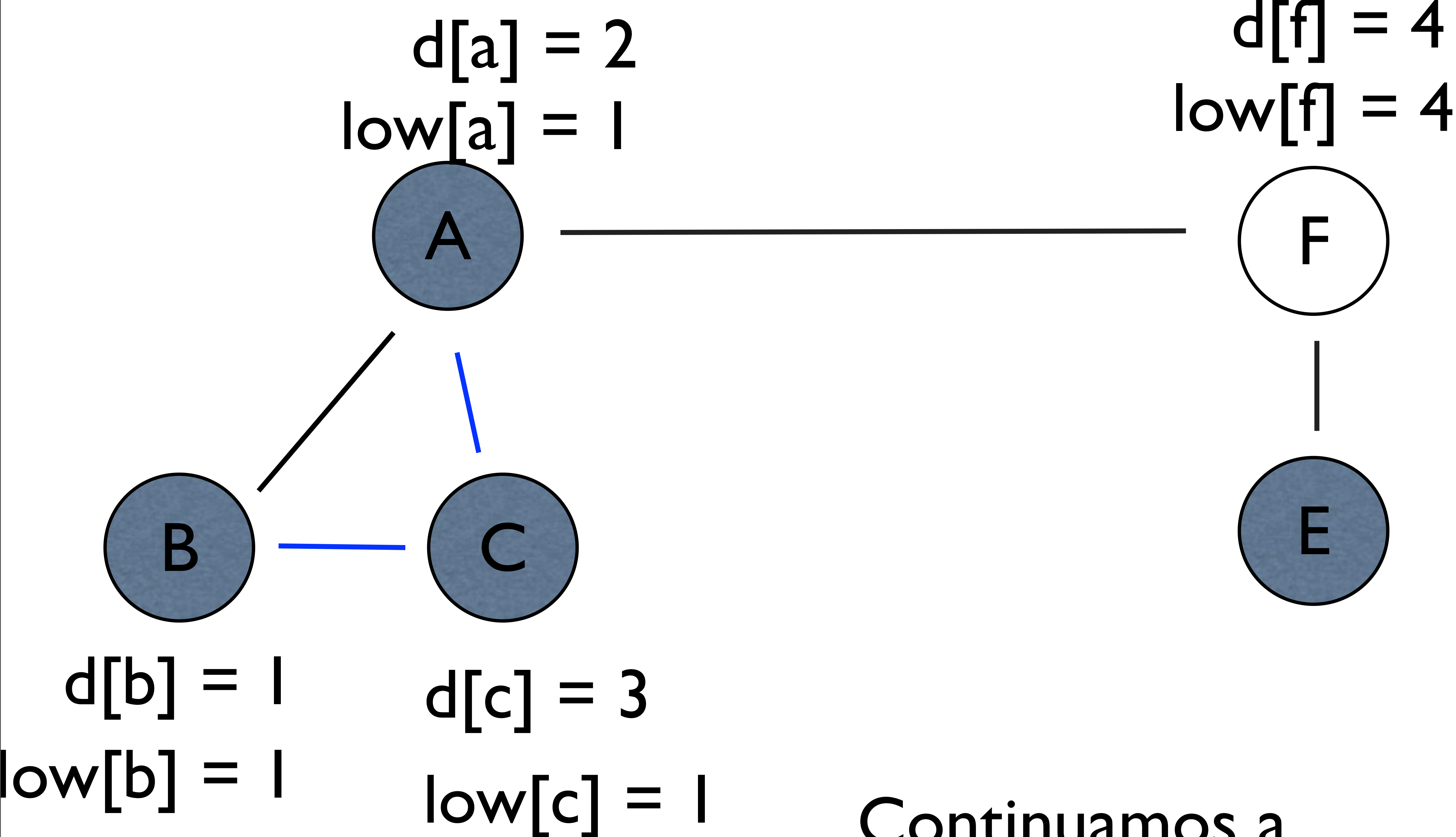
Logo, não é ponte

Algoritmo de Tarjan



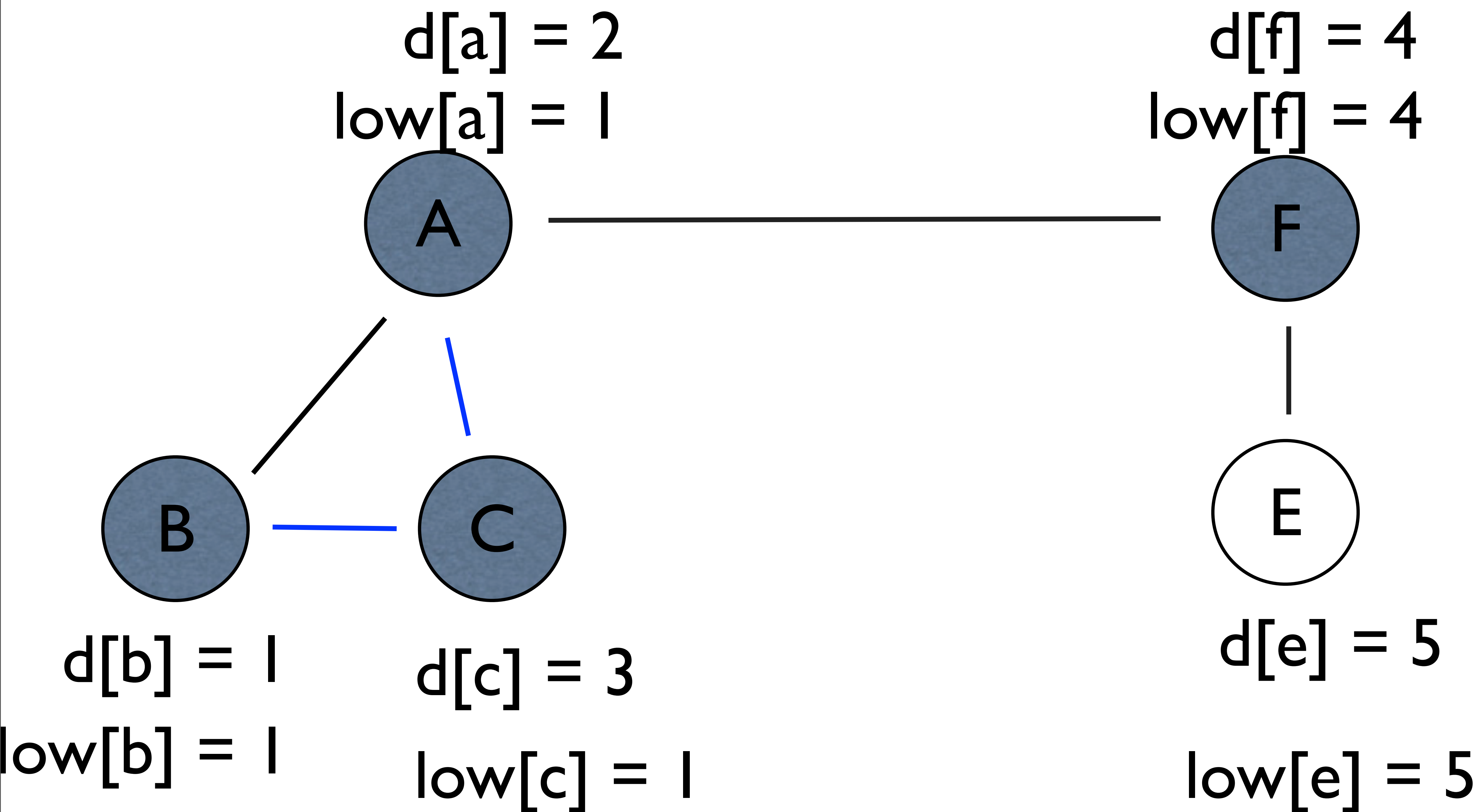
low[a] é alterado, pois está dentro do ciclo

Algoritmo de Tarjan

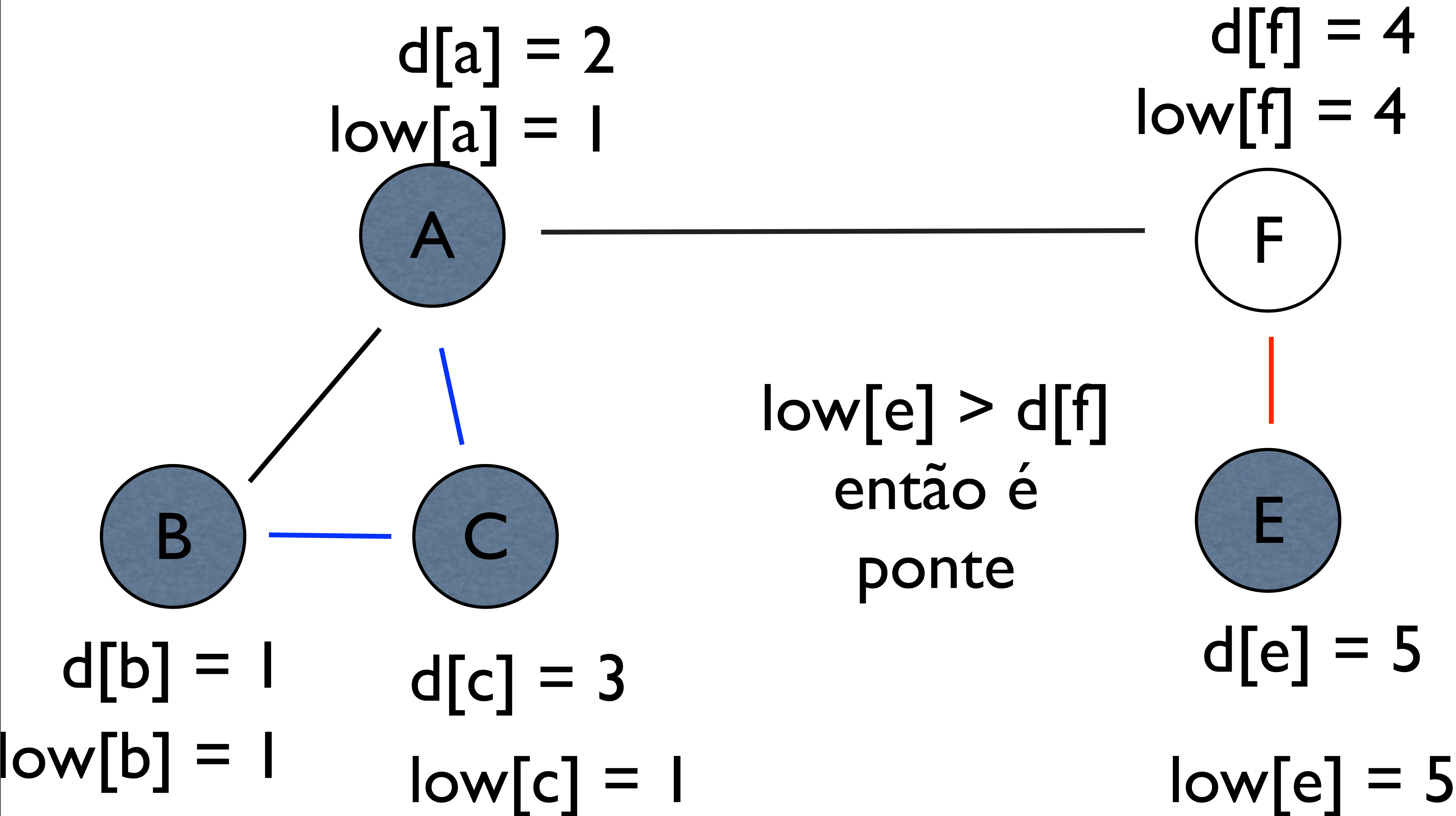


Continuamos a visitação

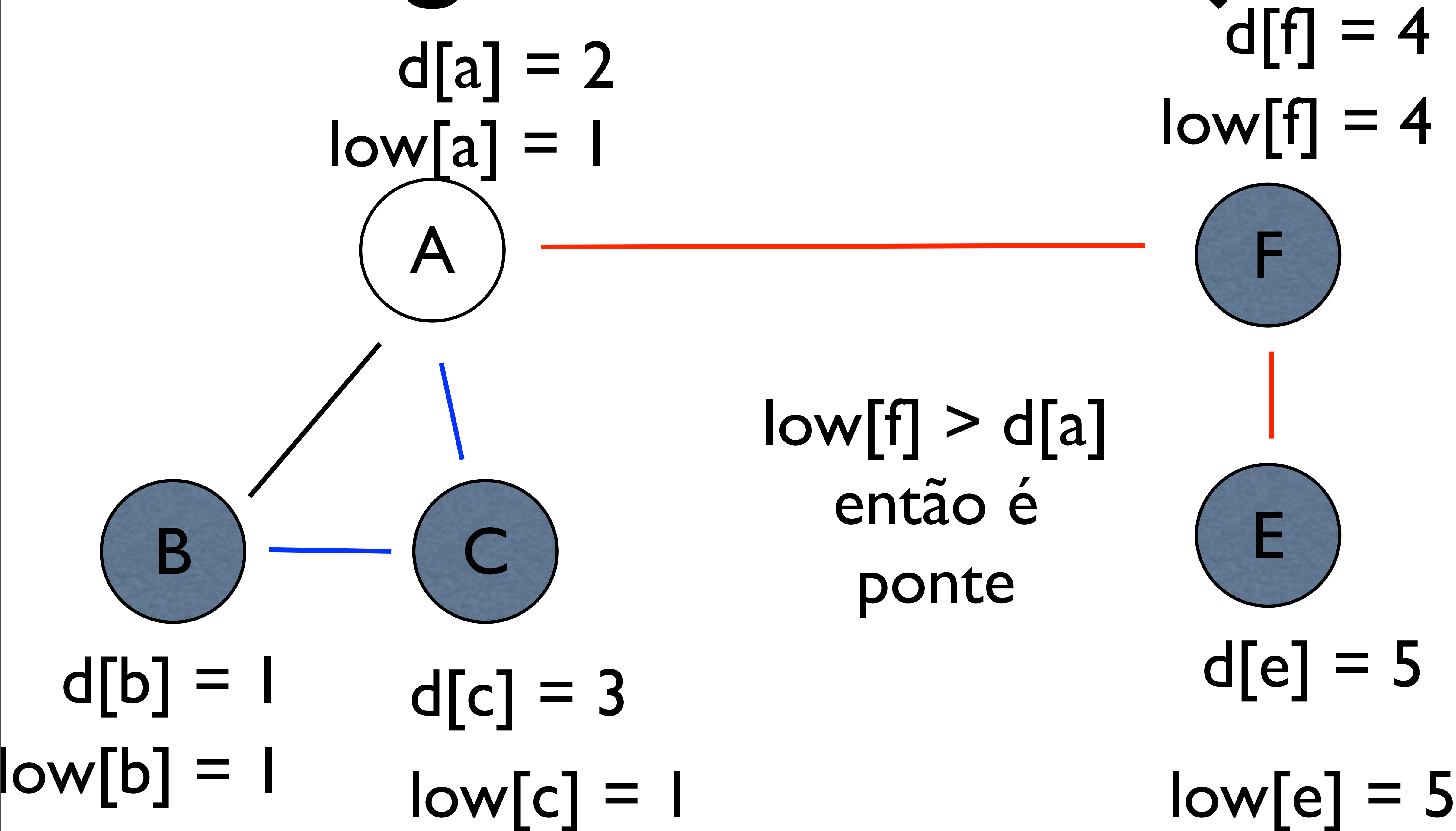
Algoritmo de Tarjan



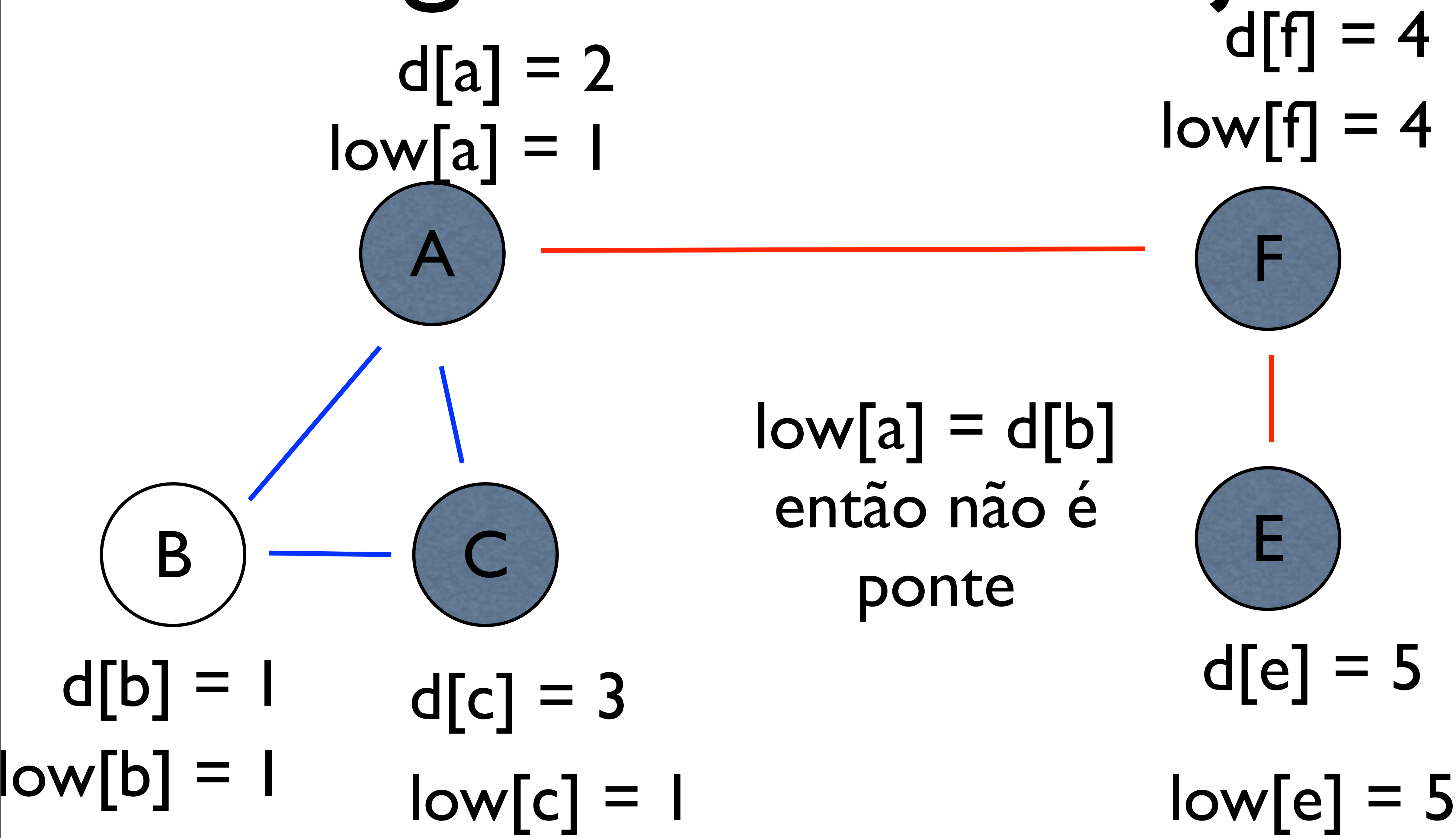
Algoritmo de Tarjan



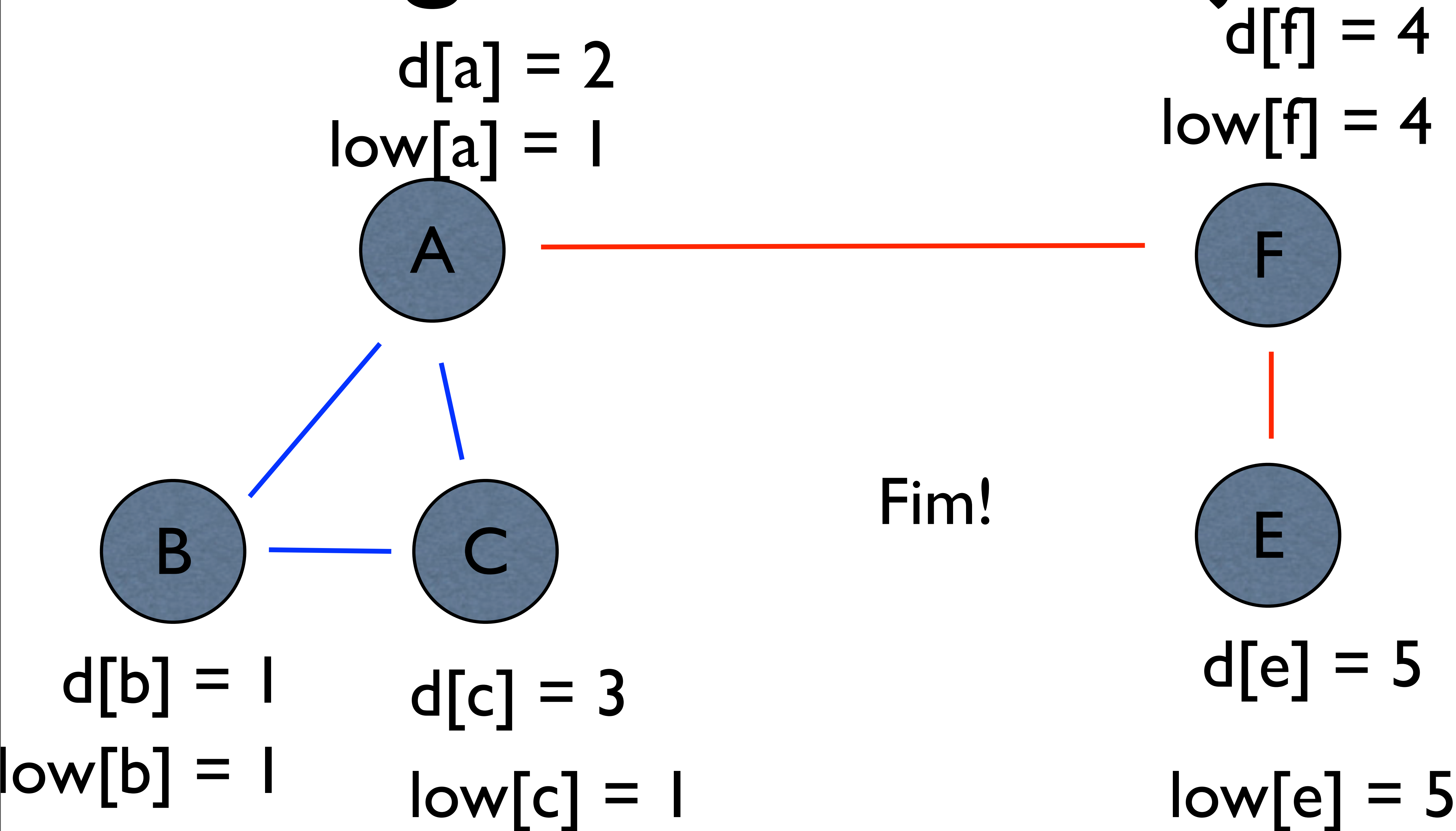
Algoritmo de Tarjan



Algoritmo de Tarjan



Algoritmo de Tarjan



Algoritmo de Tarjan

```
DFS(G)
    FOREACH (u ∈ V)
        color[u] = WHITE; p[u] = NULL; bridge[u] = FALSE ;
    time = 0
    FOREACH (u ∈ V)
        IF (color[u] == WHITE)
            DFS_Visit (u)
```

Algoritmo de Tarjan

```
DFS_Visit(u)
    d[u] = ++time
    low[u] = d[u]
    color[u] = GREY
    FOREACH (v ∈ Adj(u))
        IF (color[v] == WHITE)
            children[u]++
            p[v] = u
            DFS_Visit(v)
            low[u] = min(low[v], low[u])
            IF (low[v] > d[u])
                bridge[u] = TRUE
        ELSE IF (color[v] == GREY AND p[u] != v)
            low[u] = min(low[u], d[v])
    color[u] = BLACK
```

Complexidade

- Linear!!
  - O (V+E), como na busca em profundidade