Grafos 2-aresta-conexos e 2-conexos

Grafos 2-aresta-conexos e 2-conexos

1. 2-aresta-conexidade;

Grafos 2-aresta-conexos e 2-conexos

1. 2-aresta-conexidade; arestas de corte

Grafos 2-aresta-conexos e 2-conexos

1. 2-aresta-conexidade; arestas de corte

   - *Aresta de corte* ou *ponte*:

Grafos 2-aresta-conexos e 2-conexos

1. 2-aresta-conexidade; arestas de corte

   - *Aresta de corte* ou *ponte*: remoção aumenta o número de componentes do grafo

Grafos 2-aresta-conexos e 2-conexos

1. 2-aresta-conexidade; arestas de corte
   - *Aresta de corte* ou *ponte*: remoção aumenta o número de componentes do grafo
2. 2-conexidade;

Grafos 2-aresta-conexos e 2-conexos

1. 2-aresta-conexidade; arestas de corte
   - *Aresta de corte* ou *ponte*: remoção aumenta o número de componentes do grafo
2. 2-conexidade; vértices de corte

Grafos 2-aresta-conexos e 2-conexos

1. 2-aresta-conexidade; arestas de corte
   - *Aresta de corte* ou *ponte*: remoção aumenta o número de componentes do grafo
2. 2-conexidade; vértices de corte
   - *Vértice de corte* ou *ponto de articulação*:

Grafos 2-aresta-conexos e 2-conexos

1. 2-aresta-conexidade; arestas de corte
   - *Aresta de corte* ou *ponte*: remoção aumenta o número de componentes do grafo

2. 2-conexidade; vértices de corte
   - *Vértice de corte* ou *ponto de articulação*: remoção aumenta o número de componentes do grafo

Grafos 2-aresta-conexos e 2-conexos

1. 2-aresta-conexidade; arestas de corte
   - *Aresta de corte* ou *ponte*: remoção aumenta o número de componentes do grafo

2. 2-conexidade; vértices de corte
   - *Vértice de corte* ou *ponto de articulação*: remoção aumenta o número de componentes do grafo

Em geral:

Grafos 2-aresta-conexos e 2-conexos

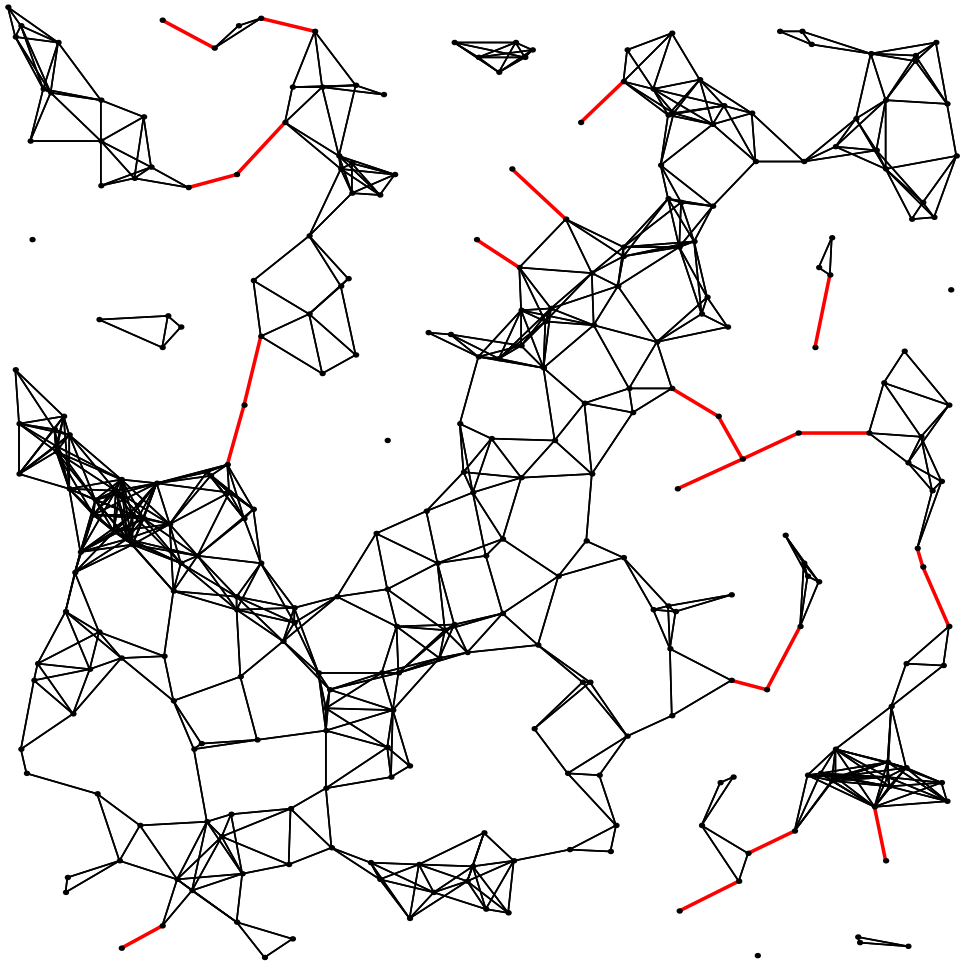
1. 2-aresta-conexidade; arestas de corte
   - *Aresta de corte* ou *ponte*: remoção aumenta o número de componentes do grafo

2. 2-conexidade; vértices de corte
   - *Vértice de corte* ou *ponto de articulação*: remoção aumenta o número de componentes do grafo

Em geral: $2 \mapsto k$ ($k \geq 2$)

Aresta-conexidade; pontes

Aresta-conexidade; pontes



Aresta-conexidade; pontes (cont.)

Aresta-conexidade; pontes (cont.)

G é 2-*aresta conexo* se G é conexo e não tem pontes.

Aresta-conexidade; pontes (cont.)

G é 2-*aresta conexo* se G é conexo e não tem pontes. (Outro termo: *aresta-conexo*)

Aresta-conexidade; pontes (cont.)

G é 2-*aresta conexo* se G é conexo e não tem pontes. (Outro termo: *aresta-conexo*)

Como determinar a existência/encontrar pontes?

Um exemplo (Fig. 18.16)

Um exemplo (Fig. 18.16)

```
yoshi@erdos:~/Main/www/2004i/mac328/exx/sink/ykDRAWs$ a.out 13 100 < fig18.6.gr
13 vertices, 16 edges
 0:  6  5  1
 1:  2  0
 2:  6  1
 3:  5  4
 4: 11  9  5  3
 5:  4  3  0
 6:  7  2  0
 7: 10  8  6
 8: 10  7
 9: 11  4
10:  8  7
11: 12  9  4
12: 11
```

```
   v : 0  1  2  3  4  5  6  7  8  9 10 11 12
 st[]: 0  2  6  4  5  0  0  6 10 11  7  4 11
pre[]: 0  6  5 12  8  7  1  2  4 11  3  9 10
low[]: 0  0  0  7  7  7  0  2  2  8  2  8 10
0-0 tree
  0-6 tree
    6-7 tree
      7-10 tree
        10-8 tree
          8-10 parent
          8-7 back
        10-7 parent
      7-8 down
      7-6 parent
```

```
  6-2 tree                 |          11-4 parent
    2-6 parent             |        4-9 down
    2-1 tree               |        4-5 parent
      1-2 parent           |        4-3 tree
      1-0 back             |          3-5 back
  6-0 parent               |          3-4 parent
0-5 tree                   |      5-3 down
  5-4 tree                 |      5-0 parent
    4-11 tree              |    0-1 down
      11-12 tree
        12-11 parent
      11-9 tree
        9-11 parent
        9-4 back
```

Propriedade que caracteriza pontes

Propriedade que caracteriza pontes

(Propriedade 18.5) Em qualquer ABP, um arco tipo árvore (v, w) é uma ponte se e só se não existe um arco ascendente conectando um descendente de w a um ancestral de w.

Propriedade que caracteriza pontes

(Propriedade 18.5) Em qualquer ABP, um arco tipo árvore (v, w) é uma ponte se e só se não existe um arco ascendente conectando um descendente de w a um ancestral de w.

- *Prova*: Se existe tal arco ascendente, claramente $\{v, w\}$ não é uma ponte.

Propriedade que caracteriza pontes

(Propriedade 18.5) Em qualquer ABP, um arco tipo árvore (v, w) é uma ponte se e só se não existe um arco ascendente conectando um descendente de w a um ancestral de w.

- *Prova*: Se existe tal arco ascendente, claramente $\{v, w\}$ não é uma ponte. Reciprocamente, se $\{v, w\}$ não é aresta de corte, então em $G - \{v, w\}$ os vértices v e w estão no mesmo componente. Isto é, existe um caminho de w a v em $G - \{v, w\}$. Qualquer tal caminho precisa conter um arco ascendente da subárvore enraizada em w com a outra ponta fora desta subárvore.

Algoritmo para identificação de pontes?

Algoritmo para identificação de pontes?

Para cada w computamos `low[w]`, o menor valor de `pre[u]` com u acessível a partir de w por um caminho contido na subárvore enraizada em w, seguida de um *arco* ascendente.

Programa 18.7

Programa 18.7

```
void bridgeR(Graph G, Edge e)
  { link t; int v, w = e.w;
    pre[w] = cnt++; low[w] = pre[w];
    for (t = G->adj[w]; t != NULL; t = t->next)
      if (pre[v = t->v] == -1)
        {
          bridgeR(G, EDGE(w, v));
          if (low[w] > low[v]) low[w] = low[v];
          if (low[v] == pre[v])
            { bcnt++; printf("%d-%d\n", w, v); }
        }
      else if (v != e.v)
        if (low[w] > pre[v]) low[w] = pre[v];
  }
```