Grafos: Biconectividade

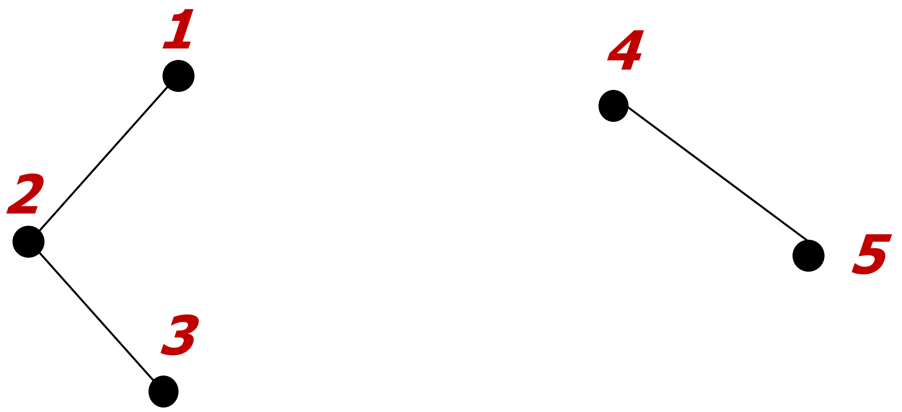
Aula 12

Prof. André Britto
Modificada por: Prof. Breno Piva

Vértices de Corte (Articulações)

Num grafo *G* um vértice *v* é um **vértice de corte** se o número de componentes conexas em G-v é maior que o número de componentes conexas em G.

Ex.:



Vértices de Corte (Articulações)

Se G é um grafo simples, não trivial, então um vértice v em G é **vértice de corte** se $C(G-v) > C(G)$, onde $C(r)$ denota o número de componentes de r.

Teorema

Um vértice v numa árvore é vértice de corte **se e somente se** $g(v) > 1$.

Vértices de Corte (Articulações)

Teorema

Um vértice *v* numa árvore é vértice de corte **se e somente se** ***g(v) > 1***

Prova

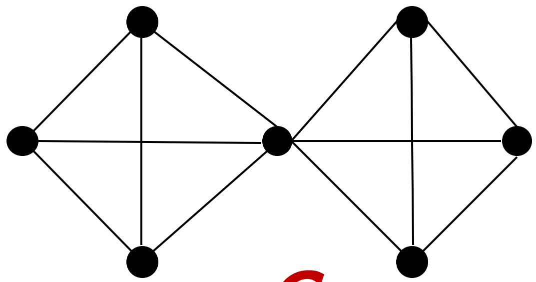
=>Suponha que a hipótese seja válida e que g(v) ≤ 1. Se *g(v) = 0*, então temos um grafo trivial e por conseguinte *v* não é de corte. Se *g(v) = 1 e G* é árvore*, v* é uma folha *e G-v* é conexo, *C(G-v) = C(v) = 1*. Portanto, *v* não é de corte.

<= Suponha que *g(v) >1* e sejam *u* e *w* dois vértices adjacentes de *v*. Como *G* é árvore, pelo teorema fundamental de árvores, o caminho *(u, v, w)* é o único caminho entre *u* e *w* em *G*. Portanto, não existe nenhum caminho entre *u* e *w* em *G-v* e *u* e *w* estão em componentes distintos. Assim, *C(G-v) > C(G)=1*. Logo *v* é vértice de corte.

Biconexidade

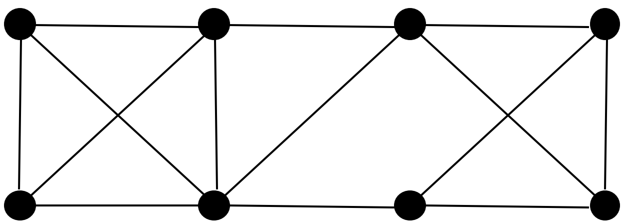
Um grafo *G* é **biconexo** se existem pelo menos dois caminhos disjuntos ligando qualquer par *(v,w)* de vértices de *G*.

Ex.:



G

G é conexo mas não é biconexo



G′

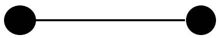
G′ é biconexo

Biconexidade

Proposição 1

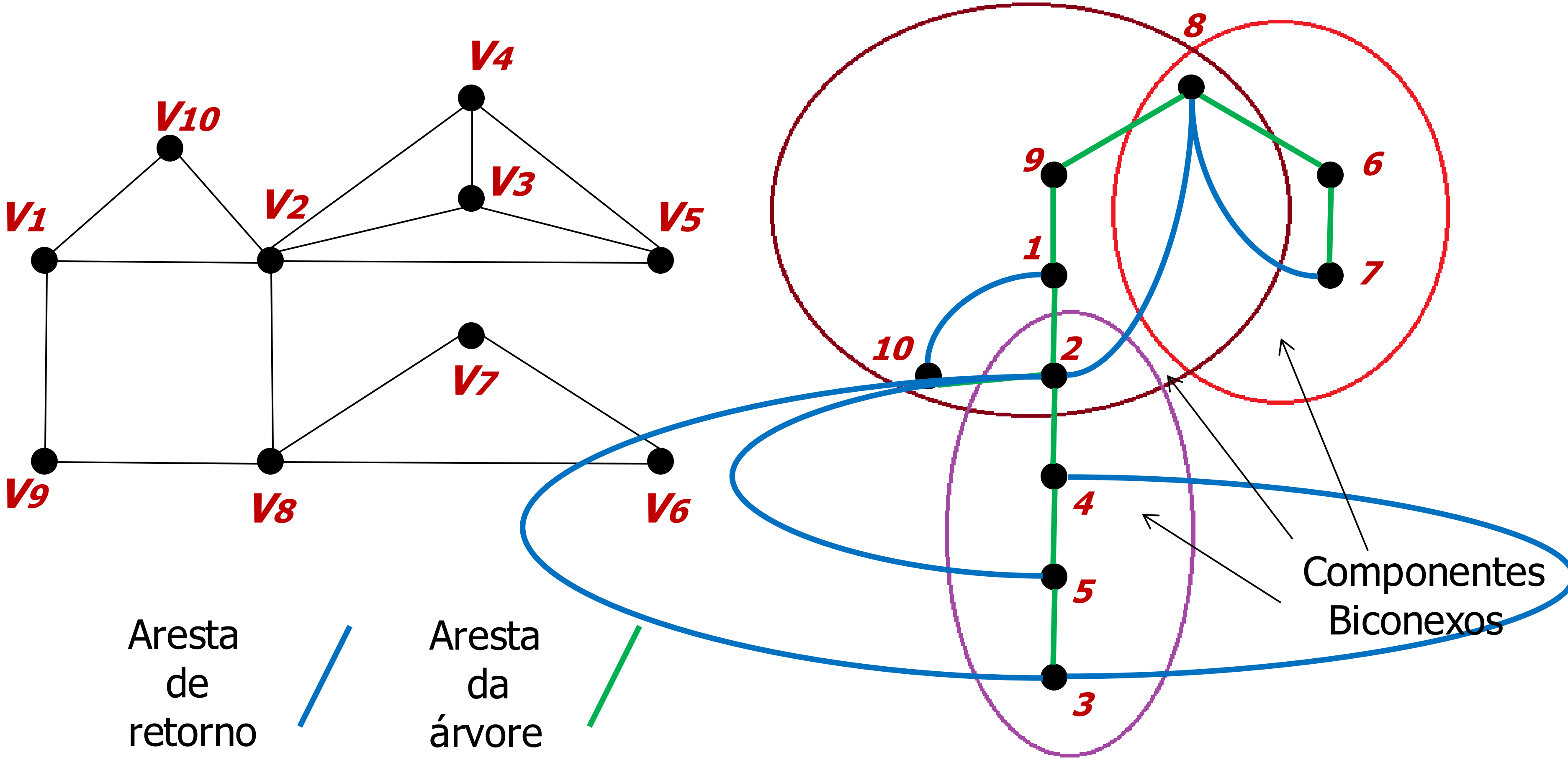
Um grafo com $|VG| \geq 3$ é biconexo se e somente se não contém vértice de corte.

Provar como exercício!

Componentes biconexos de um grafo *G* são os subgrafos maximais de *G* que sejam biconexos em vértices ou isomorfos a K_2 ().

Aplicação de Busca em Profundidade

Determinação dos componentes biconexos de um grafo.



Aplicação de Busca em Profundidade

Proposição 2

Seja G' um subgrafo de G e $v \in VG'$ (onde v não é a raiz de G) o primeiro vértice alcançado pelo algoritmo de B.P. aplicado em G. O vértice v é vértice de corte se e somente se não existem arestas de retorno de vértices em VG'-v a ancestrais de v na árvore de profundidade.

Provar como exercício!

Obs: Para o caso da raiz basta observar o número de filhos na árvore de profundidade. Se for maior que um, então v é de corte.

Aplicação de Busca em Profundidade

Determinação de Biconectividade

- Identificação de Vértices de Corte (Proposição 1)
- Observação das arestas de retorno na Busca em Profundidade (Proposição 2)

Aplicação de Busca em Profundidade

Como computar os vértices alcançados pelas arestas de retorno para poder determinar se v é de corte?

Aplicação de Busca em Profundidade

Como computar os vértices alcançados pelas arestas de retorno para poder determinar se v é de corte?

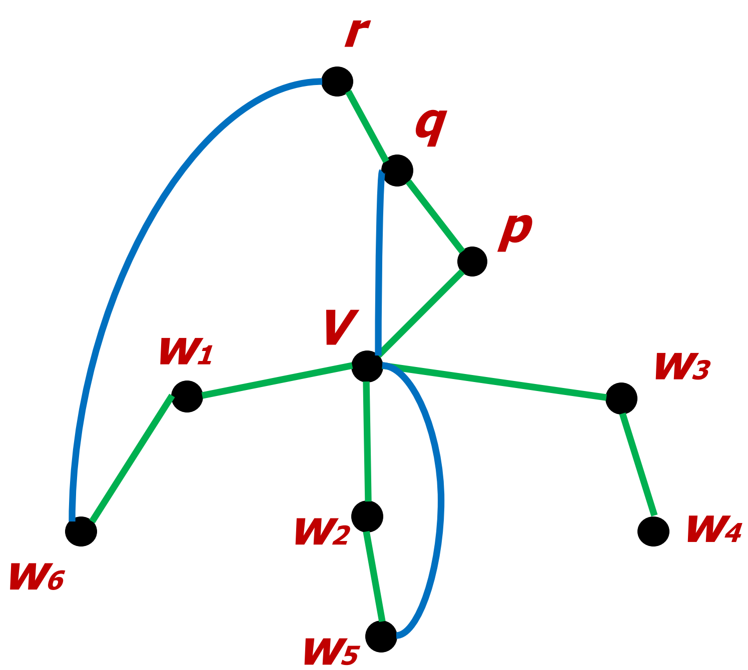
Armazenando informação na EAD

High(v) é o mais alto vértice na árvore de profundidade ligado por uma aresta de retorno a v ou a um descendente de v nesta árvore.

Aplicação de Busca em Profundidade

Se w_1, w_2,..., w_k são filhos de v então *High(v)* pode ser facilmente computado se soubermos os *High*(w_1), *High*(w_2),...,*High*(w_k). Ele é o maior dentre os *High* dos filhos de v e dos vértices ligados a v por uma aresta de retorno.

Ex.:



High(w_1) = r
High(w_2) = v
High(w_3) = w_3
High(v) = r

Aplicação de Busca em Profundidade

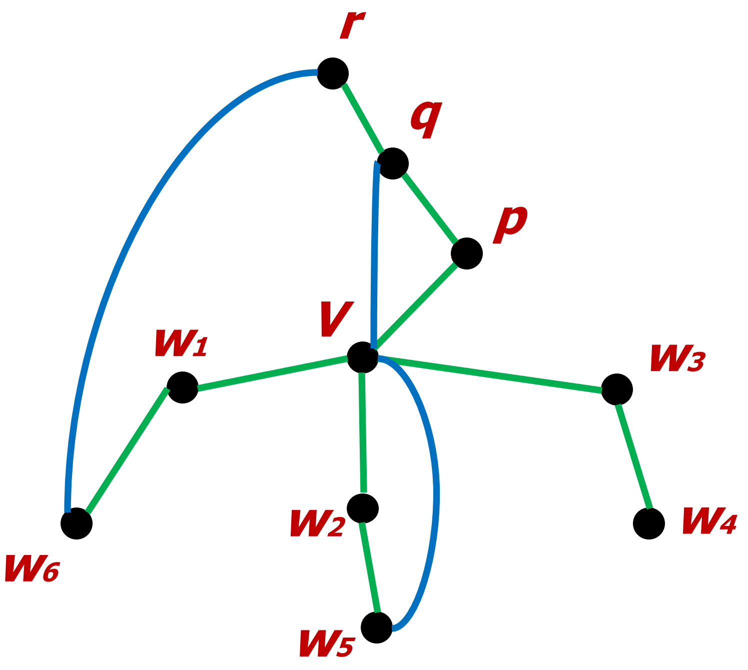
Proposição 3

Um vértice *v* é vértice de corte se possui um filho *w*, tal que *High(w)* não é um vértice mais alto que *v* na árvore de profundidade.

Observe que a Proposição 3 é análoga à Proposição 2 e portanto a prova é similar.

Aplicação de Busca em Profundidade

Exemplo: *v* é articulação pois $High(w_2) = v$ e $High(w_3) = w_3$. Existe mais alguma articulação neste grafo?



Componentes Biconexos

O algoritmo usa P.E. para determinar *High* e uma pilha auxiliar para determinação dos componentes biconexos.

```
algoritmo componentesBiconexos(G,v,n);
{dados: grafo G conexo e não orientado, v(raiz de busca) e n número
de vértices de G.}
procedimento BC(v);
   início
      PE[v]:= num;
      num:= num+1;
      empilhe(pilha,v); {Pilha está vazia no início}
      High[v]:= PE[v]; {valor inicial do High}
      para todas as arestas (v,w) faça
         início
            empilhe(pilha,(v,w));
```

Componentes Biconexos

```
Se w não é pai de v então
    Se PE[w] = 0 então
        início
            BC(w);
            Se High[w] ≥ PE[v] então
            {v desconecta w do resto do grafo}
                Início
                    {remova todas as arestas e vértices
                     da pilha até encontrar(v,w)e marque
                     o subgrafo que eles formam como
                     componente biconexo, marque v como
                     pertencente ao componente}
                fim
            High[v]:= min(High[v],High[w]);
        fim
```

Componentes Biconexos

```
            └─ senão {aresta de retorno}
                  High[v]:= min(High[v],PE[w]);
      fim {fim das arestas}
   fim {fim do procedimento}

início {corpo do algoritmo}
   para todos os vértices w de G faça
      PE[w]:= 0;
   num:= 1;
   pilhaVazia(Pilha);
   BC(v);
fim
```

Componentes Biconexos

```
            senão {aresta de retorno}
                High[v]:= min(High[v],PE[w]);
        fim {fim das arestas}
    fim {fim do procedimento}

início {corpo do algoritmo}
    para todos os vértices w de G faça
        PE[w]:= 0;
    num:= 1;
    pilhaVazia(Pilha);
    BC(v);
fim
```

Complexidade?

Componentes Biconexos

```
        └─ senão {aresta de retorno}
                High[v]:= min(High[v],PE[w]);
      fim {fim das arestas}
   fim {fim do procedimento}

início {corpo do algoritmo}
   para todos os vértices w de G faça
      PE[w]:= 0;
   num:= 1;
   pilhaVazia(Pilha);
   BC(v);
fim
```

Complexidade?

O(*n+m*)

Exercícios Recomendados

- Szwarcfiter: 4.10-4.14
- Udi Manber: 7.14, 7.83-7.85
- Bondy e Murty: 2.3.1-2.3.2.

Referências

- Seção 4.4 do Szwarcfiter, J. L., *Grafos e Algoritmos Computacionais,* Ed. Campus, 1983.

- Seção 8.3 do Jungnickel, D., *Graphs, Networks and Algorithms,* Springer, 2007.

- Seções 2.2 e 2.3 do Bondy J. A. e Murty U. S. R., *Graph Theory with Applications*, Elsevier, 1976.