

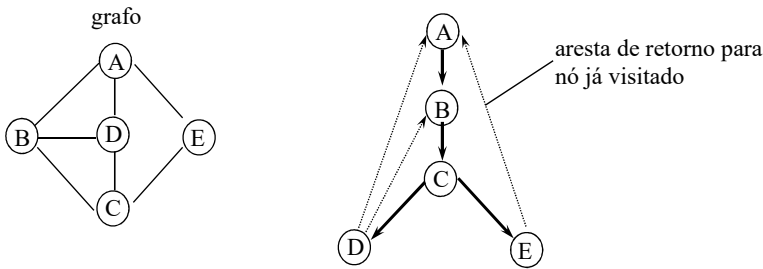
Algoritmos em Grafos: Conectividade

R. Rossetti, L. Ferreira, H. L. Cardoso, F. Andrade
FEUP, MIEIC, CAL

Grafos não dirigidos

Conectividade

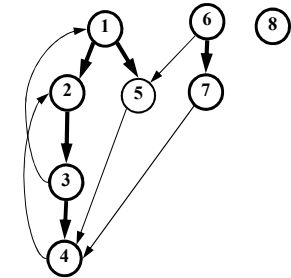
- Um grafo não dirigido é conexo sse uma pesquisa em profundidade a começar em qualquer nó visita todos os nós



Pesquisa em profundidade

```
1: class Graph { ...
2:   void dfs() {
3:     for (Vertex v : vertexSet)
4:       v.visited = false;
5:     for (Vertex v : vertexSet)
6:       if (! v.visited)
7:         dfs(v);
           //v passa a ser raiz duma árvore dfs
8:   }
9:   void dfs( Vertex v ) {
10:    v.visited = true;
11:    // fazer qualquer coisa c/ v aqui
12:    for (Edge e : v.adj)
13:      if ( ! e.dest.visited )
14:        dfs( e.dest );
15:    // ou aqui
16:  }
17: }
```

Exemplo em grafo dirigido (vértices numerados por ordem de visita e dispostos por profundidade de recursão):



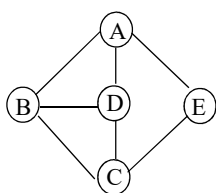
Arestas a traço forte: árvore de expansão em profundidade

Na DFS podem ser produzidas várias árvores, porque a pesquisa pode ser repetida a partir de várias fontes (ao contrário da BFS que só produz uma). O conjunto das várias árvores é conhecido como Floresta DFS.

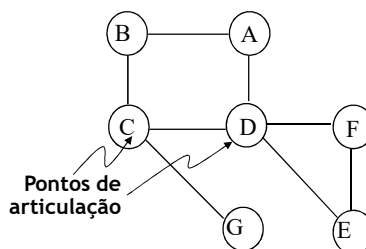
Biconectividade e Pontos de Articulação

- Grafo conexo não dirigido é **biconexo** se não existe nenhum vértice cuja remoção torne o resto do grafo desconexo
- **Pontos de articulação**: vértices que tornam o grafo desconexo
- Aplicação - rede com tolerância a falhas

Biconexo



Não biconexo



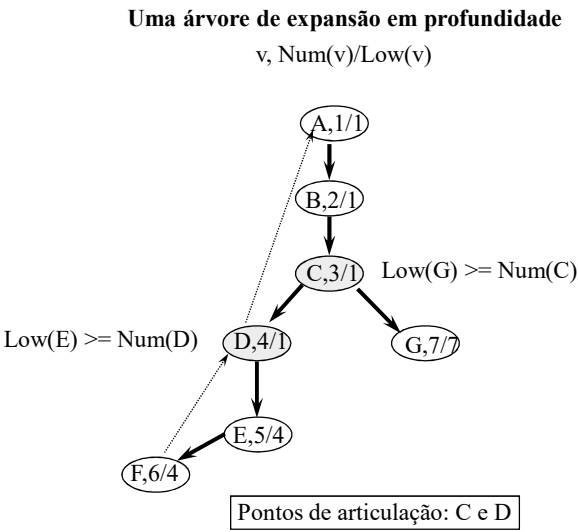
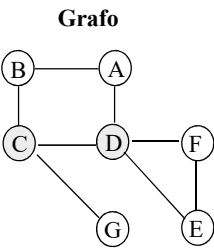
Algoritmo de detecção de pontos de articulação

- Início num vértice qualquer
- Pesquisa em profundidade, numerando os vértices ao visitá-los — $\text{Num}(v)$, em pré-ordem (antes de visitar adjacentes)
- Para cada vértice v , na árvore de visita em profundidade, calcular $\text{Low}(v)$: o menor número de vértice que se atinge com zero ou mais arestas na árvore e possivelmente uma aresta de retorno
- Vértice v é ponto de articulação se tiver um filho w tal que $\text{Low}(w) \geq \text{Num}(v)$
- A raiz é ponto de articulação sse tiver mais que um filho na árvore

Cálculo de Low(v)

- Low(v) é mínimo de
 - Num(v)
 - o menor Num(w) de todas as arestas (v, w) de retorno
 - o menor Low(w) de todas as arestas (v, w) da árvore
- Na visita em profundidade, inicializa-se Low(v)=Num(v) antes de visitar adjacentes, e vai-se actualizando o valor de Low(v) a seguir a visita a cada adjacente
- Realizável em tempo $O(|E| + |V|)$

Exemplo



Pseudo-código

```
// Procura Pontos de Articulação usando dfs
// Contador global e inicializado a 1
void findArt( Vertex v) {
    v.visited = true;
    v.low = v.num = counter++;
    for each w adjacent to v
        if( !w.visited ) {                // ramo da árvore
            w.parent = v;
            findArt(w);
            v.low = min(v.low, w.low);
            if(w.low >= v.num )
                System.out.println(v, "Ponto de articulação");
        }
        else
            if ( v.parent != w )          //aresta de retorno
                v.low = min(v.low, w.num);
    }
```



Universidade do Porto
Faculdade de Engenharia

Algoritmos em Grafos: Conectividade, CAL

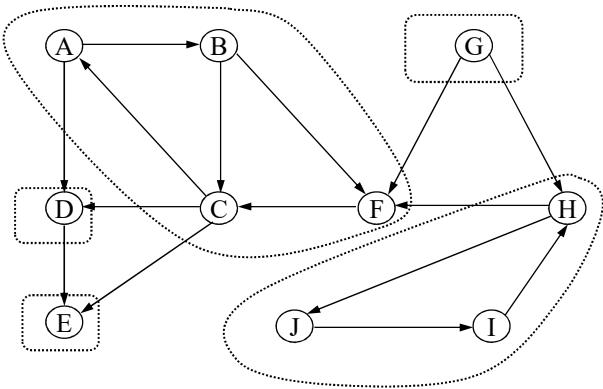
Grafos dirigidos



Universidade do Porto
Faculdade de Engenharia

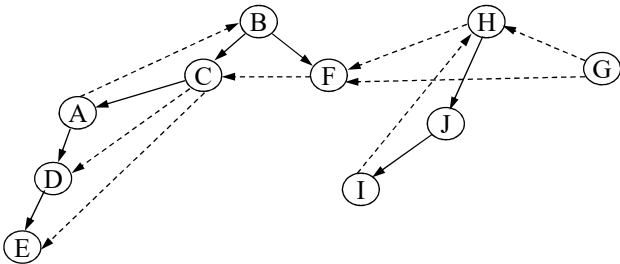
Algoritmos em Grafos: Conectividade, CAL

Componentes fortemente conexos



Árvore de expansão

- Pesquisa em profundidade induz uma árvore/floresta de expansão
- Para além das arestas genuínas da árvore, há arestas para nós já marcados
 - arestas de retorno para um antepassado – (A,B), (I,H)
 - arestas de avanço para um descendente – (C,D), (C,E)
 - arestas cruzadas para um nó não relacionado – (F,C), (G,F)

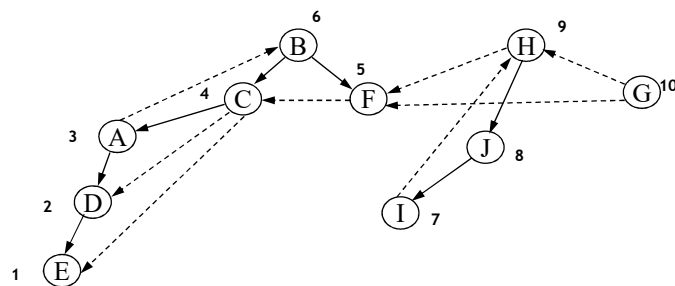


Componentes fortemente conexos

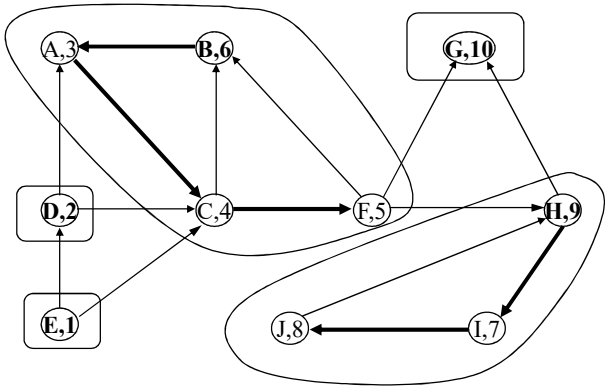
■ Método:

- Pesquisa em profundidade no grafo G determina floresta de expansão, numerando vértices em pós-ordem (ordem inversa de numeração em pré-ordem)
- Inverter todas as arestas de G (grafo resultante é G_r)
- Segunda pesquisa em profundidade, em G_r , começando sempre pelo vértice de numeração mais alta ainda não visitado
- Cada árvore obtida é um componente fortemente conexo, i.e., a partir de um qualquer dos nós pode chegar-se a todos os outros

Numeração em pós-ordem

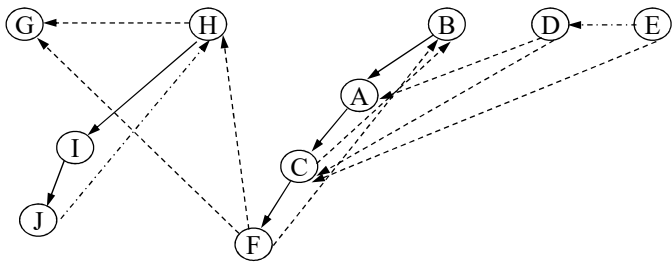


Inversão das arestas e nova visita



G_r : obtido de G por inversão de todas as arestas
Numeração: da travessia de G em pós-ordem

Componentes fortemente conexos



Travessia em pós-ordem de G_r
Componentes fortemente conexos:
{G}, {H, I, J}, {B, A, C, F}, {D}, {E}

Referência e informação adicional

- “Data Structures and Algorithm Analysis in Java”, Second Edition, Mark Allen Weiss, Addison Wesley, 2006