Projeto Olímpico de Programação

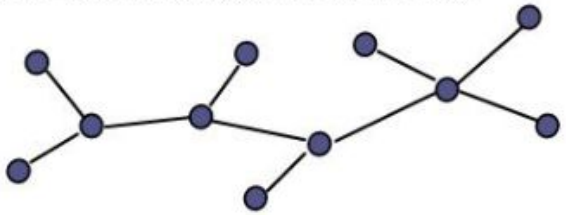
Cycles, Bridges, Articulation Points and SCC

Paulo Felipe Feitosa paulofelipe.feitosa@gmail.com

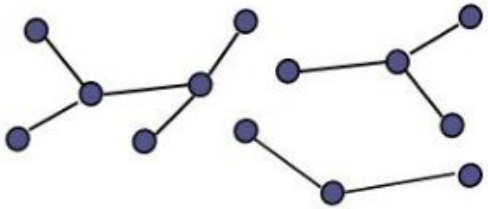
Cycles - A little Theory

- Para identificação de Ciclos em Grafos, precisamos de algumas definições.
  - Uma busca em profundidade (DFS) forma uma floresta de busca em profundidade (Spanning Forests), que contém várias árvores de busca em profundidade (Spanning Trees).

- Árvore: grafo conexo sem circuitos.



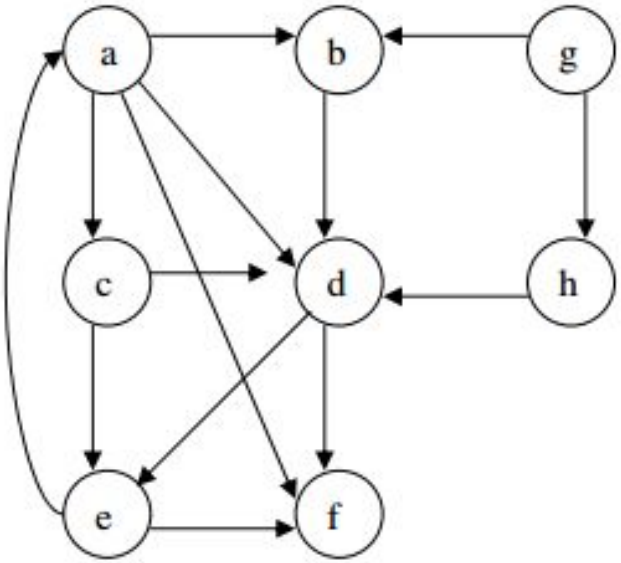
- Floresta: grafo cujas componentes conexas são árvores.



Cycles - A little Theory

- O Vértice Raiz da Busca em Profundidade é quem configura as Spanning Forests e Spanning Trees, é necessário ter em mente isso.

- Problema: Informe se um dado Grafo tem pelo menos um ciclo. Note que o grafo é direcionado.

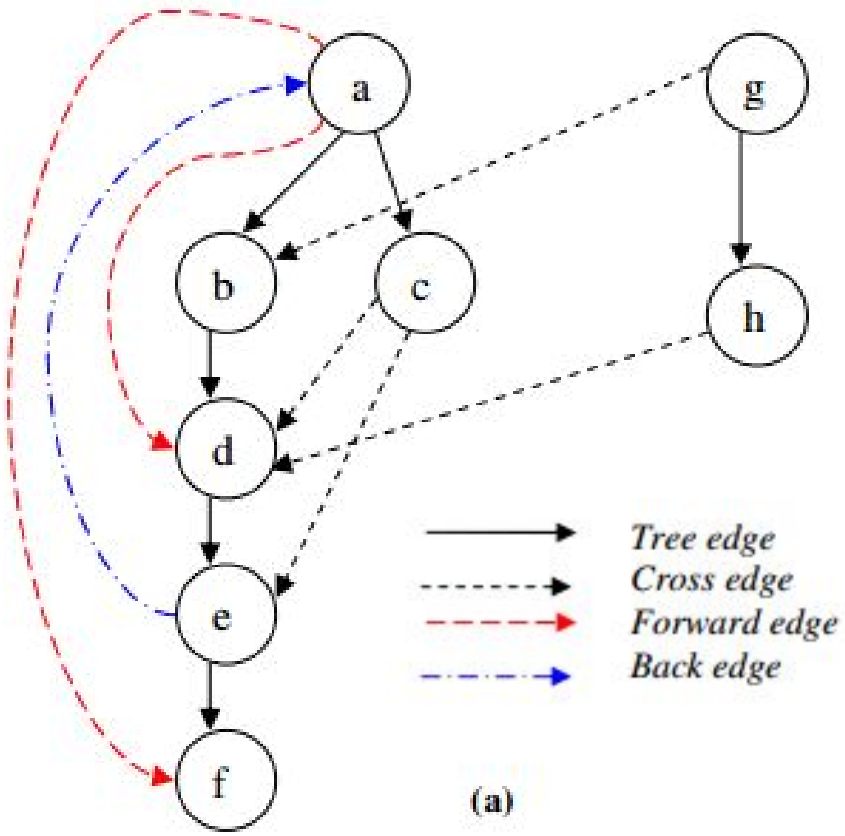


Cycles - A little Theory

- De acordo com a Teoria, nós podemos classificar as arestas de um grafo, que está sendo explorado por uma DFS(**U**), da seguinte forma:
  - Se o vértice **V** ainda não foi visitado, então, a aresta (**U**, **V**) é uma **Tree Edge** (da árvore de busca).
  - Senão, se o vértice **V** já foi visitado, então:
    - Se **V** é um vértice **antecessor** (sendo explorado) ao vértice **U**, então:
      - Se **V** é um vértice **pai** de **U**, então a aresta (**U**, **V**) é **Bidirecional**.
      - Senão, a aresta (**U**, **V**) é uma **Back Edge** (**Ciclo Encontrado**).
    - Se **V** é um vértice **descendente** de **U**, então a aresta (**U**, **V**) é uma **Forward Edge**.
    - Senão, a arestão (**U**, **V**) é uma **Cross Edge**.

Cycles

- Calma que a implementação é muito simples!!!
  - Classification of Edges



Cycles

- Pra que serve isso Paulo?
  - Os próximos assuntos que vamos estudar: Pontes e Articulações, Componentes Fortemente Conectadas.

- Questão Metrô

Articulation Points and Bridges (Undirected Graph)

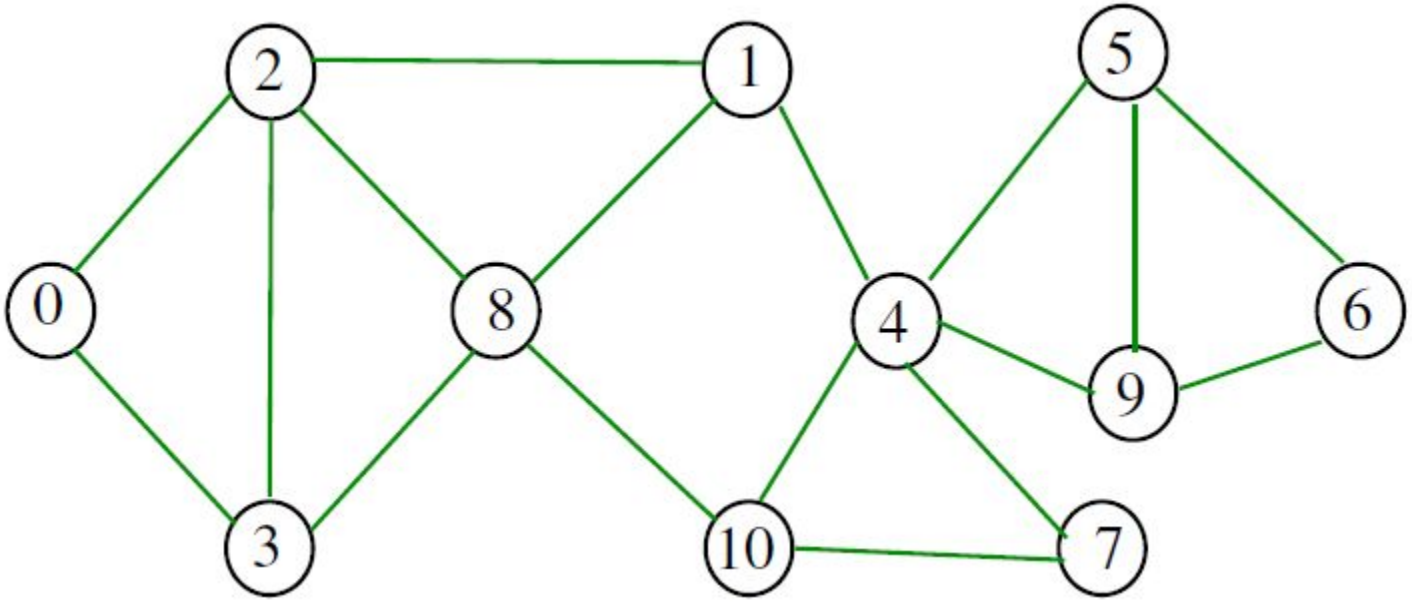
- Um Ponto de Articulação (Vértice de Corte) é definido como um Vértice que, quando removido, deixa o Grafo desconectado.

- Outra definição: Um Ponto de Articulação é um vértice de um grafo, tal que, sua remoção provoca um aumento no número de componentes conectadas do Grafo.

- OBS: Um grafo sem nenhum ponto de Articulação é chamado de “Biconectado”.

Articulation Points and Bridges (Undirected Graph)

- Similarmente, uma Ponte é definida como uma aresta em um grafo, tal que, sua remoção deixa G desconectado (ou provoca o aumento no número de componentes conectadas de G).
- Encontre as Pontes e os Pontos de Articulações do Grafo abaixo.



Articulation Points and Bridges (Undirected Graph)

- Algoritmo Ingênuo para resolver o problema de Pontos de Articulação:
  - Conte o número de Componentes Conectadas do Grafo original;
  - Para cada vértice **V**:
    - "Remova o vértice **V**";
    - Conte novamente o número de Componentes Conectadas do Grafo;
    - Se a quantidade de Componentes Conectadas aumenta, então, **V** é um ponto de Articulação.
- $O(V^2 + VE)$.

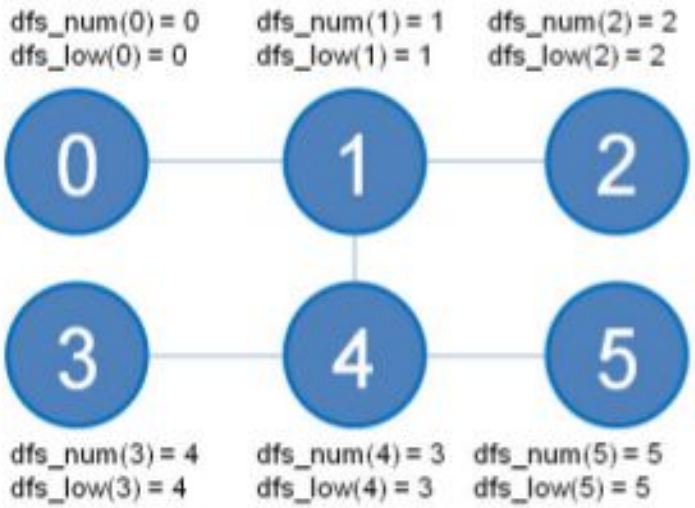
Articulation Points and Bridges (Undirected Graph)

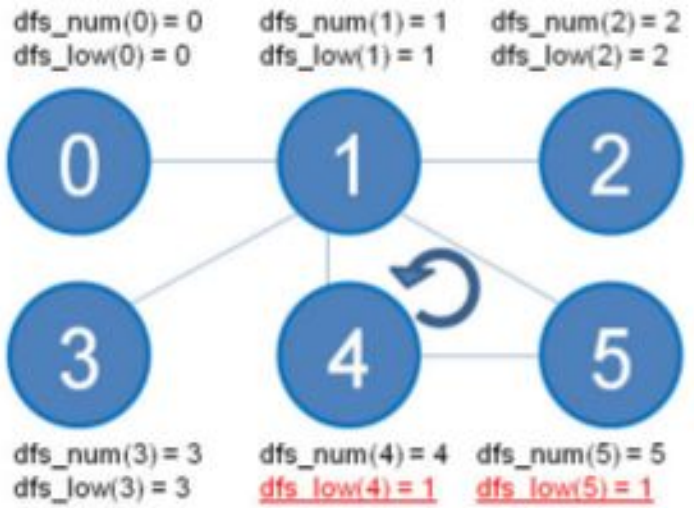
- Algoritmo de Hopcroft e Tarjan - O(V + E):
- Nós vamos manter dois números: dfs_desc(u) e dfs_low(u). Onde, dfs_desc(u) é o tempo de descoberta do vértice **U** e dfs_low(u) é o menor tempo de descoberta alcançado na Árvore de Busca em Profundidade produzida por DFS(**U**).
- Quando um vértice é visitado, dfs_desc(u) = dfs_low(u). Então, dfs_low(u) só pode ser menor que dfs_desc(u) se existe um ciclo (**Back Edge**) na Árvore de Busca em Profundidade de **U**. (Revisar classificação de Arestas).

Articulation Points and Bridges (Undirected Graph)

- Arestas Bidirecionais não são contabilizadas como ciclos.

- Exemplo: (A DFS começa com DFS(0)) e dfs_num == dfs_desc.





Articulation Points and Bridges (Undirected Graph)

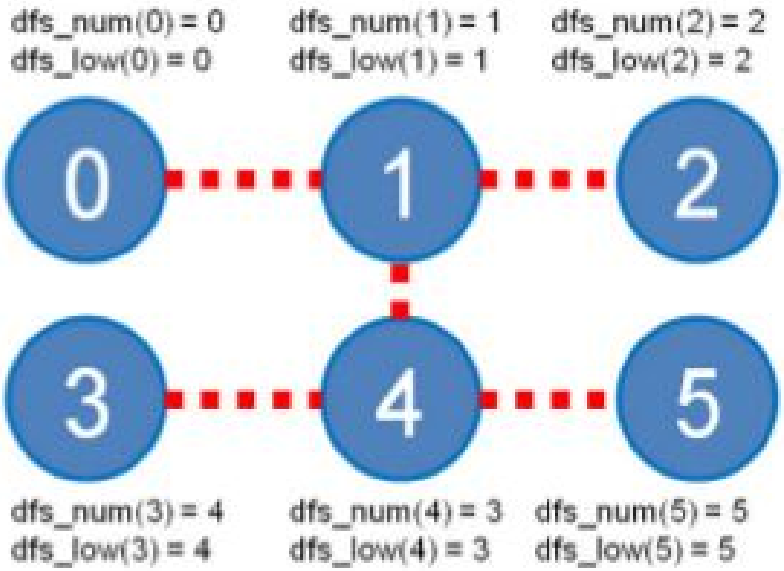
- Quando estamos em um vértice **U** com **V** como seu vizinho e dfs_low(v) >= dfs_desc(u), então **U** é um ponto de Articulação.
- O fato de dfs_low(v) não ser menor do que dfs_desc(u) implica que não existe **Back Edge** do vértice **V** que pode chegar em outro vértice **W** com um menor dfs_desc(w) do que dfs_desc(u).
- Um vértice **W** com um dfs_desc(w) menor do que dfs_desc(u) de **U** implica que **W** é um antecessor de **U** na Árvore de Busca em Profundidade.

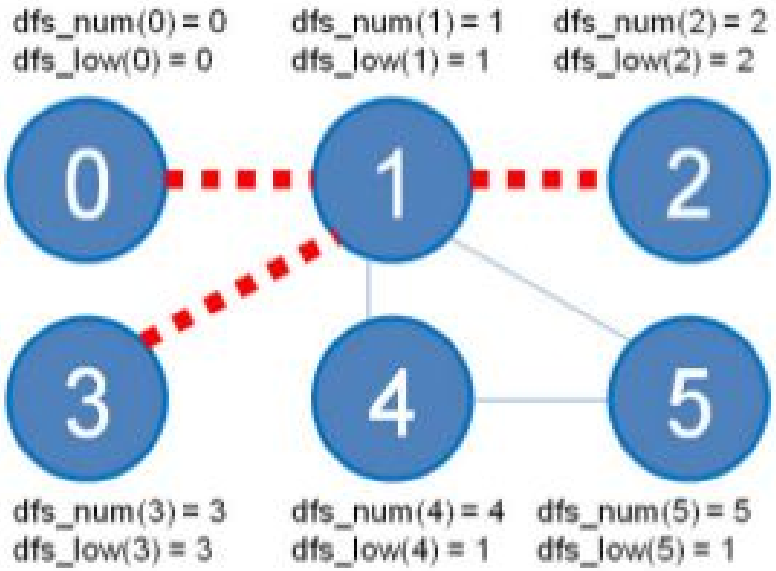
Articulation Points and Bridges (Undirected Graph)

- Se dfs_low(v) não for menor do que dfs_desc(u), então, para chegar em um antecessor de **U** a partir de **V** é necessário passar por **U**, logo, se o vértice **U** for removido, o grafo ficará desconectado.
- Contudo, ainda existe um caso especial: A raiz da Árvore de Busca em Profundidade (O vértice escolhido para a chamada da DFS) é considerada uma Articulação se ele tem mais de um filho na Árvore de Busca em Profundidade.

Articulation Points and Bridges (Undirected Graph)

- Exemplo:





- O processo para encontrar pontes é parecido. Quando dfs_low(v) > dfs_desc(u), então, a aresta de (**U**, **V**) é uma ponte.

Articulation Points and Bridges (Undirected Graph)

- Sem desespero, porque a implementação é simples!!!
  - Articulation Point and Bridges
- Pra que danado serve isso Paulo?
  - Para encontrar Pontos de Articulações e Pontes em Grafos, essa eu tenho certeza!
  - Articulation Points and Bridges normalmente são parte de uma solução para um determinado problema.
  - Vai ficar mais legal quando chegarmos à Strongly Connected Components. (Me lembrem disso!!!).

Strongly Connected Components (Directed Graph)

- Uma Componentes Fortemente Conectada em um Grafo Direcionado é definida da seguinte forma: Para qualquer par de vértice **U** e **V**, contidos na SCC, existe um caminho de **U** para **V** e de **V** para **U**.
- Qual a diferença entre **Componentes Fortemente Conectadas** e **Componentes Conectadas**?

Strongly Connected Components (Directed Graph)

- Algoritmo de Tarjan - O(V + E):
- A ideia básica do Algoritmo é que SCCs (Strongly Connected Components) formam **subárvores** na Árvore de Busca em Profundidade.
- Vamos usar novamente dfs_num(u) e dfs_low(u). Além disso, uma pilha será implementada para manter os vértices que estão correntemente sendo explorados.
- Porém, agora a condição para atualizar o valor de dfs_low(u) é a seguinte: apenas vértices que estão sendo correntemente explorados podem atualizar o valor de dfs_low(u).

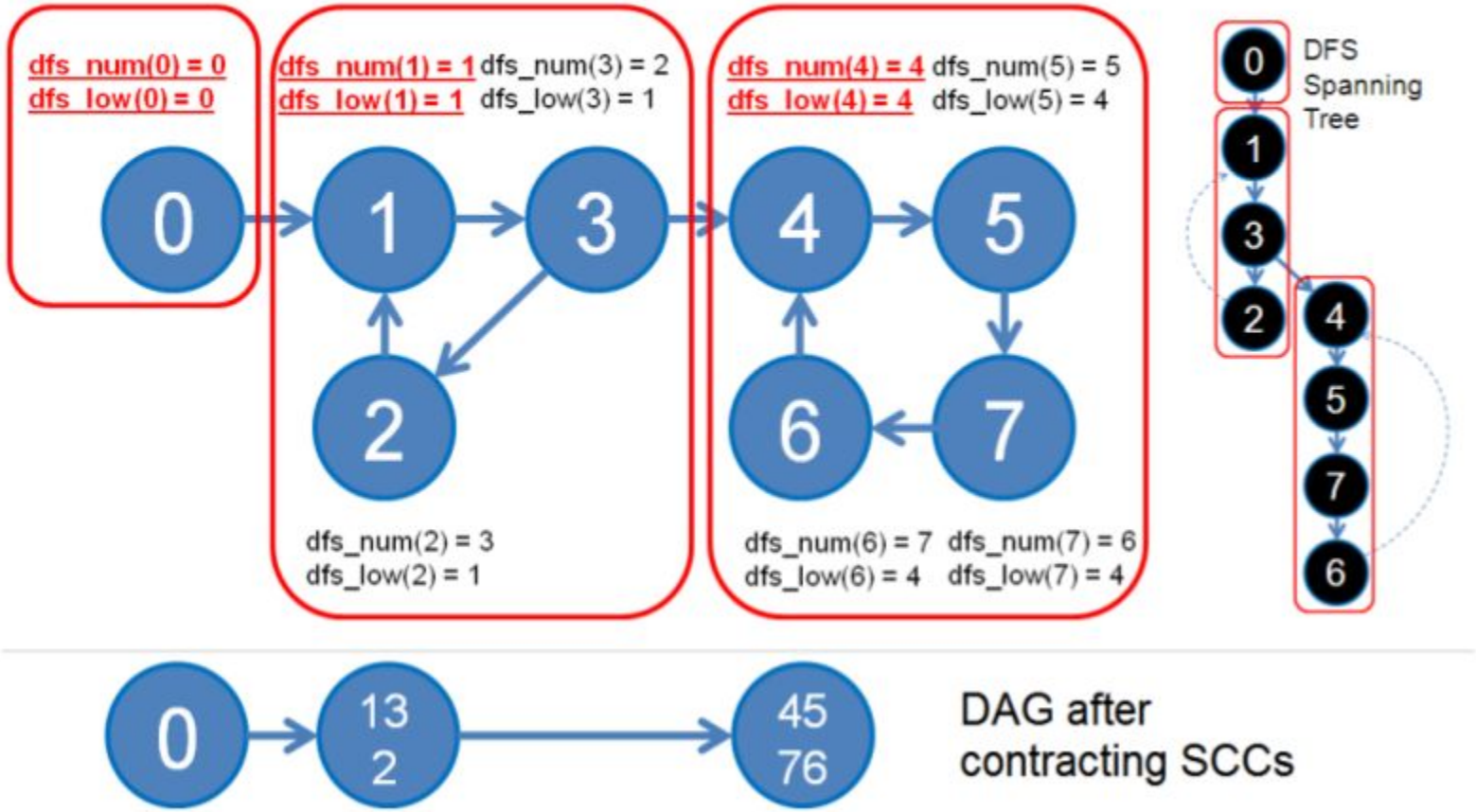
Strongly Connected Components (Directed Graph)

- Agora, para este algoritmo, se tivermos um vértice **U** na Árvore de Busca em Profundidade com dfs_low(u) = dfs_num(u), então, podemos concluir que ele é uma raiz de uma subtree, consequentemente, uma raiz de uma SCC.

- E os membros da SCC de **U** são identificados desempilhando a pilha, implementada, até encontrarmos o vértice **U** (raiz da SCC) denovo.

Strongly Connected Components (Directed Graph)

- Exemplo:



Strongly Connected Components (Directed Graph)

- O legal de SCC é que podemos encontrar todas Pontes contidas no Grafo, porém, não temos as informações de Vértices que são Articulações.
- Implementação simples: Tarjan
- Normalmente SCC também é uma parte da solução de um problema, exemplo: Transformar um DAG (Directed Acyclic Graph) contraindo ciclos em um vértice só (Resumir uma SCC à um vértice).

- Alguns problemas no URI.

Strongly Connected Components (Directed Graph)

- Implemente um Tarjan para resumir um DAG.
- Importante: Pág 160 do Livro.

Referências

- Competitive Programming 3.
- http://www.csd.uoc.gr/~hy583/papers/ch3_4.pdf
- https://courses.csail.mit.edu/6.006/fall11/rec/rec14.pdf
- http://cs.stackexchange.com/questions/11116/difference-between-cross-edges-and-forward-edges-in-a-dft