Grafos

Detecção de ciclos

Prof. Edson Alves

Faculdade UnB Gama

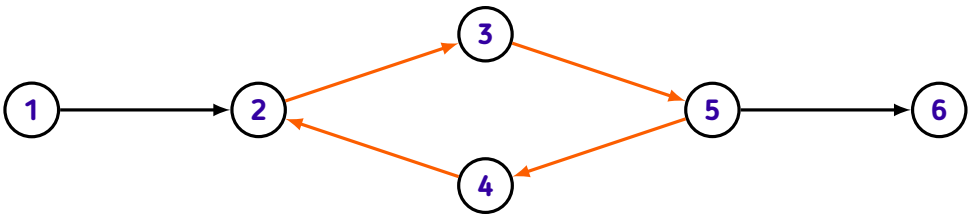
Ciclos

Ciclos

Seja G um grafo não-direcionado. Um **ciclo** é um caminho, com três ou mais arestas distintas, cujos pontos de partida e de chegada são iguais.

Ciclos

Seja G um grafo não-direcionado. Um **ciclo** é um caminho, com três ou mais arestas distintas, cujos pontos de partida e de chegada são iguais.



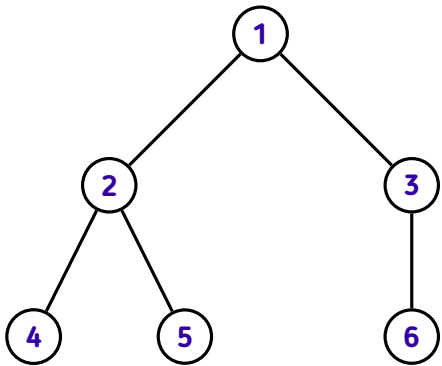
Grafos acíclicos

Grafos acíclicos

Um grafo é dito **acíclico** se não possui ciclos. Árvores são grafos acíclicos.

Grafos acíclicos

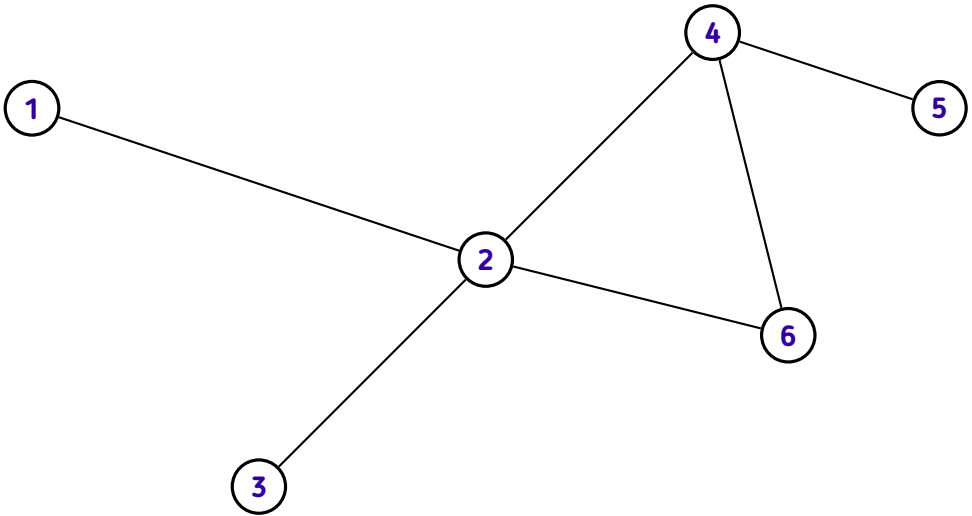
Um grafo é dito acíclico se não possui ciclos. Árvores são grafos acíclicos.

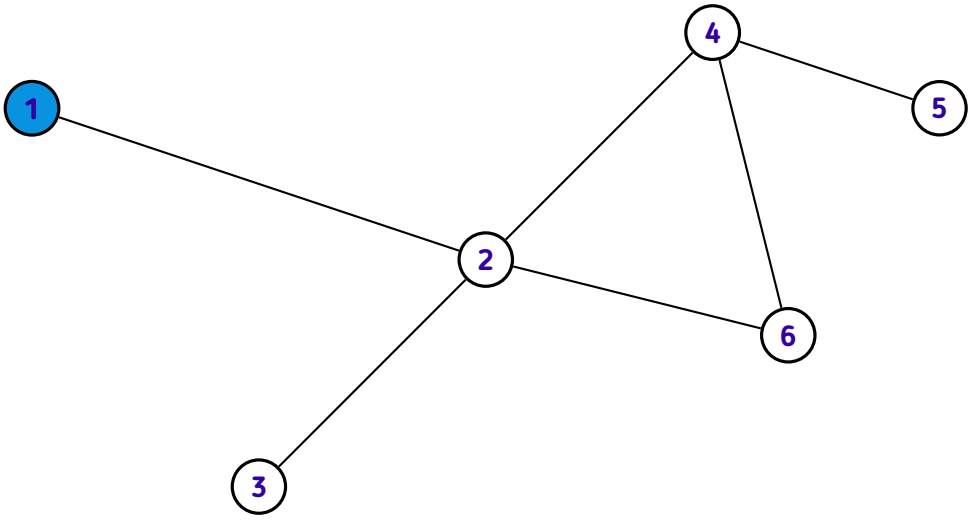


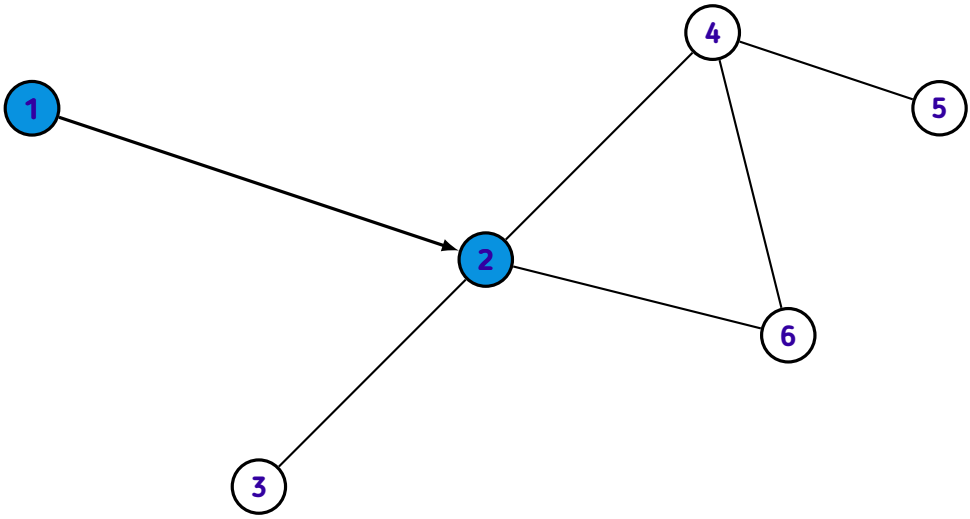
Detecção de ciclos

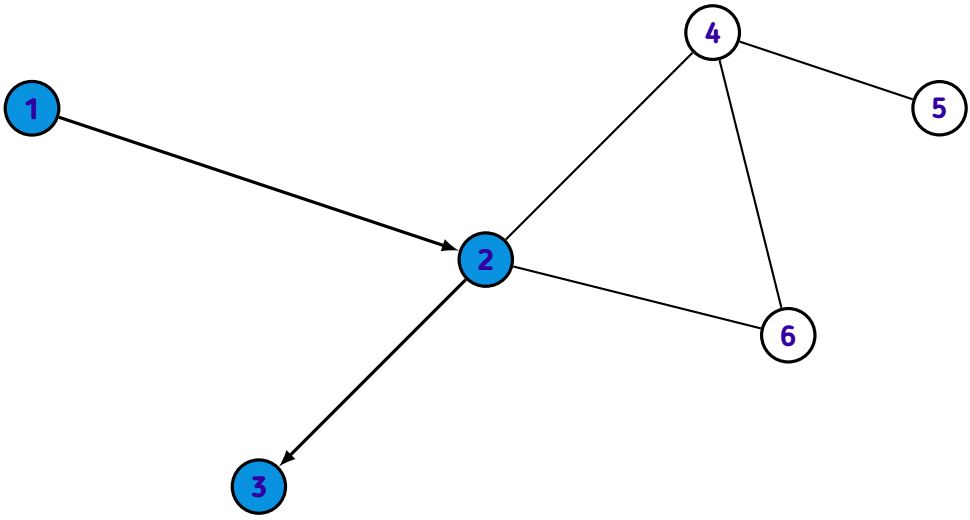
Detecção de ciclos

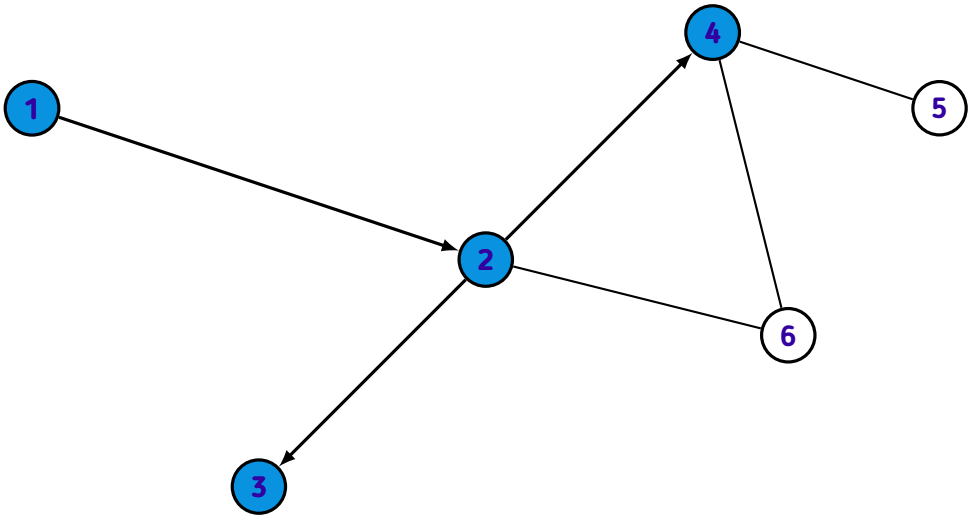
Considere uma travessia por profundidade. Se, durante a travessia, um dos dos vizinhos v de u já foi visitado, e v não é o vértice p que descobriu u na travessia, então existe um ciclo que começa e termina em u e que passa por v.

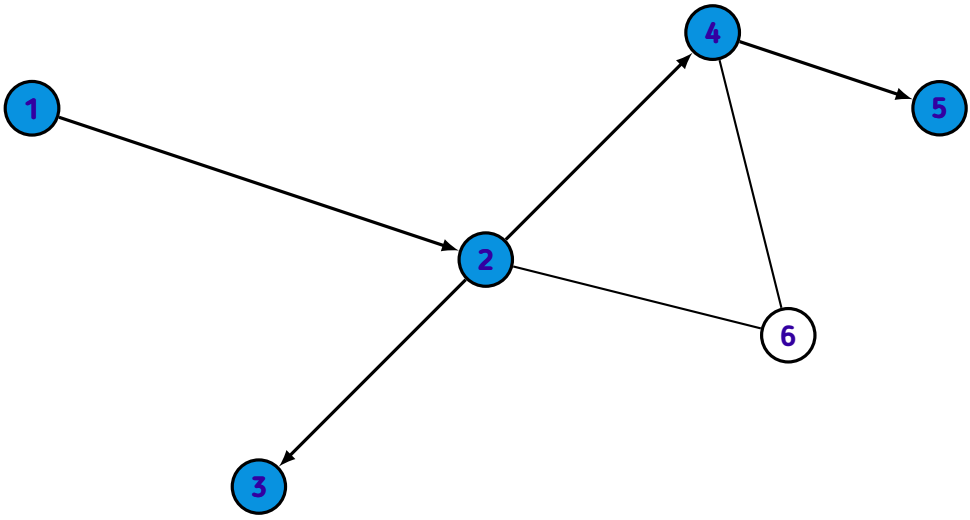

4
1
5
2
6
3

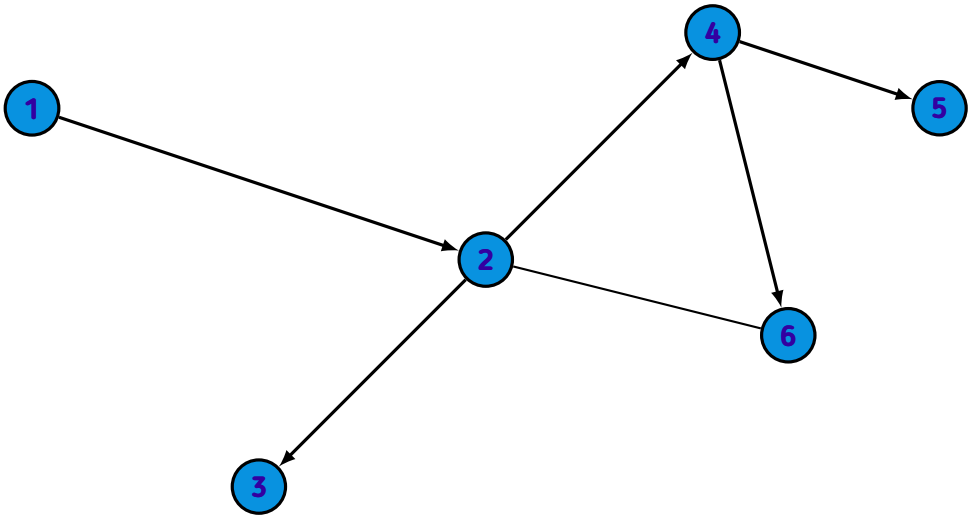

1
2
3
4
5
6

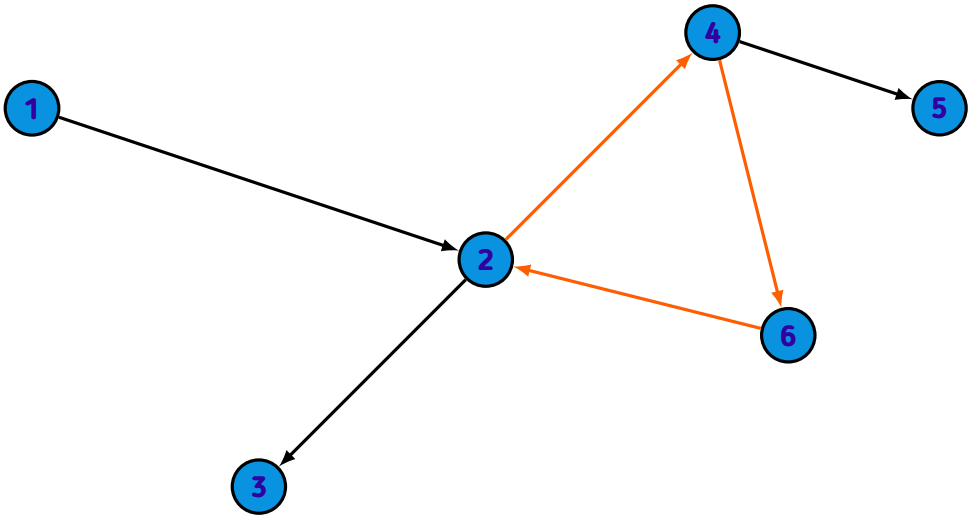

1
2
3
4
5
6


1
2
3
4
5
6


1
2
3
4
5
6


1
2
3
4
5
6


1
2
3
4
5
6


1
2
3
4
5
6

```
bool dfs(int u, int p = -1)
{
    if (visited[u])
        return false;

    visited[u] = true;

    for (auto v : adj[u])
    {
        if (visited[v] and v != p)
            return true;

        if (dfs(v, u))
            return true;
    }

    return false;
}
```

```
bool has_cycle(int N)
{
    visited.reset();

    for (int u = 1; u <= N; ++u)
        if (not visited[u] and dfs(u))
            return true;

    return false;
}
```

Grafos não-direcionados e ciclos

Grafos não-direcionados e ciclos

Se G é um grafo simples não-direcionado com V vértices e E arestas, então G tem ao menos um ciclo se $E \geq V$.

```
void dfs(int u, function<void(int)> process)
{
    if (visited[u])
        return;

    visited[u] = true;

    process(u);

    for (auto v : adj[u])
        dfs(v, process);
}
```

```cpp
bool has_cycle(int N) {
    visited.reset();

    for (int u = 1; u <= N; ++u)
        if (not visited[u])
        {
            vector<int> cs;
            size_t edges = 0;

            dfs(u, [&](int u) {
                cs.push_back(u);

                for (const auto& v : adj[u])
                    edges += (visited[v] ? 0 : 1);
            });

            if (edges >= cs.size()) return true;
        }

    return false;
}
```

Problemas sugeridos

1. **AtCoder Beginner Contest 167 – Problem D: Teleporter**
2. **AtCoder Beginner Contest 174 – Problem C: Repsept**
3. **Educational Codeforces Round 36 – Problem D: Almost Acyclic Graph**
4. **OJ 10116 – Robot Motion**

Referências

1. **HALIM**, **Felix**; **HALIM**, **Steve**. ***Competitive Programming 3*, 2010.**

2. **LAAKSONEN**, **Antti**. ***Competitive Programmer's Handbook*, 2018.**

3. **SKIENA**, **Steven**; **REVILLA**, **Miguel**. ***Programming Challenges*, 2003.**