Algoritmos básicos en árboles.

Funciones de utilidad para cuando un grafo es árbol. La función `set_root` regresa un vector con el padre de cada vértice, (-1 si es la raíz).

La función `height_map` regresa la altura del vértice. Equivalente (pero más rápido) a correr dijkstra.

REQUIERE: Graph

```
#include "Graph.hpp"

#include <cmath>
#include <set>
#include <stack>

using Vertex = Graph::Vertex;

std::vector<Vertex> set_root(const Graph& G, Vertex root)
{
    std::vector<Vertex> parent(G.num_vertices());

    parent[root] = Graph::INVALID_VERTEX;

    std::stack<Vertex> frontier;
    frontier.emplace(root);

    while (!frontier.empty())
    {
        auto p = frontier.top();
        frontier.pop();

        for (auto u : G.neighbors(p))
        {
            if (parent[p] == u)
                continue;
            parent[u] = p;
            frontier.emplace(u);
        }
    }
    return parent;
}

std::vector<int> height_map(const Graph& G, Vertex root)
{
    std::vector<int> level(G.num_vertices(), -1);
```

```
    level[root] = 0;

    std::stack<Vertex> frontier;
    frontier.emplace(root);

    while (!frontier.empty())
    {
        auto p = frontier.top();
        frontier.pop();
        int current_level = level[p];

        for (auto u : G.neighbors(p))
        {
            if (level[u] != -1)
                continue;
            level[u] = current_level + 1;
            frontier.emplace(u);
        }
    }

    return level;
}

using namespace std;

template <class T>
std::ostream& operator<<(std::ostream& os, const std::vector<T>& A)
{
    for (const auto& x : A)
        os << x << ' ';
    return os;
}

int main()
{
    Graph tree(5);
    tree.add_edge(1, 0);
    tree.add_edge(1, 2);
    tree.add_edge(2, 3);
    tree.add_edge(2, 4);

    auto parents = set_root(tree, 1);
    std::cout << "Parents: " << parents << std::endl;
```

```
    auto height = height_map(tree, 1);
    std::cout << "Heights: " << height << std::endl;

    return 0;
}
```

Output:

```
Parents: 1 -1 1 2 2
Heights: 1 0 1 2 2
```