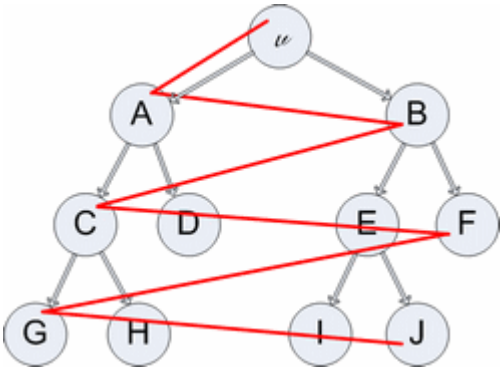
Breitensuche

Erklärung der Breitensuche



- **Grundprinzip:** Ein Algorithmus zur Durchsuchung eines Graphen.
- **Startknoten:** Beginnt bei einem ausgewählten Startknoten.
- **Schichtweise Suche:** Durchsucht den Graph in Ebenen.
- **FIFO-Warteschlange:** Verwendet eine Warteschlange (First-In-First-Out), um Knoten zu verwalten.
- **Entfernung:** Knoten werden nach ihrer Entfernung vom Startknoten besucht.
- **Visitenliste:** Verwendet eine Liste, um bereits besuchte Knoten zu markieren und Duplikate zu vermeiden.
- **Untersucht Kanten:** Betrachtet alle benachbarten Knoten des aktuellen Knotens und fügt sie der Warteschlange hinzu, falls sie noch nicht besucht wurden.
- **Anwendungsbereiche:** Kürzeste Pfadprobleme in ungewichteten Graphen, Netzwerksuche, etc.

C++ Code

```cpp
#include <iostream>
#include <vector>
#include <queue>

using namespace std;

// Graph als Adjazenzliste
vector<vector<int>> graph;

// Breitensuche
void bfs(int start) {
    vector<bool> visited(graph.size(), false);
    vector<int> distance(graph.size(), -1);
    queue<int> q;

    q.push(start);
    visited[start] = true;
    distance[start] = 0;

    while (!q.empty()) {
        int current = q.front();
        q.pop();

        for (int neighbor : graph[current]) {
            if (!visited[neighbor]) {
                q.push(neighbor);
                visited[neighbor] = true;
                distance[neighbor] = distance[current] + 1;
            }
        }
```

Von: Pascal Hammer, Rene Marquardt, Jan Römermann, Julian Wenzel, Adrian Wieg

```
    }

    // Ausgabe der Distanzen
    cout << "Kuerzeste Wege vom Startknoten " << start << ":\n";
    for (int i = 0; i < graph.size(); ++i) {
        cout << "Knoten " << i << ": " << distance[i] << "\n";
    }
}

int main() {
    // Beispielgraph (Adjazenzliste)
    graph = {
        {1, 2},      // Knoten 0
        {0, 2, 3},   // Knoten 1
        {0, 1, 4},   // Knoten 2
        {1},         // Knoten 3
        {2, 5},      // Knoten 4
        {4}          // Knoten 5
    };

    int startknoten = 0;// Startknoten (default)
    cout << "Bitte Nummer des Startknoten eingeben:";
    cin  >> startknoten;
    bfs(startknoten);

    return 0;
}
```

Quellen

[1] Krumke, Noltemeier: Graphentheoretische Konzepte und Algorithmen, Kapitel 7, ab S. 147, Springer, 2012

[2] Knebl: Algorithmen und Datenstrukturen, Kapitel 4, ab S. 133, Springer, 2019

[3] https://www.studysmarter.de/schule/informatik/algorithmen-und-datenstrukturen/breitensuche/
(Stand: 30.05.2024)

[4] http://www.burgnetz.de/otg/informatik/graphen/breitensuche.html
(Stand: 30.05.2024)