拓扑排序模板（C++）

以下是两种常用的拓扑排序实现方法：Kahn算法（基于入度）和DFS算法。

方法一：Kahn算法（基于BFS）

```cpp
#include <iostream>
#include <vector>
#include <queue>
using namespace std;

vector<int> topologicalSort(int n, vector<vector<int>>& graph) {
    vector<int> inDegree(n, 0);
    vector<vector<int>> adj(n); // 邻接表

    // 构建图和入度表
    for (auto& edge : graph) {
        int u = edge[0], v = edge[1];
        adj[u].push_back(v);
        inDegree[v]++;
    }

    queue<int> q;
    // 将所有入度为0的节点加入队列
    for (int i = 0; i < n; i++) {
        if (inDegree[i] == 0) {
            q.push(i);
        }
    }

    vector<int> result;
    while (!q.empty()) {
        int u = q.front();
        q.pop();
        result.push_back(u);

        // 减少所有邻居节点的入度
        for (int v : adj[u]) {
            inDegree[v]--;
            if (inDegree[v] == 0) {
                q.push(v);
            }
        }
    }

    // 如果结果中的节点数不等于总节点数，说明存在环
    if (result.size() != n) {
        return {}; // 返回空数组表示有环，无法拓扑排序
    }
```

```
    return result;
}

int main() {
    // 示例：6个节点，边为[[5,2],[5,0],[4,0],[4,1],[2,3],[3,1]]
    int n = 6;
    vector<vector<int>> graph = {{5,2}, {5,0}, {4,0}, {4,1}, {2,3}, {3,1}};

    vector<int> result = topologicalSort(n, graph);

    if (result.empty()) {
        cout << "图中存在环，无法进行拓扑排序" << endl;
    } else {
        cout << "拓扑排序结果： ";
        for (int node : result) {
            cout << node << " ";
        }
        cout << endl;
    }

    return 0;
}
```

方法二：DFS算法

```
#include <iostream>
#include <vector>
#include <stack>
using namespace std;

bool dfs(int u, vector<vector<int>>& adj, vector<int>& visited, stack<int>&
st) {
    visited[u] = 1; // 标记为正在访问

    for (int v : adj[u]) {
        if (visited[v] == 1) return false; // 发现环
        if (visited[v] == 0 && !dfs(v, adj, visited, st)) return false;
    }

    visited[u] = 2; // 标记为已访问
    st.push(u);
    return true;
}

vector<int> topologicalSortDFS(int n, vector<vector<int>>& graph) {
    vector<vector<int>> adj(n);
    for (auto& edge : graph) {
        int u = edge[0], v = edge[1];
        adj[u].push_back(v);
    }
```

```
    vector<int> visited(n, 0); // 0:未访问, 1:正在访问, 2:已访问
    stack<int> st;

    for (int i = 0; i < n; i++) {
        if (visited[i] == 0) {
            if (!dfs(i, adj, visited, st)) {
                return {}; // 存在环
            }
        }
    }

    vector<int> result;
    while (!st.empty()) {
        result.push_back(st.top());
        st.pop();
    }

    return result;
}

int main() {
    int n = 6;
    vector<vector<int>> graph = {{5,2}, {5,0}, {4,0}, {4,1}, {2,3}, {3,1}};

    vector<int> result = topologicalSortDFS(n, graph);

    if (result.empty()) {
        cout << "图中存在环，无法进行拓扑排序" << endl;
    } else {
        cout << "拓扑排序结果: ";
        for (int node : result) {
            cout << node << " ";
        }
        cout << endl;
    }

    return 0;
}
```

使用说明

1. **输入格式**:
   - `n`: 图中的节点数量
   - `graph`: 边的列表，每条边表示为`[u, v]`，表示从节点`u`指向节点`v`的有向边
2. **输出**:
   - 如果图是有向无环图(DAG)，返回一个拓扑排序序列
   - 如果图中存在环，返回空数组
3. **算法选择**:

- Kahn算法更直观，易于理解，且可以检测环
- DFS算法在某些情况下可能更高效，但实现稍复杂

4. **时间复杂度**：两种算法的时间复杂度均为O(V+E)，其中V是顶点数，E是边数

根据具体问题选择合适的算法实现拓扑排序。