离散数学课程设计
报告书

设计题目：输入图的邻接矩阵，判定该图是否为强连通图

姓名学号：302023315399 郑华展

任课教师：程珍

提交日期：2024.05.21

计算机科学与技术学院

一、实验目的

掌握判定该图是否为强连通图的方法

二、实验内容

输入：一个图的邻接矩阵 输出：图的最大强连通分量，以及该图是否为强连通图

三、相关概念

邻接矩阵

邻接矩阵（Adjacency Matrix）是表示顶点之间相邻关系的矩阵。打个比方说，矩阵 A 中的元素 aij 储存的是 i 点到 j 点的路径信息。

强连通图

强连通图是指在有向图 G 中，如果对于每一对 vi、vj，vi≠vj，从 vi 到 vj 和从 vj 到 vi 都存在路径，则称 G 是强连通图。

强连通分量

有向图中的极大强连通子图称做有向图的强连通分量

四、问题分析

如果一个图是强连通图，那么从图中的任意一个顶点出发，都应该能够到达图中的任意其他顶点，也就是说，任意两个顶点之间都应该存在一条路径。邻接矩阵的幂运算实际上就是将这种路径的数量进行累积，从而得到某个顶点到其他所有顶点的可达性。

当我们进行邻接矩阵的幂运算时，例如将邻接矩阵进行平方、立方等运算，结果矩阵中的元素表示了两个顶点之间存在的长度为 2、3 等的路径数量。因此，如果进行了 2 到 n-1 次幂运算，而结果矩阵中某个元素为 0，意味着对应的两个顶点之间不存在长度为 2 到 n-1 的路径，也就是说它们之间不可达。

将结果矩阵的 1 到 n-1 次幂相加，并对角线元素置为 1，得到的可达矩阵，表示了从一个顶点出发，经过长度为 1 到 n-1 的任意路径，能够到达的所有顶点。如果这个可达矩阵中的所有元素都为 1，那么任意两个顶点之间都存在路径，即图是强连通图。

五、测试数据

我们使用以下程序作为数据生成器，以作为测试之用

```cpp
#include <cstdlib>
#include <ctime>
#include <iostream>
#include <vector>

using namespace std;

int getRandomInt(int min, int max) {
    return min + rand() % (max - min + 1);
}

void printAdjacencyMatrix(const vector<vector<int>>& matrix) {
    for (const auto& row : matrix) {
        for (int val : row) {
            cout << val << " ";
        }
        cout << endl;
    }
}

// 生成强联通图
void generateStronglyConnectedGraph(vector<vector<int>>& adjMatrix, int n) {
    // 保证强连通
    for (int i = 0; i < n; ++i) {
        adjMatrix[i][(i + 1) % n] = 1;
    }

    // 添加随机路径
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            if (i != j && getRandomInt(0, 1) == 1) {
                adjMatrix[i][j] = 1;
            }
        }
    }
}

// 生成非强连通图
void generateNonStronglyConnectedGraph(vector<vector<int>>& adjMatrix, int n) {
    // Split the nodes into two sets to ensure non-strong connectivity
    int splitPoint = n / 2;

    for (int i = 0; i < splitPoint; ++i) {
        for (int j = 0; j < splitPoint; ++j) {
            if (i != j && getRandomInt(0, 1) == 1) {
                adjMatrix[i][j] = 1;
            }
        }
    }
    for (int i = splitPoint; i < n; ++i) {
        for (int j = splitPoint; j < n; ++j) {
            if (i != j && getRandomInt(0, 1) == 1) {
                adjMatrix[i][j] = 1;
```

```
            }
        }
    }

    // 添加随机边
    for (int i = 0; i < splitPoint; ++i) {
        for (int j = splitPoint; j < n; ++j) {
            if (getRandomInt(0, 1) == 1) {
                adjMatrix[i][j] = 1;
            }
        }
    }
}

int main() {
    srand(time(0));

    int n;
    bool isStronglyConnected = true;  // true 生成强连通图，false 生成非强联通图

    cout << "输入点的个数 > ";
    cin >> n;

    vector<vector<int>> adjMatrix(n, vector<int>(n, 0));

    if (isStronglyConnected) {
        generateStronglyConnectedGraph(adjMatrix, n);
    } else {
        generateNonStronglyConnectedGraph(adjMatrix, n);
    }

    printAdjacencyMatrix(adjMatrix);

    return 0;
}
```

六、重要代码分析

求邻接矩阵 2~n-1 次方对应每个点之和

```
int sum, x = 2;
    while (x < dot) {
        cout << "矩阵的" << x << "次方为: " << endl;
        for (int i = 0; i < dot; i++) {
            for (int j = 0; j < dot; j++) {
                sum = 0;
                for (int k = 0; k < dot; k++) {
                    sum +=
                        abs(b[i][k]) *
                        abs(adj[k][j]);  // 有向图中有-1表示反向，所以加上绝对值
```

```
                }
                c[i][j] = sum;
                acc[i][j] += c[i][j];
                cout << c[i][j] << " ";
            }  // j
            cout << endl;
            for (int i = 0; i < dot; i++) {
                for (int j = 0; j < x; j++) {
                    b[i][j] = c[i][j];
                }
            }
        }  // i
        ++x;
    }  // while
```

生成可达矩阵

```
for (int i = 0; i < dot; i++)  // 形成此图的可达矩阵
    {
        for (int j = 0; j < x; j++) {
            if (acc[i][j] || i == j)
                acc[i][j] = 1;
        }
    }
```

七、完整代码

generater.cpp

```
#include <cstdlib>
#include <ctime>
#include <iostream>
#include <vector>

using namespace std;

int getRandomInt(int min, int max) {
    return min + rand() % (max - min + 1);
}

void printAdjacencyMatrix(const vector<vector<int>>& matrix) {
    for (const auto& row : matrix) {
        for (int val : row) {
            cout << val << " ";
        }
        cout << endl;
    }
}
```

```
// 生成强联通图
void generateStronglyConnectedGraph(vector<vector<int>>& adjMatrix, int n) {
    // 保证强连通
    for (int i = 0; i < n; ++i) {
        adjMatrix[i][(i + 1) % n] = 1;
    }

    // 添加随机路径
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            if (i != j && getRandomInt(0, 1) == 1) {
                adjMatrix[i][j] = 1;
            }
        }
    }
}

// 生成非强连通图
void generateNonStronglyConnectedGraph(vector<vector<int>>& adjMatrix, int n) {
    // Split the nodes into two sets to ensure non-strong connectivity
    int splitPoint = n / 2;

    for (int i = 0; i < splitPoint; ++i) {
        for (int j = 0; j < splitPoint; ++j) {
            if (i != j && getRandomInt(0, 1) == 1) {
                adjMatrix[i][j] = 1;
            }
        }
    }
    for (int i = splitPoint; i < n; ++i) {
        for (int j = splitPoint; j < n; ++j) {
            if (i != j && getRandomInt(0, 1) == 1) {
                adjMatrix[i][j] = 1;
            }
        }
    }

    // 添加随机边
    for (int i = 0; i < splitPoint; ++i) {
        for (int j = splitPoint; j < n; ++j) {
            if (getRandomInt(0, 1) == 1) {
                adjMatrix[i][j] = 1;
            }
        }
    }
}

int main() {
    srand(time(0));

    int n;
    bool isStronglyConnected = true;  // true 生成强连通图, false 生成非强联通图
```

```
    cout << "输入点的个数 > ";
    cin >> n;

    vector<vector<int>> adjMatrix(n, vector<int>(n, 0));

    if (isStronglyConnected) {
        generateStronglyConnectedGraph(adjMatrix, n);
    } else {
        generateNonStronglyConnectedGraph(adjMatrix, n);
    }

    printAdjacencyMatrix(adjMatrix);

    return 0;
}
```

resolver.cpp

```
#include <math.h>
#include <iostream>
using namespace std;

int main() {
    cout << "请输入顶点数: ";
    int dot;
    cin >> dot;
    int adj[dot][dot], b[dot][dot], c[dot][dot],
        acc[dot][dot];  // 邻接矩阵与可达矩阵
    cout << "请按次序输入每行数据，形成此图的邻接矩阵 (" << dot << "*" << dot
         << "矩阵" << "): " << endl;
    for (int i = 0; i < dot; i++)  // 邻接矩阵赋值
    {
        for (int j = 0; j < dot; j++) {
            cin >> adj[i][j];
            acc[i][j] = adj[i][j];
            b[i][j] = acc[i][j];
            c[i][j] = acc[i][j];
        }
        getchar();
    }

    int sum, x = 2;  // 求邻接矩阵2~n-1次方对应每个点之和
    while (x < dot) {
        cout << "矩阵的" << x << "次方为: " << endl;
        for (int i = 0; i < dot; i++) {
            for (int j = 0; j < dot; j++) {
                sum = 0;
                for (int k = 0; k < dot; k++) {
                    sum +=
                        abs(b[i][k]) *
                        abs(adj[k][j]);  // 有向图中有-1表示反向，所以加上绝对值
                }
```

```
                c[i][j] = sum;
                acc[i][j] += c[i][j];
                cout << c[i][j] << " ";
            }  // j
            cout << endl;
            for (int i = 0; i < dot; i++) {
                for (int j = 0; j < x; j++) {
                    b[i][j] = c[i][j];
                }
            }
        }  // i
        ++x;
    }  // while

    for (int i = 0; i < dot; i++)  // 形成此图的可达矩阵
    {
        for (int j = 0; j < x; j++) {
            if (acc[i][j] || i == j)
                acc[i][j] = 1;
        }
    }

    cout << "此图的可达矩阵为: " << endl;
    for (int i = 0; i < dot; i++) {
        for (int j = 0; j < dot; j++) {
            cout << acc[i][j] << " ";
        }
        cout << endl;
    }

    for (int i = 0; i < dot; i++)  // 判断是否为强连通图
    {
        for (int j = 0; j < dot; j++) {
            if (!acc[i][j]) {
                cout << "此图是非强连通图! " << endl;
                return 0;
            }
        }
    }
    cout << "此图为强联通图!" << endl;
}
```