Warshall 算法计算传递闭包

2153726 罗宇翔

(经济与管理学院, 信息管理与信息系统)

目录

1 实验目的 **1**

2 实验内容 **1**

3 实验环境 **1**

3.1 Visual Studio Code 1

3.2 g++ 1

4 实验原理和方法 **2**

5 实验代码 **2**

6 实验数据及结果分析 **6**

1 实验目的

利用 $C++$ 代码实现利用 $Warshall$ 算法求关系矩阵的传递闭包。

2 实验内容

给定一个关系矩阵，利用 $Warshall$ 算法求其传递闭包。

3 实验环境

3.1 Visual Studio Code

Version: 1.89.1

Browser:

Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)

AppleWebKit/537.36 (KHTML, like Gecko) Code/1.89.1

Chrome/120.0.6099.291 Electron/28.2.8 Safari/537.36

3.2 g++

Apple clang version 14.0.0 (clang-1400.0.29.202)

Target: x86_64-apple-darwin21.6.0

Thread model: posix

4 实验原理和方法

Algorithm 1 Warshall

Input: R

Output: t(R)

$R_t = R$
for $k \leftarrow 1$ **to** n **do**
  for $i \leftarrow 1$ **to** n **do**
    for $j \leftarrow 1$ **to** n **do**
      $R_t[i,j] \leftarrow R_t[i,j] + R_t[i,k] \cdot R_t[k,j]$
    end for
  end for
end for
return R_t

5 实验代码

```
#include <iostream>
#include <sstream>
#include <vector>

using namespace std;

typedef vector< vector< int > > Matrix;

/**
    * @brief Get the matrix object
    *
    * @return Matrix
    */
Matrix get_matrix();
```

```
/**
    * @brief Print the matrix object
    *
    * @param Matrix
    */
void print_matrix(Matrix &matrix);

/**
    * @brief check the legality of the matrix
    *
    * @param matrix
    * @return true
    * @return false
    */
bool validation(Matrix &matrix);

Matrix transitive_closure(Matrix matrix);

int main() {
    while (true) {
        cout << "any to start, q to quit\n";
        string expression;
        cin >> expression;
        if (expression == "q") {
            cout << "thanks for using\n";
            break;
        }
        Matrix a;
        a = get_matrix();
        if (validation(a)) {  // if valid, print the result
            cout << "transitivity closure:\n";
            Matrix tMatrix = transitive_closure(a);
```

```
            print_matrix(tMatrix);
        } else {  // else raise exception
            cout << "invalid input, please check\n";
        }
    }
    return 0;
}

Matrix get_matrix() {
    Matrix matrix;
    string line;
    cout << "Enter the matrix elements (separate elements with
   spaces, and rows with newlines). Enter EOF to finish:\n";
    while (getline(cin, line))  // split the stream by spaces
   and newlines
    {
        if (line == "EOF")
            break;
        istringstream iss(line);
        vector< int > row;
        int num;
        while (iss >> num)
            row.push_back(num);
        matrix.push_back(row);
    }
    matrix.erase(matrix.begin());
    return matrix;
}

void print_matrix(Matrix &matrix) {
    for (int i = 0; i < matrix.size(); i++) {
        for (int j = 0; j < matrix[i].size(); j++) {
```

```
            cout << matrix[i][j] << " ";
        }
        cout << endl;
    }
}

bool validation(Matrix &matrix) {
    int rows = matrix.size();
    for (int row = 0; row < rows; row++) {
        if (matrix[row].size() != rows) {
            return false;
        }
        for (int col = 0; col < matrix[row].size(); col++) {
            if (matrix[row][col] != 0 && matrix[row][col] != 1)
    {
                printf("invalid input in (%d,%d) %d\n", row,
   col, matrix[row][col]);
                return false;
            }
        }
    }
    return true;
}

Matrix transitive_closure(Matrix matrix) {
    for (int k = 0; k < matrix.size(); k++)
        for (int i = 0; i < matrix.size(); i++)
            for (int j = 0; j < matrix.size(); j++)
                matrix[i][j] = matrix[i][j] || (matrix[i][k] &&
    matrix[k][j]);
    return matrix;
}
```

6 实验数据及结果分析

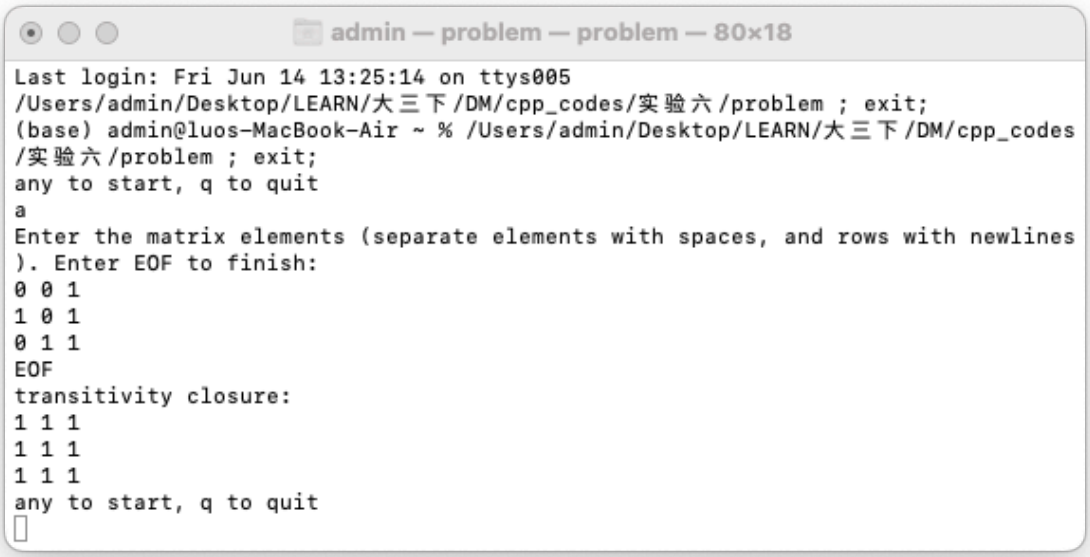

Figure 1：运行结果