システムソフトウェア特論演習

課題０3　説明レポート

劉　立坤

学籍番号：2IE19337P

Contents

[1. Assignment contents 2](#_Toc10994226)

[2. Calculation tasks 2](#_Toc10994227)

[2.1 Task 03-01: Adjacency matrix calculation 2](#_Toc10994228)

[2.2 Task 03-02: Reachable/Unreachable 3](#_Toc10994229)

[3. Programming tasks 3](#_Toc10994230)

[3.1 Program diagram and flow description 3](#_Toc10994231)

[3.2 Modules’ functions 4](#_Toc10994232)

[4. Code details 4](#_Toc10994233)

[4.1 Constants and Variables 4](#_Toc10994234)

[4.1.1 Global Constants and Variables 4](#_Toc10994235)

[4.1.2 Local Variables 4](#_Toc10994236)

[4.2 Program description 5](#_Toc10994237)

[4.2.1 CSV Reader 5](#_Toc10994238)

[4.2.2 Matrix Multiplication 6](#_Toc10994239)

[4.2.3 main 6](#_Toc10994240)

[5. Operating results 8](#_Toc10994241)

# Assignment contents

For the directed graph shown on the right:

1. Find its adjacency matrix.

2. Find all reachable nodes within 3 steps from node 0 using matrix calculation. (The initial node arrives in 0 steps treat as possible nodes)

3. For the same directed graph, proving the following propositions using matrix calculation. (The initial node arrives in 0 steps treat as possible nodes)

1. Every node is reachable starting from node 0.
2. There is unreachable node starting from node 5.

Implementation of reachability judgment unit:

1. Consider the following 10-nodes directed graph:
2. Each node has its own natural number of 0 9 and no duplicates. And is given as ID.
3. G’s adjacency matrix A is a 10X10 square matrix, its component is given as a satisfying of the following equation:
4. For directed graph G’s adjacency matrix A, input the ID of 2 nodes in G, create a program to determine whether it is possible to reach node from node in G.
5. The input method of the adjacency matrix and node ID are free but cannot be hard coded into the program.
6. Note the input method in report.

# Calculation tasks

## Task 03-01: Adjacency matrix calculation

For the given graph on the right side, its adjacency matrix

To start with node 0, the staring vector is:

To find the reachable nodes in 3 steps, we multiply matrix with vector 3 times:

Pulsing the staring vector, the results shows that all the nodes are reachable from node 0 in 3 steps. I.e. from node 0, in 3 steps can reach node 0, 1, 2, 3, 4, 5.

## Task 03-02: Reachable/Unreachable

1. From conclusion above, we know that all nodes are reachable from node 0.
2. To start from node 5, the starting vector is: .

For a 6-nodes graph, the maximum step is 5 steps therefore, the adjacency matrix should be multiplied by 5 times:

From the results above, we’ve notice that from node 5, there’re 3 unreachable nodes: node 0, node 1 and node 2.

# Programming tasks

## Program diagram and flow description

A screenshot of a cell phone

Description automatically generatedSince manual input of a 10X10 matrix (100 elements) prove to be troublesome, this program allows both manual input and input from a csv file and provide user with options.

After receiving adjacency matrix from input, the program will ask for star node and end node, then vectorize the start node to prepare for matrix multiplication.

The program then multiplies the adjacency matrix for 10 times, for each time, we loop through the result vector and for any non-zero value’s position, we set reach vector’s corresponding position’s value to 1, indicating that this node is reached.

After loop is over, we check whether the position of reach vector is 1, if so then this node is reachable, else unreachable.

## Modules’ functions

|  |  |
| --- | --- |
| Module | Function |
| csv\_reader | Accept adjacency matrix’s pointer and ask user for file name as input. Read the csv file and write data into the adjacency matrix. |
| matrix\_multiplication | Take the pointer of 2 input matrix(vector) and a result matrix as well as the size of both input matrix and perform matrix multiplication using for loop and return the result to result matrix(vector). |
| main | Initialize variables and drive the whole program. |

# Code details

## Constants and Variables

### Global Constants and Variables

Global Constants

|  |  |  |
| --- | --- | --- |
| Constant name | Initial value | Function |
| MAX\_FILE\_BUFFER\_SIZE | 500 | Max acceptable character count for a single line in input file. |

Global Variables

|  |  |  |
| --- | --- | --- |
| Variable Name | Type | Function |
| start\_vector | 1X10 int vector | Store the starting vector for each multiplication. |
| result\_vector | 1X10 int vector | Store the result vector for each multiplication. |
| reachable\_vector | 1X10 int vector | Storing whether the corresponding node is reached during the process. |
| adjacency\_matrix | 10X10 int matrix | Store the adjacency matrix given by the graph G. |
| input\_cmd | string | Store the temporary user command. |
| m | int | Starting node |
| n | int | Ending node |

### 4.1.2 Local Variables

|  |  |  |
| --- | --- | --- |
| Variable name | Type | Function |
| \*fp | FILE Pointer | Point to an open file for input and output |
| buf | String | File read buffer |
| file\_name | String | Storing file name |
| return\_code | int | Status code |

## Program description

### CSV Reader

Input: adjacency matrix’s pointer

Return: -1 or -2 or none

Description:

This function askes user for addition input(filename) and read the adjacency matrix from file.

### Matrix Multiplication

Input: input matrix A and B’s pointer, result matrix C’s pointer, A and B’s size

Return: none

Description:

This function just does simple linear algebra work via loop.

### main

Input: none

Return: 0, -1 or -2

Description:

This is the main driver code. This particular part initializes global variables and handle the input from file.

This part of code handles manual input option which directly take adjacency matrix from keyboard.

This part of code takes start and end note and vectorize the input

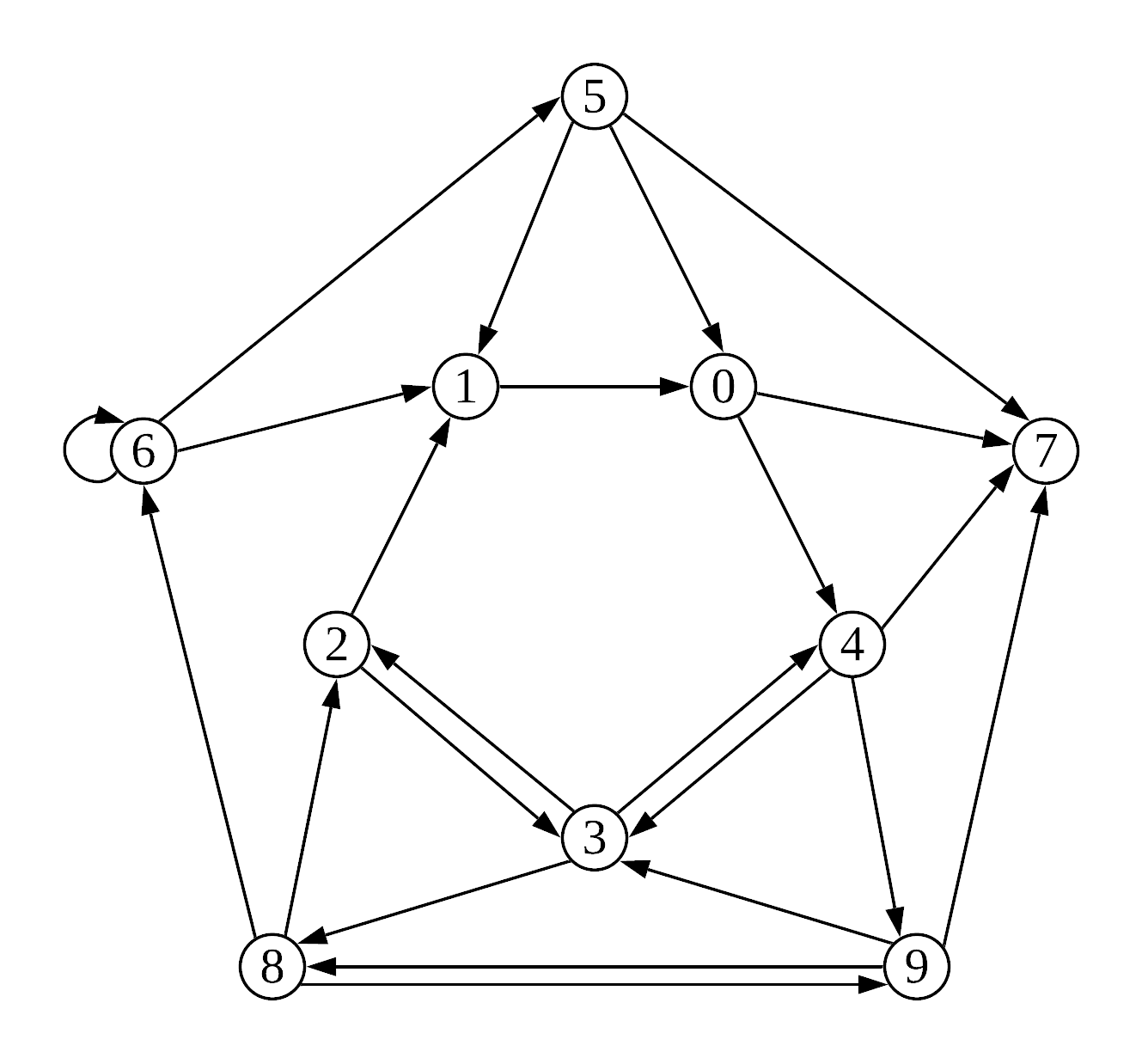
This part of code executes the multiplication process

This part of code prints out the result vector

This part handles whether the node is reachable or not.

# Operating results

In this phase, we’ve implemented a 10-nodes graph as following:



The corresponding adjacency matrix is:

Test 01:

A screenshot of a cell phone

Description automatically generated

In this test scenario, we’ve tested from node 2 to node 8, the result in fact, shows that from node 2 can not only reach node 8 but also every node in the graph.

Test 02:

A screen shot of a computer

Description automatically generated

From the graph we can see that node 7 is isolated and can only be reached but cannot reach other nodes, so we’ve tested from node 7 to node 2. The result shows that node 7 can only reach node 7 which is step 0 and therefore the conclusion is right.

Test 03:

A screen shot of a computer

Description automatically generated

In this test, we’ve tested from the isolated node 7 to itself, and the “Destination reachable” is the right conclusion.