システムソフトウェア特論演習

課題０３　説明レポート

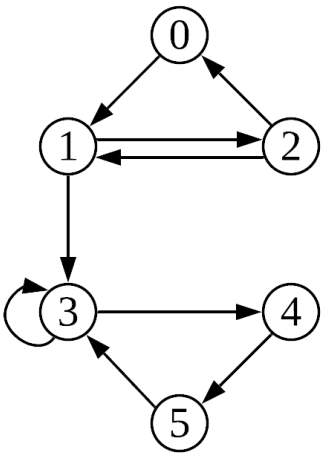
劉　立坤

学籍番号：2IE19337P

Contents

1. Assignment contents

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)

   a. Every node is reachable starting from node 0.
   b. There is unreachable node starting from node 5.

Implementation of reachability judgment unit:

1. Consider the following 10-nodes directed graph:
   a. Each node has its own natural number of 0 9 and no duplicates. And is given as ID.
   b. G's adjacency matrix A is a 10X10 square matrix, its (i,j) component a_{ij} is given as a satisfying of the following equation:

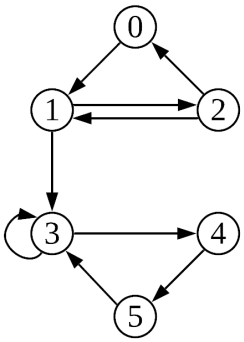
$$a_{ij} = \begin{cases} 1\ (where\ node\ i\ can\ reach\ node\ j\ directly) \\ 0\ (where\ node\ i\ cannot\ reach\ node\ j\ directly) \end{cases}$$

2. For directed graph G's adjacency matrix A, input the ID m, n of 2 nodes in G, create a program to determine whether it is possible to reach node n from node m in G.
   a. The input method of the adjacency matrix and node ID are free but cannot be hard coded into the program.
   b. Note the input method in report.

2. Calculation tasks

2.1 Task 03-01: Adjacency matrix calculation

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



$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

To start with node 0, the staring vector is: $x_0 = (1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0)$

To find the reachable nodes in 3 steps, we multiply matrix A with vector x_0 3 times:

$$x_1 = x_0 A = (0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0)$$

$$x_2 = x_1 A = (0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0)$$

$$x_3 = x_2 A = (1 \quad 1 \quad 0 \quad 1 \quad 1 \quad 1)$$

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.

2.2 Task 03-02: Reachable/Unreachable

a. From conclusion above, we know that all nodes are reachable from node 0.

b. To start from node 5, the starting vector is: $x_0 = (0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1)$.
For a 6-nodes graph, the maximum step is 5 steps therefore, the adjacency matrix should be multiplied by 5 times:

$$x_1 = x_0 A = (0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0)$$

$$x_2 = x_1 A = (0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0)$$

$$x_3 = x_2 A = (0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1)$$

$$x_4 = x_3 A = (0 \quad 0 \quad 0 \quad 2 \quad 1 \quad 1)$$

$$x_5 = x_4 A = (0 \quad 0 \quad 0 \quad 3 \quad 2 \quad 1)$$

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

3. Programming tasks

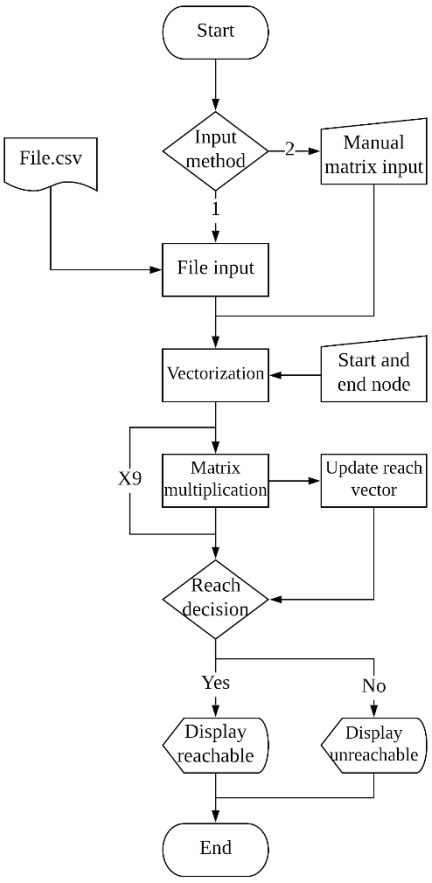
3.1 Program diagram and flow description

Since 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 nth position of reach vector is 1, if so then this node is reachable, else unreachable.



3.2 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.

4. Code details

4.1 Constants and Variables

4.1.1 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

4.2 Program description

4.2.1 CSV Reader

Input: adjacency matrix's pointer

Return: -1 or -2 or none

```
int csv_reader(int *adjacency_matrix) {
   FILE *fp;
   char file_name[50];
   char buf[MAX_FILE_BUFFER_SIZE] = "";
   int j = 0;
   while (1) {
      printf("Pleas specify file name [test_ma-
trix.csv]: \n");
      fgets(&file_name[0], sizeof(file_name), stdin);
      if (file_name[0] == '\n') {
         strcpy(file_name, "test_matrix.csv");}
      else if (file_name[0] == 'e')
         return -1;
      strtok(file_name, "\n");
      if ((fp = fopen(file_name, "r")) == NULL)
{     //File validation check
         printf("%s", file_name);
         printf("File could not be opened. Retry input
or exit by typing 'e'\n");}
      else
         break;
   }
   if (buf == NULL) {                          //Memory
check
      printf("No memory available.\n");
      return -2;
   }
   while (fgets(buf, 255, fp) != NULL) {
      int str_length = strlen(buf);            //Get cur-
rent line's length
      for (int i = 0; i < str_length; i++) {
         if ((buf[i] >= 48) && (buf[i] <= 57)) {
            *(adjacency_matrix + j) = (buf[i] - 48);
            j++;
         }}}}//CSV Data Reader
```

Description:

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

4.2.2 Matrix Multiplication

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

Return: none

```
void matrix_multiplication(int *A, int *B, int *C,
int A_row, int A_column, int B_row, int B_col-
umn) {
   if (A_column != B_row)
      printf("Error on matrix dimension\n");
   else {
      for (int i = 0; i < A_row; i++) {
         for (int j = 0; j < B_column; j++) {
            *(C + j + i * B_column) = 0;
            for (int k = 0; k < B_row; k++) {
               *(C + j + i * B_column) += *(A + i *
A_column + k) * *(B + j + k * B_column);
            }
         }
      }
   }
}
```

Description:

This function just does simple linear algebra work via loop.

4.2.3 main

Input: none

Return: 0, -1 or -2

```
int main() {
   int start_vector[1][10] = {0};
   int result_vector[1][10];
   int reachable_vector[10] = {0};
   int adjacency_matrix[10][10];
   char input_cmd[20];
   int m, n;
   printf("Please select Adjacency matrix input method (1 for csv
file input, 2 for manual input)[1]: \n");
   fgets(&input_cmd[0], sizeof(input_cmd), stdin);
   if (input_cmd[0] == '\n' || input_cmd[0] == '1') {
      int return_code = csv_reader(&adjacency_matrix[0]);
      if ( return_code == -1){
         printf("Program exit on user command.");
         exit(-1); }
      else if (return_code == -2) {
         printf("Exit on lack of memory");
         exit(-2); } }//File adjacency matrix input
```

Description:

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

```
else{
   printf("Please input Adjacency matrix:\n");
   for (int i = 0; i < 10; i++) {
      scanf("%d %d %d %d %d %d %d %d %d %d", &adja-
cency_matrix[i][0], &adjacency_matrix[i][1], &adjacency_ma-
trix[i][2],
           &adjacency_matrix[i][3], &adjacency_matrix[i][4],
&adjacency_matrix[i][5], &adjacency_matrix[i][6],
           &adjacency_matrix[i][7],
           &adjacency_matrix[i][8], &adjacency_matrix[i][9]);
   }
}//Manual adjacency matrix input
printf("Please input start and end node ID:\n");
scanf("%d %d", &m, &n);
start_vector[0][m] = 1;    //Initialize the start vector
reachable_vector[m] = 1; //The 0th step on the node itself

for (int i = 0; i < 10; i++) {
   matrix_multiplication(&start_vector[0][0], &adjacency_ma-
trix[0][0], &result_vector[0][0], 1, 10, 10, 10);
   memcpy(&start_vector[0][0], &result_vector[0][0], sizeof
start_vector);
   for (int j = 0; j < 10; j++){
      if (start_vector[0][j]>0)
         reachable_vector[j]=1;
   }
}
   printf("Reachable Vector is:\n");
   for (int i = 0; i < 10; i++)
      printf("%d ", reachable_vector[i]);
   printf("\n");

   if (reachable_vector[n] > 0)
      printf("Destination reachable\n");
   else
      printf("Destination unreachable\n");
   return 0;
}
```

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

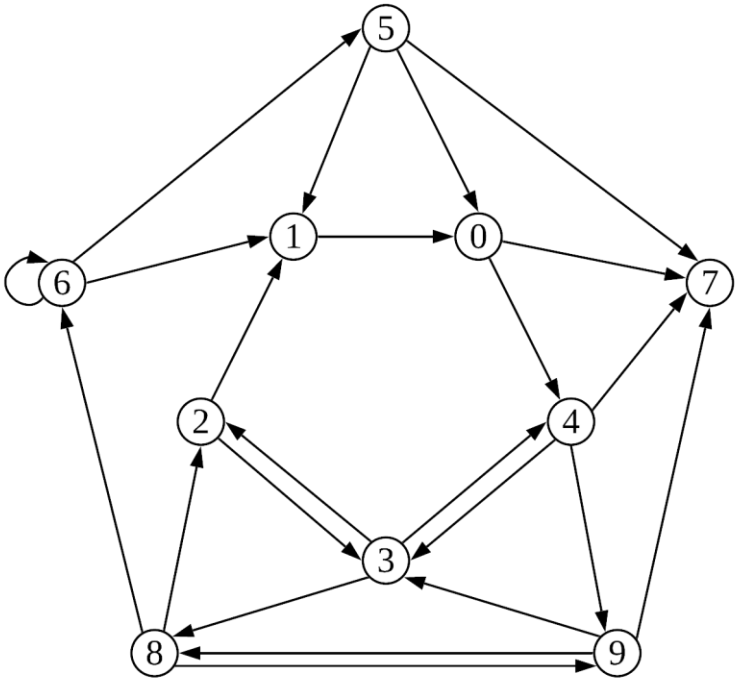
This part of code executes the multiplication process

This part of code prints out the result vector

This part handles whether the node is reachable or not.

5. Operating results

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



The corresponding adjacency matrix is:

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

Test 01:

```
C:¥Users¥det_c¥Desktop¥Projects¥School-Works¥System_Software¥assignment3¥cmake-build-debug>assignment3.exe
Please select Adjacency matrix input method (1 for csv file input, 2 for manual input)[1]:

Pleas specify file name [test_matrix.csv]:

Please input start and end node ID:
2 8
Reachable Vector is:
1 1 1 1 1 1 1 1 1 1
Destination reachable
```

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:

```
C:¥Users¥det_c¥Desktop¥Projects¥School-Works¥System_Software¥assignment3¥cmake-build-debug>assignment3.exe
Please select Adjacency matrix input method (1 for csv file input, 2 for manual input)[1]:

Pleas specify file name [test_matrix.csv]:

Please input start and end node ID:
7 2
Reachable Vector is:
0 0 0 0 0 0 0 1 0 0
Destination unreachable
```

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:

```
C:¥Users¥det_c¥Desktop¥Projects¥School-Works¥System_Software¥assignment3¥cmake-build-debug>assignment3.exe
Please select Adjacency matrix input method (1 for csv file input, 2 for manual input)[1]:

Pleas specify file name [test_matrix.csv]:

Please input start and end node ID:
7 7
Reachable Vector is:
0 0 0 0 0 0 0 1 0 0
Destination reachable
```

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