Rajshahi University of Engineering & Technology

CSE 2102: Sessional Based on CSE 2101

Lab Report 05

Dated: 17.04.18

Submitted to

Rizoan Toufiq

Assistant Professor

Dept. of Computer Science & Engineering

&

Instructor, CSE 2102

Submitted by

Fuad Al Abir

Roll: 1603021

Section: A

Dept. of Computer Science & Engineering

**Experiment No. 3**

**Name of the Experiment: Algorithms, Number Theory and Cryptography**

**1. EXPERIMENT [34]**

Given an m × k matrix A and a k × n matrix B, find AB.

**SOLUTION:**

#include <iostream>

using namespace std;

int main() {

int m, k, n, i, j, q;

cout << "Enter m, k, n: ";

cin >> m >> k >> n;

int matrix\_one[m][k], matrix\_two[k][n], multiMatrix[m][n];

cout << "Matrix\_One:\n";

for(i = 0; i < m; i++) {

for(j = 0; j < k; j++) {

cin >> matrix\_one[i][j];

}

}

cout << "Matrix\_Two:\n";

for(i = 0; i < k; i++) {

for(j = 0; j < n; j++) {

cin >> matrix\_two[i][j];

}

}

// calculation goes here

for(i = 0; i < m; i++) {

for(j = 0; j < n; j++) {

multiMatrix[i][j] = 0;

for(q = 0; q < k; q++) {

multiMatrix[i][j] = multiMatrix[i][j] + matrix\_one[i][q] \* matrix\_two[q][j];

}

}

}

// output

cout << "Matrix Product:\n";

for(i = 0; i < m; i++) {

for(j = 0; j < n; j++) {

cout << multiMatrix[i][j] << " ";

}

cout << endl;

}

}

OUTPUT:

Enter m, k, n: 3 3 3

Matrix\_One:

1 2 3

4 5 6

7 8 9

Matrix\_Two:

1 2 2

1 3 1

2 2 1

Matrix Product:

9 14 7

21 35 19

33 56 31

**2. EXPERIMENT [35]**

Given a square matrix A and a positive integer n, find An.

**SOLUTION:**

#include <iostream>

using namespace std;

int main() {

int m, n, i, j, l, q;

cout << "Enter order of square matrix: ";

cin >> m;

cout << "Enter n: ";

cin >> n;

int matrix[m][m], multiMatrix[m][m] = {0};

for(i = 0; i < m; i++) {

for(j = 0; j < m; j++) {

multiMatrix[i][j] = 0;

}

}

cout << "Enter the Matrix:\n";

for(i = 0; i < m; i++) {

for(j = 0; j < m; j++) {

cin >> matrix[i][j];

}

}

// calculation goes here

for(l = 0; l < n; l++) {

for(i = 0; i < m; i++) {

for(j = 0; j < m; j++) {

for(q = 0; q < m; q++) {

multiMatrix[i][j] = multiMatrix[i][j] + matrix[i][q] \* matrix[q][j];

}

}

}

}

// output

cout << "Matrix Product:\n";

for(i = 0; i < m; i++) {

for(j = 0; j < m; j++) {

cout << multiMatrix[i][j] << " ";

}

cout << endl;

}

}

OUTPUT:

Enter order of square matrix: 3

Enter n: 3

Enter the Matrix:

1 2 3

1 2 1

3 2 1

Matrix Product:

36 36 24

18 24 18

24 36 36

**3. EXPERIMENT [36]**

Given a square matrix, determine whether it is symmetric.

**SOLUTION:**

#include <iostream>

using namespace std;

int main() {

int n, flag = 1;

cout << "Enter the order of the matrix: ";

cin >> n;

int matrix[n][n];

cout << "Enter the matrix\n";

for(int i = 0; i < n; i++) {

for(int j = 0; j < n; j++) {

cin >> matrix[i][j];

}

}

for(int i = 0; i < n; i++) {

for(int j = 0; j < n; j++) {

if(i == j) continue;

if(matrix[i][j] != matrix[j][i]) {

flag = 0;

break;

}

}

}

if(1 == flag) {

cout << "SYMMETRIC MATRIX." << endl;

} else {

cout << "Non-SYMMATRIC MATRIX." << endl;

}

}

OUTPUT:

Enter the order of the matrix: 3

Enter the matrix

1 2 3

2 4 5

3 5 1

SYMMETRIC MATRIX.

Enter the order of the matrix: 3

Enter the matrix

1 2 3

1 2 3

1 2 3

Non-SYMMATRIC MATRIX.

**4. EXPERIMENT [37]**

Given two m × n Boolean matrices,find their meet and join.

**SOLUTION:**

#include <iostream>

using namespace std;

int main() {

int m, n, i, j;

cout << "Enter row: ";

cin >> m;

cout << "Enter column: ";

cin >> n;

int matrix\_one[m][n], matrix\_two[m][n], matrix\_join[m][n], matrix\_meet[m][n];

// input matrix\_one

cout << "Enter matrix\_one:\n";

for(i = 0; i < m; i++) {

for(j = 0; j < n; j++) {

cin >> matrix\_one[i][j];

}

}

cout << "Enter matrix\_two:\n";

for(i = 0; i < m; i++) {

for(j = 0; j < n; j++) {

cin >> matrix\_two[i][j];

}

}

cout << endl;

for(i = 0; i < m; i++) {

for(j = 0; j < n; j++) {

if(1 == matrix\_one[i][j] || 1 == matrix\_two[i][j]) {

matrix\_join[i][j] = 1;

} else matrix\_join[i][j] = 0;

}

}

for(i = 0; i < m; i++) {

for(j = 0; j < n; j++) {

if(1 == matrix\_one[i][j] && 1 == matrix\_two[i][j]) {

matrix\_meet[i][j] = 1;

} else matrix\_meet[i][j] = 0;

}

}

cout << "Join of matrix\_one and matrix\_two:\n";

for(i = 0; i < m; i++) {

for(j = 0; j < n; j++) {

cout << matrix\_join[i][j] << " ";

}

cout << endl;

}

cout << "Meet of matrix\_one and matrix\_two:\n";

for(i = 0; i < m; i++) {

for(j = 0; j < n; j++) {

cout << matrix\_meet[i][j] << " ";

}

cout << endl;

}

}

OUTPUT:

Enter row: 2

Enter column: 3

Enter matrix\_one:

1 0 0

0 0 1

Enter matrix\_two:

1 1 0

1 1 1

Join of matrix\_one and matrix\_two:

1 1 0

1 1 1

Meet of matrix\_one and matrix\_two:

1 0 0

0 0 1

**5. EXPERIMENT [38]**

Given an m × k Boolean matrix A anda k × n Boolean matrix B,find the

Boolean product of A and B.

**SOLUTION:**

#include <iostream>

using namespace std;

int main() {

// matrix order variable;

int m, k, n;

// loop variable

int i, j, q;

cout << "Enter m, k, n: ";

cin >> m >> k >> n;

// matrix definition

int matrix\_one[m][k], matrix\_two[k][n], bool\_matrix\_product[m][n];

cout << "Enter matrix\_one:\n";

for(i = 0; i < m; i++) {

for(j = 0; j < k; j++) {

cin >> matrix\_one[i][j];

}

}

cout << "Enter matrix\_two:\n";

for(i = 0; i < k; i++) {

for(j = 0; j < n; j++) {

cin >> matrix\_two[i][j];

}

}

// calculations goes here

for(i = 0; i < m; i++) {

for(j = 0; j < n; j++) {

bool\_matrix\_product[i][j] = 0;

for(q = 0; q < k; q++) {

bool\_matrix\_product[i][j] = bool\_matrix\_product[i][j] || (matrix\_one[i][q] && matrix\_two[q][j]);

}

}

}

cout << "\nBoolean product:\n";

for(i = 0; i < m; i++) {

for(j = 0; j < n; j++) {

cout << bool\_matrix\_product[i][j] << " ";

}

cout << endl;

}

}

OUTPUT:

Enter m, k, n: 3 3 3

Enter matrix\_one:

1 0 0

0 0 1

0 0 0

Enter matrix\_two:

1 1 0

0 1 1

1 1 1

Boolean product:

1 1 0

1 1 1

0 0 0

**6. EXPERIMENT [39]**

Given a square Boolean matrix A and a positive integer n, find A[n].

**SOLUTION:**

#include <iostream>

using namespace std;

int main() {

int m, n, i, j, l, q;

cout << "Enter order of square matrix: ";

cin >> m;

cout << "Enter n: ";

cin >> n;

int matrix[m][m], multiMatrix[m][m] = {0};

for(i = 0; i < m; i++) {

for(j = 0; j < m; j++) {

multiMatrix[i][j] = 0;

}

}

cout << "Enter the Matrix:\n";

for(i = 0; i < m; i++) {

for(j = 0; j < m; j++) {

cin >> matrix[i][j];

}

}

// calculation goes here

for(l = 0; l < n; l++) {

for(i = 0; i < m; i++) {

for(j = 0; j < m; j++) {

for(q = 0; q < m; q++) {

multiMatrix[i][j] = multiMatrix[i][j] || (matrix[i][q] && matrix[q][j]);

}

}

}

}

// output

cout << "Matrix Product:\n";

for(i = 0; i < m; i++) {

for(j = 0; j < m; j++) {

cout << multiMatrix[i][j] << " ";

}

cout << endl;

}

}

OUTPUT:

Enter order of square matrix: 4

Enter n: 2

Enter the Matrix:

1 1 0 0

0 1 0 1

1 0 1 0

1 0 0 1

Matrix Product:

1 1 0 1

1 1 0 1

1 1 1 0

1 1 0 1