University of Tehran

Faculty of Algorithms and Computations

Course Prof:

Dr Moeini

Solving:

Mustafa Mohammadi Gharasuie

Std-ID: 810897030

TA exercises topic:

Verifying Randomized matrices multiplication

Date:

28 March. 2018

As defined in the teacher assistance classroom for randomized algorithms, students should write a computer program to verifying randomized matrices multiplication. This algorithm is implemented by Matlab program by myself on 28 March 2018.

This program consists seven matrices as below table with definitions

Variable name	Variable type	diemention	Description
mt_A	Matrix	n*n	The first matrix A
mt_B	Matrix	n*n	The second matrix B
mt_correct_mult	Matrix	n*n	Multiplication of matrices A and B
mt_Incorrect_mult	Matrix	n*n	Arbitrary n*n matrix
vt_rnd	Vector	n*1	Random vector n*1 with {0,1} elements
mt_cr_pr	Matrix	n*n	Calculate *FREIVALDS* formula for mt_A, mt_B, and mt_correct_mult
mt_Incr_pr	Matrix	n*n	Calculate *FREIVALDS* formula for mt_A, mt_B, and mt_Incorrect_mult

mt_correct_mult is result of correct multiplication of two matrices mt_A, and mt_B. mt_incorrect_mult is either a random values matrix, or multiplication of our two matrices with little different in some elements.

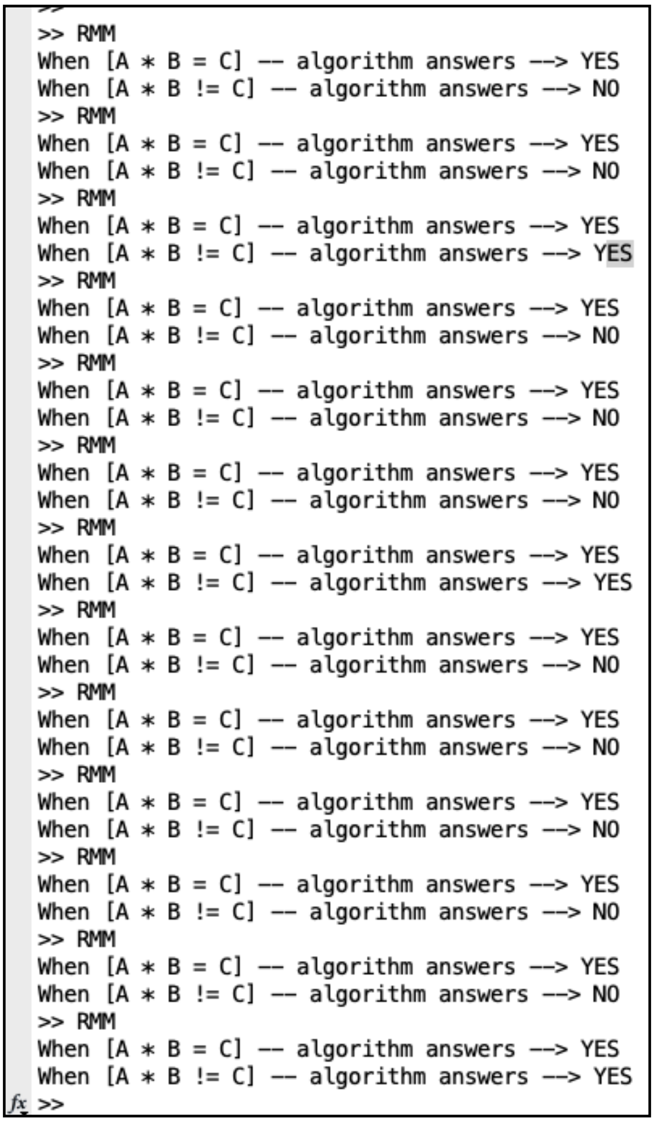
This simple code is shown as follow:

```
% input matrices
mt_A = [2, 7 ; 6, 4];

mt_B = [1, 3 ; 1, 2];

% multiply two intended matrices for our project
mt_correct_mult = mt_A * mt_B;


% arbitrary matrix
mt_Incorrect_mult = [90, 20 ; 10, 15];


% define N*1 random {0,1} vector vt_rnd

vt_rnd = randi(2, [2,1])-1;

% implement FREIVALDS Formula on both situation(correct and incorrect
% matrix)
mt_cr_pr = (mt_A * (mt_B * vt_rnd)) - (mt_correct_mult * vt_rnd);
mt_inCr_pr = (mt_A * (mt_B * vt_rnd)) - (mt_Incorrect_mult * vt_rnd);

% check the program computation
if mt_cr_pr == 0
  disp('When [A * B = C] -- algorithm answers --> YES');
else
    disp('When [A * B = C] -- algorithm answers --> NO');
end;

if mt_inCr_pr == 0
  disp('When [A * B != C] -- algorithm answers --> YES');
else
    disp('When [A * B != C] -- algorithm answers --> NO');
end;
```

My written algorithms outputs two kind of situation for correct and wrong multiplication of input mt_A, and mt_B matrices. I calculate that the probability of answering “Yes” is less than or equal to one half when $A * B \notin C$. This is known as one-sided error.

By iterating the algorithm k times and returning "Yes" only if all iterations yield "Yes", a runtime of $O(kn^2)$ and error probability of less than $\frac{1}{2^k}$ is achieved.

The image below show the answers in some independent runs:



```
>> RMM
When [A * B = C] -- algorithm answers --> YES
When [A * B != C] -- algorithm answers --> NO
>> RMM
When [A * B = C] -- algorithm answers --> YES
When [A * B != C] -- algorithm answers --> NO
>> RMM
When [A * B = C] -- algorithm answers --> YES
When [A * B != C] -- algorithm answers --> YES
>> RMM
When [A * B = C] -- algorithm answers --> YES
When [A * B != C] -- algorithm answers --> NO
>> RMM
When [A * B = C] -- algorithm answers --> YES
When [A * B != C] -- algorithm answers --> NO
>> RMM
When [A * B = C] -- algorithm answers --> YES
When [A * B != C] -- algorithm answers --> NO
>> RMM
When [A * B = C] -- algorithm answers --> YES
When [A * B != C] -- algorithm answers --> YES
>> RMM
When [A * B = C] -- algorithm answers --> YES
When [A * B != C] -- algorithm answers --> NO
>> RMM
When [A * B = C] -- algorithm answers --> YES
When [A * B != C] -- algorithm answers --> NO
>> RMM
When [A * B = C] -- algorithm answers --> YES
When [A * B != C] -- algorithm answers --> NO
>> RMM
When [A * B = C] -- algorithm answers --> YES
When [A * B != C] -- algorithm answers --> NO
>> RMM
When [A * B = C] -- algorithm answers --> YES
When [A * B != C] -- algorithm answers --> NO
>> RMM
When [A * B = C] -- algorithm answers --> YES
When [A * B != C] -- algorithm answers --> YES
fx >>
```

good luck!