

COP 4531: Analysis of Algorithms

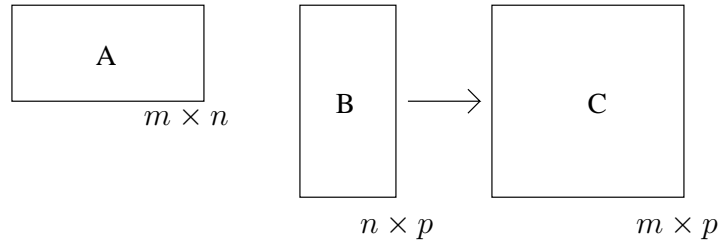
Topic: PROGRAMMING ASSIGNMENT
Sep 30, 2004

LECTURER: *Piyush Kumar*

SCRIBED BY: *Jesse Birch*

If you find any errors, please let me know at piyush@acm.org.

You are given two matrices $A_{m \times n}$, $B_{n \times p}$. You need to multiply $A_{m \times n}$, $B_{n \times p}$ to create $C_{m \times p}$.



You will need to implement two different algorithms for matrix multiplication. The first implementation will be a simple (for loop based) algorithm, whereas the second implementation will be a recursive one.

Problem 1 Calculate

$$C_{ij} = \sum_{k=1}^n a_{ik}b_{kj}$$

using nested for-loops. Note that this algorithm takes $O(n^3)$ time to run (as does the algorithm in the next problem).

Problem 2 Implement a divide-and-conquer based matrix multiplication algorithm (similar to Strassen's algorithm). In this implementation you divide both A and B in two block matrices each. This is done recursively.

We will represent two dimensional matrices by an array (actually a pointer to the first element). Assume row major order (That is the way C stores its matrices in the memory):

```
matmul( REAL *A, REAL *B, int m, int n, int p)
{
  Aij = (i - 1) * n + j;
  Bij = (i - 1) * p + j;
}
```

For the recursive matrix multiplication algorithm, let us assume that

$$\delta = \max\{m, n, p\}$$

There are three cases:

Then the recursive matrix multiplication algorithm cuts the input matrix with the longest/largest side (δ) into two parts and recursively multiplies them according to the given figures. Note that in all of these cases we can divide the bigger matrix (matrix with the largest side) into two pieces such that we can break the problem into subproblems on which we can recurse. If the recursion reaches a stage when $\delta < \text{constant}$ (for instance, 10), we use the function we implemented in (1).

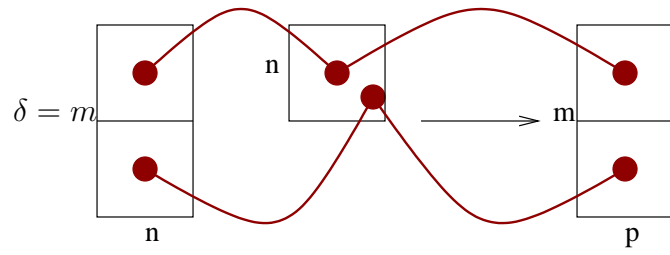


Figure 1: **Case 1**

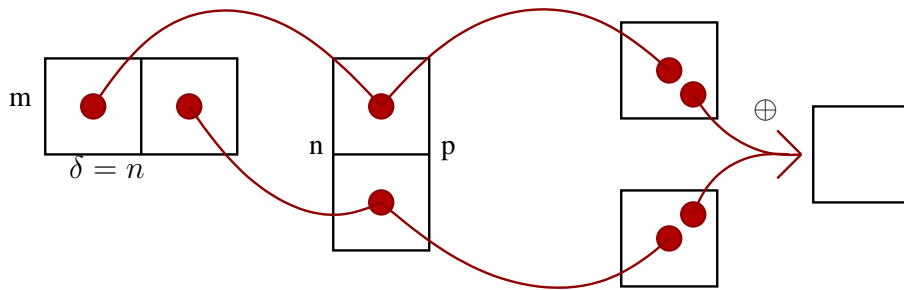


Figure 2: **Case 2**

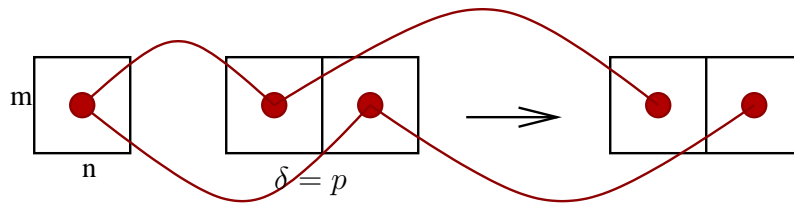


Figure 3: **Case 3**

1 Extra Credit

Write another program that changes the constant in (2) and measures the speed of matrix multiplication. Find the constant that makes your program fastest. Automate this process such that it fine-tunes (calculates its constant) on any computer before a user can use your matrix-multiplication routine.

2 Other Trivia

The due date for the assignment is Oct 30. The only file that you need to submit is `matmul.c` if you are not doing the extra credit for the assignment. The output of the `matmul.c` program should remain intact, except of course the timings will change. Each one of you has to individually do the programming assignment. I will use automatic code checkers to find out codes that resemble (so if you even think of copying someone else's code off the web or from a fellow student, come and talk to me first).