Data Structure and Algorithm

Homework #1

Due: 5pm, Tuesday, March 13, 2012

TA email: dsa1@csie.ntu.edu.tw

=== Homework submission instructions ===

- For Problem 1, submit your source code, a shell script to compile the source, and a brief documentation to the SVN server (katrina.csie.ntu.edu.tw). You should create a new folder "hw1" and put these three files in it.
- The filenames of the source code, the shell script, and the documentation file should be "main.c", "compile.sh", and "report.txt", respectively; you will get some penalties in your grade if your submission do not follow the naming rule.
- The documentation file should be in plain text format (.txt file). In the documentation file you should explain how your code works, and anything you would like to convey to the TAs.
- For Problem 2 to Problem 5, submit the answers through the CEIBA system (electronic copy) or to the TA in R508 (hard copy).
- Each student may only choose to submit the homework in one way; either all as hard copies or all through CEIBA except the programming assignment. If you submit your homework partially in one way and partially in the other way, you might only get the score of the part submitted as hard copies or the part submitted through CEIBA (the part that the TA chooses).
- If you choose to submit the answers of the writing problems through CEIBA, please combine the answers of all writing problems into only ONE file in the doc/docx or pdf format, with the file name in the format of "hw1_[student ID].{pdf,docx,doc}" (e.g. "hw1_b00902888.pdf"); otherwise, you might only get the score of one of the files (the one that the TA chooses).
- Discussions with others are encouraged. However, you should write down your solutions by your own words. In addition, you have to write the sources you consulted from or people you discussed with on the first page of your solution to that problem.
- No late submission of the homework will be given any score (for that portion).

Problem 1. Basic Matrix Operation (30%)

1. (10%) Matrix multiplication (which is your Homework #0).

2. (20%) You have implemented a matrix multiplication algorithm in Homework #0. Now, write a program to perform several more operations on more matrices. The operations supported by your program should include add, multiplication, and brackets. The input has the following format (download a sample of the input, "hw1_input_example", from the course website).

$N \leftarrow$ Number of matrices

$r_1 \quad c_1 \leftarrow$ Number of rows and columns of the first matrix

$a_{11}\ a_{12}\ \ldots\ \ldots\ a_{1c_1} \leftarrow$ the c_1 numbers of the first row

$a_{21}\ a_{22}\ \ldots\ \ldots\ a_{2c_1} \leftarrow$ the c_1 numbers of the second row

$\vdots$

$a_{r_1 1}\ a_{r_1 2}\ \ldots\ \ldots\ a_{r_1 c_1} \leftarrow$ the c_1 numbers of the r_1-th row

$r_2 \quad c_2 \leftarrow$ Number of rows and columns of the second matrix

$b_{11}\ b_{12}\ \ldots\ \ldots\ b_{1c_2} \leftarrow$ the c_2 numbers of the first row

$\vdots$

$b_{r_2 1}\ b_{r_2 2}\ \ldots\ \ldots\ b_{r_2 c_2} \leftarrow$ the c_2 numbers of the r_2-th row

$r_3 \quad c_3 \leftarrow$ Number of rows and columns of the third matrix

$\vdots$

$r_N \quad c_N \leftarrow$ Number of rows and columns of the $N-th$ matrix

$N_{11}\ N_{12}\ \ldots\ \ldots\ N_{1c_N} \leftarrow$ the c_N numbers of the first row

$\vdots$

$N_{r_N 1}\ N_{r_N 2}\ \ldots\ \ldots\ N_{r_N c_N} \leftarrow$ the c_N numbers of the r_N-th row

$Expression_1$

$Expression_2$

$\vdots$

$Expression_K$

(Note that you should read the expressions until seeing a EOF)

The expressions follow these rules:

(a) Capital letters are used to represent each matrix. A is always the first matrix, B is always the second matrix, ..., and Z is always the 26-th matrix.

(b) The operations include add '+', multiplication '*', left bracket '(', and right bracket ')'.

(c) The expressions are always valid mathematical expressions.

If the result of the calculation for $Expression_i$ is valid (e.g. no error occurs in matrix size in calculation between two matrices), output the matrix R_i (which is an m-by-s matrix) in the following format:

$r_{i,11}\ r_{i,12}\ \ldots\ \ldots\ r_{i,1s} \leftarrow$ the s numbers of the first row

$r_{i,21}\ r_{i,22}\ \ldots\ \ldots\ r_{i,2s} \leftarrow$ the s numbers of the second row

$\vdots$

$r_{i,m1}\ r_{i,m2}\ \ldots\ \ldots\ r_{i,ms} \leftarrow$ the s numbers of the m-th row

Otherwise, output an one-line message "error".

There are K expressions from the input; therefore, you should output the results for $Expression_1$, $Expression_2$, $\ldots$, $Expression_K$, respectively.

You can utilize the following assumptions:

(a) The number of matrices, N $\leq$ 26.

(b) Every entry of the matrix from input can be stored in a 16-bit integer. (Hint: note that there is no similar assumption for the output.)

(c) Both the number of rows and columns are always smaller than 50.

(d) The number of operators (including add, multiplication, and brackets) in an expression is smaller than 50.

(e) You can assume that there will be no overflow in the process of calculation if 64-bit (*long long int*) integers are used to store the matrix elements.

(f) Take the input from the standard input device (*stdin*).

Problem 2. Asymptotic Notation (25%)

1. (15%) Let $p(n) = \sum_{i=0}^{d} a_i n^i$, where $a_d > 0$, be a degree-d polynomial in n, and let k be a constant. Use the definitions of the asymptotic notations to prove the following properties. (please refer to p.43-52 of Cormen et al. for the notations which are not covered in the lecture.)

   (a) If $k \geq d$, then $p(n) = O(n^k)$.

   (b) If $k \leq d$, then $p(n) = \Omega(n^k)$.

   (c) If $k = d$, then $p(n) = \Theta(n^k)$.

   (d) If $k > d$, then $p(n) = o(n^k)$.

   (e) If $k < d$, then $p(n) = \omega(n^k)$.

2. (10%) Rank the following functions by the order of growth; that is, find an arrangement g_1, g_2, $\ldots$, g_{15} of the functions satisfying $g_1 = \Omega(g_2)$, $g_2 = \Omega(g_3)$, $\ldots$, $g_{14} = \Omega(g_{15})$. Partition your list into equivalence classes such that $f(n)$ and $g(n)$ are in the same class if and only if $f(n) = \Theta(g(n))$. Please state the reason on your rank in detail.

   $4^{\log n} \quad n \log n \quad n! \quad n^3 \quad \sqrt{\log n}$

   $2^{\sqrt{2 \log n}} \quad e^n \quad n^2 \quad \log^2 n \quad (\sqrt{2})^{\log n}$

   $n^{\log \log n} \quad (\log n)^{\log n} \quad \log(n!) \quad n^{1/\log n} \quad (\frac{3}{2})^n$

Problem 3. Time Complexity(15%)

1. (4%) We have implemented the matrix multiplication algorithm in Homework #0. Analyze the time complexity of the algorithm you used. Assume that both input matrices A and B are square matrices, i.e., the width and the height of the matrices are the same. Assume that the number of elements in both A and B are m. Express the time complexity in m.

2. (4%) Suppose c and d are both vectors of length m. Define an m-by-m matrix C, given by $C_{ij} = c_i + i + j$ $(0 \leq i, j < m$, i and j are integers), and an m-by-m matrix D, given by $D_{ij}, = d_i - i + j$ $(0 \leq i, j < m$, i and j are integers). Derive an algorithm to calculate the multiplication CD. Analyze the time complexity of the algorithm and express it in m.

3. (7%) It is very likely that you use the same algorithm to multiply two matrices in the previous two sub-questions, so it would only seem to make sense if the time complexities derived in these two sub-questions are the same. But they are not. Explain why.

Problem 4. Stacks and queues (20%)

Stacks and queues are both abstract data types that allow only a limited number of operations on a collection of data. The major difference between them is that, when deleting items, in a stack the last inserted item is the one to be deleted, while in a queue the first inserted item is the one to be deleted.

1. (5%) You are given two queues and you are asked to implement a stack using only these two queues. Write down the procedure step by step to depict how you implement the *pop* and *push* operations.

2. (5%) You are given two stacks and you are asked to implement a queue using only these two stacks. Write down the procedure step by step to depict how you implement the *enqueue* (add an item into a queue) and *dequeue* (take an item off from a queue) operations.

3. (10%) Given a stack and a queue, design an algorithm to determine whether they have the same elements and whether these elements are inserted in the same order. In addition, determine the time complexity (using the Big-O notation) of your algorithm. (Note: The algorithm is allowed to use only O(1) additional space, and you can assume that the capacities of the given stack and queue are both infinite.)

Problem 5. 2-SUM and 3-SUM(20%)

1. (4%) You are given a sorted integer array $a_0, a_1, \ldots, a_{n-1}$ in ascending order and the elements in the array are distinct. Suppose there exist M pairs (i, j) where $0 \leq i < j \leq n-1$, and $a_i + a_j = 0$. Please design a brute-force algorithm with time complexity $O(n^2)$ to calculate the value of M.

2. (5%) Now consider binary search. Please design a more efficient algorithm with time complexity $O(n \log n)$ to solve problem 5.1.

3. (5%) Given a sorted integer array $a_0, a_1, \ldots, a_{n-1}$ in ascending order, the following algorithm is designed to find if there exists three distinct index numbers, x, y, z, such that $a_x + a_y + a_z = 0$.

   ```
   for( i=0 ; i<n-2 ; i++ ){
        X = a[i];
        p = i+1;
        q = n-1;
        while ( p<q ){
            Y = a[p];
            Z = a[q];
            if ( X+Y+Z == 0 ){
                output(true);
                exit;
            }
            else if ( X+Y+Z > 0 ){
                q = q-1;
            }
            else {
                p = p+1;
            }
        }
   }
   ```

   Please derive the worst-case time complexity of the above algorithm.

4. (6%) Continue with problem 5.1, can you design a linear-time algorithm to find out the value of M?

   (Note: for problem 5.1, 5.2, and 5.4, you have to specify your algorithm with pseudo code and prove the time complexities of your algorithms.)