Data Structure and Algorithm I

Homework #1

Due: 5pm, Thursday, October 7, 2010

Submit the answers for problem 2-5 through the CEIBA system (electronic copy) or to the TA in R432 (hard copy). You also need to submit the answers of problem 1 through the CEIBA system.

Problem 1. (20%) In linear algebra, the determinant is a special number associated with any square matrix. Suppose $A = [a_{ij}] \in R^{n \times n}$, the determinant of A, denoted by $det(A)$, can be defined recursively as the following:

1. if $n = 1$, $det(A) = a_{11}$
2. if $n \geq 2$,
   $det(A) = a_{11}det(A_{11}) - a_{12}det(A_{12}) + \cdots + (-1)^{1+n}det(A_{1n}) = \Sigma_{j=1}^{n}(-1)^{1+j}a_{1j}det(A_{1j})$
   , where A_{1j} represents the $(n-1) \times (n-1)$ sub-matrix formed from deleting the 1-th row and j-th column of A.

Example 1. Suppose $A = \begin{bmatrix} 0 & 1 & 3 \\ -2 & -3 & -5 \\ 4 & -4 & 4 \end{bmatrix}$,

$$\begin{aligned} det(A) &= (-1)^{1+1} \cdot 0 \cdot det(A_{11}) + (-1)^{1+2} \cdot 1 \cdot det(A_{12}) + (-1)^{1+3} \cdot 3 \cdot det(A_{13}) \\ &= 0 \cdot det \begin{bmatrix} -3 & -5 \\ -4 & 4 \end{bmatrix} - 1 \cdot det \begin{bmatrix} -2 & -5 \\ 4 & 4 \end{bmatrix} + 3 \cdot det \begin{bmatrix} -2 & -3 \\ 4 & -4 \end{bmatrix} \\ &= 0 - 12 + 3 \times 20 = 48 \end{aligned}$$

Write a program to calculate $det(A)$ for any n. The input has the following format (download a sample of the input, "input_example", from the course website):

$n \leftarrow$ the order of the square matrix

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

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

$\vdots$

$a_{n1}\ a_{n2}\ \ldots\ \ldots\ a_{nn} \leftarrow$ the n numbers of the n-th row

The program should follow the following criteria:

1. Every entry of the matrix can be stored in a (32-bit) integer.
2. The dimension, 'n', could be any value. This means you have to dynamically allocate the memory to store the matrix.
3. Please calculate the determinant by using a recursive algorithm since the purpose of this problem is for you to learn it practically.
4. Take the input from the standard input device (stdin).

You must upload your homework in the format of a compress zip file to the CEIBA, and the zip file should include the following three files:

1. The source code (.c file),
2. A shell script to compile the source (.sh), and
3. A document in PDF format to explain how your program works.

Your score of 20% is divided into two parts: correctness (with 5 test cases)(15%) and document explanations(5%).

Problem 2. (20%) The time complexity of selection sort is $O(n^2)$ for all cases. In textbook p.45, Example 1.22 presents the time complexity analysis of the worst-case performance of selection sort by running program on a real machine. In this problem, we ask you to analyze the **average-case** performance of selection sort in the same way and finish the following three requirements:

1. For average case, plot the curves in a similar way to Figure 1.12. However, you should draw three curves in the plot; two of them are the curves of selection sort running on two different platforms (different operating systems or different machines). and the third curve is cn^2 where you have to estimate the parameter c such that cn^2 approximate one of the two curves (pick one) you drew before. Write down the specifications (OS, CPU, memory) of the machine you used. (10%)

2. Create the table similar to Figure 1.11 for one of the two platforms you used.(5%)

3. Compare the three curves. Do they overlap with each other? Please explain the reason of the difference between them under the same complexity $O(n^2)$.(5%)

You are allowed to modify the source code in textbook. You DO NOT need to submit the source code for this problem.

Problem 3. Please answer the following questions related to the asymptotic notation.

1. Determine whether the following statement is true or false. Please show how you obtain your answer for each statement.

   (a) $n! = O(n^n)$ (2%)

   (b) $n^{1.001} + n \log n = \Theta(n^{1.001})$ (2%)

   (c) For $T(n) = 4T(n/2) + n$ and $T(1) = 1$, $T(n) = O(n^2)$ (2%)

   (d) $10n^2 + 9 = O(\log n)$ (2%)

   (e) $T(n) = T(n-1) + \frac{1}{n}$ and $T(1) = 1$, then $T(n) = O(\log n)$ (2%)

   (f) $3^2 = 2^{O(n)}$ (2%)

2. Prove Theorem 1.3 and Theorem 1.4 in the textbook:

   (a) If $f(n) = a_m n^m + ... + a_1 n + a_0$ and $a_m > 0$, then $f(n) = \Omega(n^m)$. (4%)

   (b) If $f(n) = a_m n^m + ... + a_1 n + a_0$ and $a_m > 0$, then $f(n) = \Theta(n^m)$. (4%)

Problem 4. Please write down algorithms for each of the following three problems related to strings. Note that you are not allowed to use additional memory to store parts of the input string except the original memory used for storing the string.

1. Reverse the string. For example, if the input is "abc", then the output should be "cba". Please specify the prototype of your function, and describe what each parameter and the return value represents (5%).

2. The input string includes multiple sub-strings (which contains no space character) separated by space characters, ' '. Write an algorithm to reverse all sub-strings in

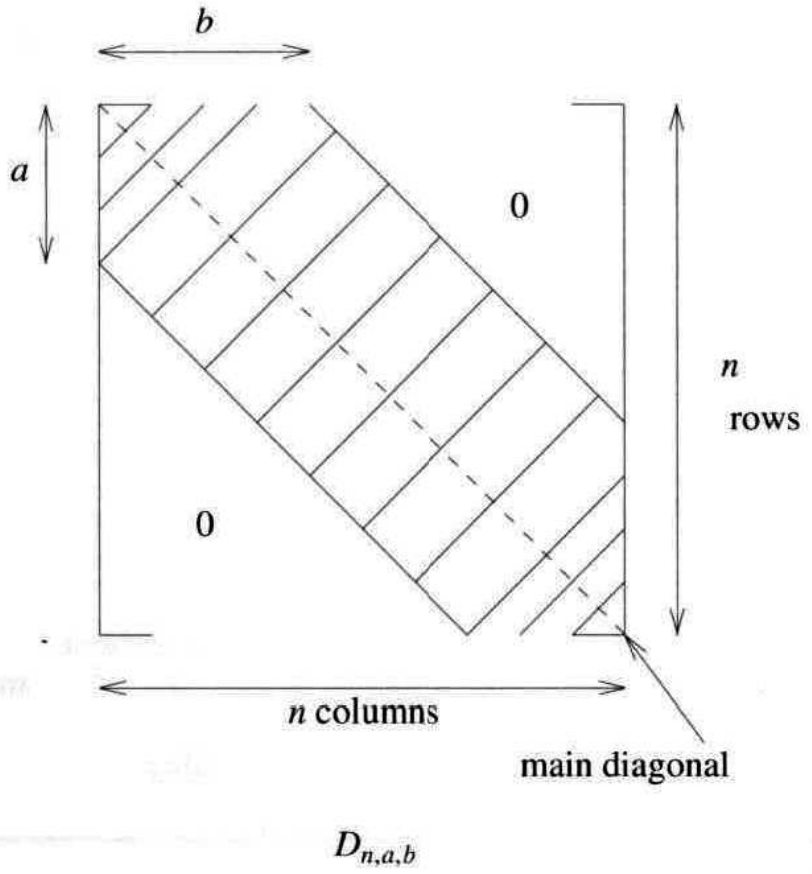

Figure 1: Generalized Band Matrix

the input. For example, if the input is "abc def ghi", then the output should be "cba fed ihg". Note that you must re-use the algorithm you developed in 1. (6%).

3. The input string includes two sub-strings (which contains no space character) separated by a space character. Write an algorithm to exchange the position of the two sub-strings. For example, if the input is "abcd ghi", then the output should be "ghi abcd". Note that you must re-use the algorithm you developed in 1. (9%).

Problem 5. A generalized band matrix $D_{n,a,b}$ is an $n \times n$ matrix in which all the nonzero terms lie in a band made up of $a-1$ diagonals below the main diagonal, the main diagonal, and b-1 bands above the main diagonal (See Figure 1).

1. How many elements are there in the band of $D_{n,a,b}$? (8%)

2. Explain how you would use a one dimensional array e to represent a generalized band matrix. Write a function $value(n, a, b, i, j, e)$ to retrieve the value of d_{ij} in the matrix from array e. (12%)