Data Structure & Algorithms
CS210A,ESO207A,ESO211
Semester I, 2012-13, CSE, IIT Kanpur

Programming Assignment II

Deadline : midnight of 7th September for the first problem and midnight of 14th September for the second problem

Note: This programming assignment consists of two problems. In case you face any problem in understanding these problems or their solution, you may contact the instructor to get some help or hint.

1 Data structure for compact representation of sparse matrices

The simplest way to represent a $n \times n$ matrix M is by a 2-dimensional array. A matrix M is said to be a sparse matrix if the nonzero entries in M are very few. Storing a sparse matrix using 2-dimensional array is not a compact way to store. However, any alternate data structure should be such that it facilitates efficient execution of various algorithms on matrix. The goal of this problem is to make you realize that there is a very elegant link based data structure for storing matrices which achieves compactness. A skeleton of the data structure you need to develop is shown in Figure 1.

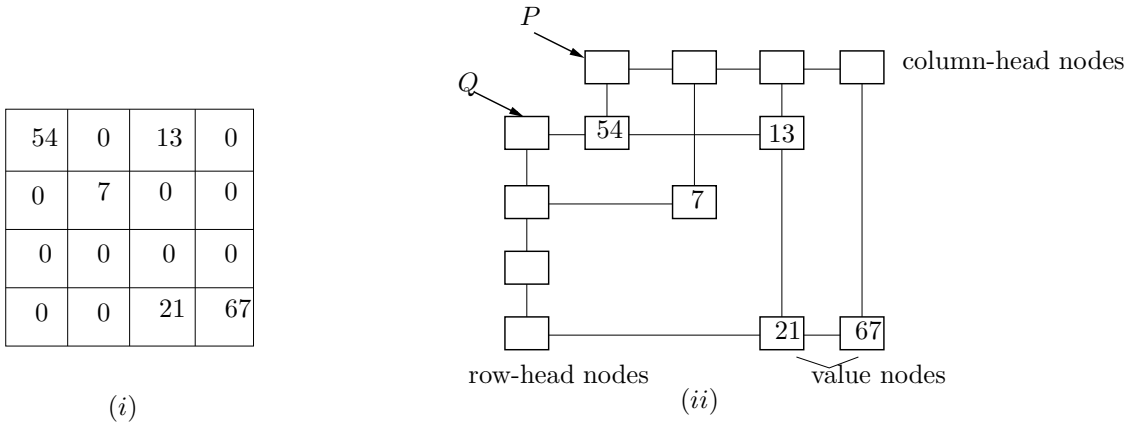

Figure 1: (ii) shows a skeleton the link based data structure for the sparse matrix shown in (i)

There are the following features of this data structure. The data structure for an $n \times n$ matrix will consist of exactly $2n + m$ nodes where m is the number of non zero entries in the matrix. Each node will have identical structure, that is, a row-head node, a value node, and a column-head node will have identical structure (your implementation has to use some way to distinguish these nodes). The matrix will be accessed by two pointers P and Q pointing to the lists of column-head nodes and row-head nodes respectively (see Figure 1).

As part of the assignment, you have to achieve the following objectives.

- Design the data structure with the above specification.
- You have to develop routines to read entries of matrices and store them in your data structure. You may assume that the non zero entries will be provided to you in the following order: First the nonzero entries of 1st row will appear, and then the nonzero entries of second row will appear, and

so on. The entries within a row will appear in the increasing order of their columns (see the sample input below).

- You have to design an algorithm to multiply two $n \times n$ matrices which are stored in the data structure designed above.

Format of the input: The first line will represent n. Thereafter the nonzero entries of matrices will appear in the following format. Each line will consist of four numbers. The first number will represent the matrix (first or second). The second and the third numbers will represent the row and column of the entry. The fourth number will represent the value of the entry. For example, 1 4 2 36 means that there is a nonzero entry in the 4th row and 2nd column of the first matrix and its value is 36. First all the nonzero entries of first matrix appear and then the nonzero entries of the second matrix appear. Finally a line consisting of a single 0 means the end of the input. You may assume that input to your program will indeed be a valid input.
Sample input:

```
3
1 1 3 25
1 2 1 33
1 2 2 14
1 3 1 8
2 1 1 45
2 1 2 79
2 3 1 109
2 3 3 56
0
```

You may print the output of your algorithm (for product of the matrices) in the above format as well.

2 Data structure for evaluating expressions

We discussed a very elegant algorithm using stacks to evaluate an arithmetic expression. The aim of this problem is to implement a slight extension of the algorithm. Firstly, let us ask ourselves what is an arithmetic expression. We know that $3 + 5^6$ is an arithmetic expression whereas neither $3 * +6 - 5$ nor $2 * (3 + 5))$ is an arithmetic expression. Can we give a precise definition of an arithmetic expression ? Think over it.

Indeed it is possible through recursion. The following is a recursive definition of an arithmetic expression:

- Each number is an arithmetic expression and its value is the same as the number. Likewise a variable of type number is also an arithmetic expression and its value is the number assigned to the variable at present.
- if α is an arithmetic expression then (α) is also an arithmetic expression whose value is the same as the value of α.
- if α and β is an arithmetic expression, then $\alpha \circ \beta$ is also an arithmetic expression where $\circ$ is a suitable binary operator.

Notice that the precedence and associativity of the operators together ensure that each expression has a unique value. In addition to the usual binary operators,

`+, - , * , / , ^`

we introduce an additional operator = which can be described as follows. If x is a variable and α is an expression, then $x = \alpha$ is also an expression. The value of expression $x = \alpha$ is defined as the value of expression α. The evaluation of $x = \alpha$ involves evaluation of α and assigning this value to variable x. Notice that = is right associative.

You have to extend the algorithm discussed in the class suitably to incorporate the new operator = which is defined above. You may assume that the expression will have at most two floating point variables and they will be denoted by x and y. You have to assume that the value of these variables is 1 before the evaluation of the expression. Your output has to print the value of x, the value of y, and the value of the expression. You may assume that input expression is a valid arithmetic expression.
Sample input and output:

- input:
  3+5^(2*3-(x+y))
  output:
  value of x is 1, value of y is 1.
  value of expression is 628.

- input:
  3*6+x=2+x*4-y
  output:
  value of x is 5, value of y is 1.
  value of expression is 23.

- input:
  3+x=2^4+3*y=3*(y+4^2)-22
  output:
  value of x is 103, value of y is 29.
  value of the expression is 106.

Note: Students from CSE department will see more of expression evaluation or "parsing" during their course on compilers or theory of computation. Compared to that, the above problem is just a toy example. However, its aim is just to address algorithmic aspect of expression evaluation.