Course: ESO207A – Data Structures and Algorithms

Indian Institute of Technology Kanpur

Programming Assignment 2 : *Data structure for compact representation of sparse matrices*

Most Important guidelines

- It is only through the assignments that one learns the most about the algorithms and data structures. You are advised to refrain from searching for a solution on the net or from a notebook or from other fellow students. Remember - **Before cheating the instructor, you are cheating yourself**. The onus of learning from a course lies first on you. So act wisely while working on this assignment.
- Refrain from collaborating with the students of other groups. If any evidence is found that confirms copying, the penalty will be very harsh. Refer to the website at the link: https://cse.iitk.ac.in/pages/AntiCheatingPolicy.html regarding the departmental policy on cheating.
- This assignment has to be done in groups of 2 only. It is your responsibility to find a partner.
- In case of any issue related to this assignment, send an email at dravya@cse.iitk.ac.in (and not to the instructor)

The Objective of the Assignment

The objective of the assignment is to make you realize that we may use link based structures to compactly represent a sparse mathematical structure.

Background of the Assignment

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.

There are the following features of this data structure. The data structure for an $n \times n$ matrix will consist of $O(m)$ nodes where m is the number of non zero entries in the matrix. Each node will have identical structure. The matrix will be accessed by two pointers (one each for row and column heads).

Tasks to be done

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 make sure that you read the input as efficiently as possible - both in terms of space and time - making use of the fact that the input is being provided to you in a specific format.
- 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 (assume the value to be within bounds of unsigned integer). 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 (assume the value to be within bounds of signed integer). 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.
To print the output, use a similar format. Each line of output will consist of three numbers. The first and the second numbers will represent the row and column of the entry of your product matrix. The third number will represent the value of the entry. The order of entries remains similar to input (increasing rows and columns). In case the final product matrix is a null matrix, print "NULL MATRIX!".
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
```

Sample output:

```
1 1 2725
1 3 1400
2 1 1485
2 2 2607
3 1 360
3 2 632
```

Points to Remember

- You would need to submit C code for the following assignment which takes input (from standard input) and prints output (to standard output) in the format given above.
- Submissions will be made on Moodle. Submit a single C file containing the code for the assignment. Both the students in a group must submit the same file on Moodle. Name of the C file should be in the following format -
  - If there are 2 persons in the team with roll numbers 123456 and 987654, then the filename should be 123456_987654_A2.c
- The evaluation would include both automated and manual components. The code would be checked against test cases which would be **60%** of your grade in the assignment. The rest **40%** would be manual grading, which would check for your approach to the solution and the coding practices (comments and indentation).
- Some sample test cases are being shared with you in this link to test your code. But the final grading will be done using a different set of test cases.
  `https://drive.google.com/drive/folders/1w-pJxOBLUy6qdMZOvJAGPV94SP6RO4oM?usp=sharing`
- In case you want to generate your own test cases to test your code, you can look up the scipy.sparse module in python, which might help with that.

At this point, stop and think about the solution to the problem. On the next few pages, there are hints to how you could try solving this. You are strongly advised to look at these hints only after you have spent sufficient time thinking over the problem.

Hint #1

You have been asked to design a data structure for a sparse 2-D array. Think of a simpler problem first - a sparse 1-D array. How would implement it? Is there a “link” based data structure suitable for it? What kind of information would you need to store in this data structure to access the elements correctly? How does access of element work in this data structure?

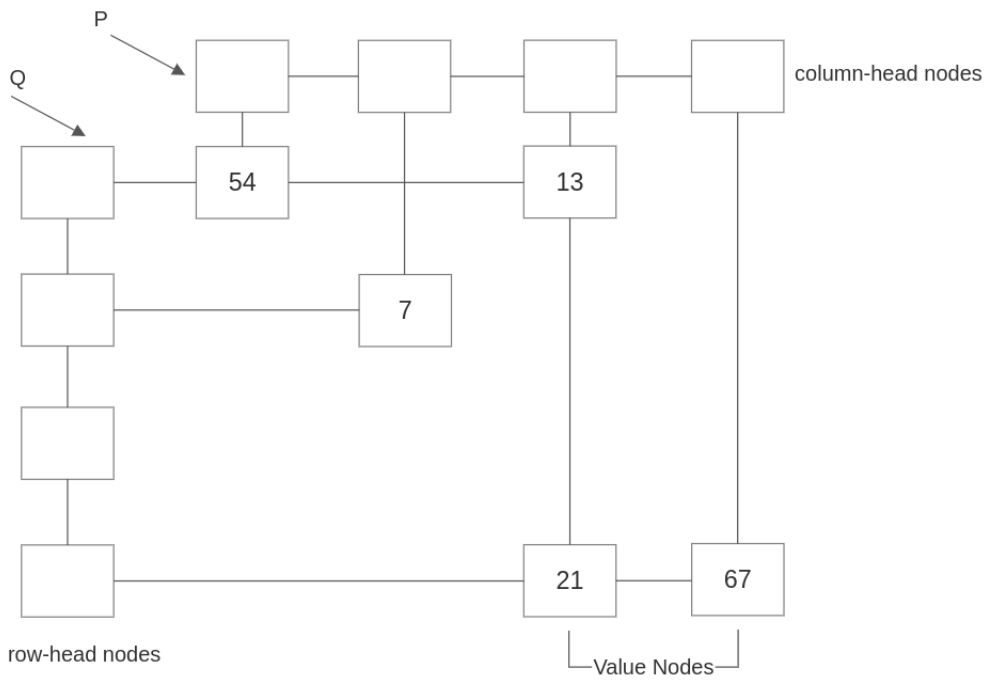
Hint #2

How can you replicate your solution for 1-D array to a 2-D array? How would accessing elements work here? What is the direction of the links in your “link” based data structure? Is it just one direction or more than that? (Remember that you need to perform matrix multiplication, which would require efficient access of elements in a row as well as in a column).

Hint #3

For the matrix given below, the diagram at follows might shed some light on how your data structure should look.

54	0	13	0
0	7	0	0
0	0	0	0
0	0	21	67



Hint #4

Remember, the more efficient your code is in terms of space and time, the better it is (the optimizations carry some marks). How would you achieve that? Don't make any nodes in your data structure that are not really needed. For example, there is a nonzero number of nodes in the diagram on page 6 that can be avoided. Also exploit the fact that the input given to you is in a specific format. Reading the input and building the corresponding matrix from this input should take $O(m)$ time, and the space complexity for the data structure should also be $O(m)$. (Here m is the number of non-zero entries in the matrix)