Xiao Feng

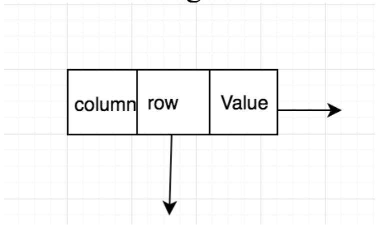
Lab3 Analysis:

Data Structure:

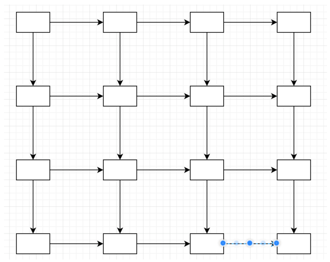
I utilized a 2D linked list structure to store input matrix and a recursion to complete determinant calculation of the matrix. Compare to lab2, which implemented with array data structure. Linked list has the following benefit:

1. Linked list does not require to specify the size of the matrix when initiating the matrix. The constructor of linked list matrix doesn't need any parameter like columns or rows. It initiates a node and one pointer to right and one pointer to the below.
2. Efficiency of memory. Array requires a whole segment of space to store data values, while linked list nodes does not need to be located next to each other, nodes are connected through pointers.
3. For a sparse matrix, linked list can skip zeros only save non-zeros as nodes. It largely saves memory.

I used multi Linked list structure. One node has two pointers, one to the right of the node and one to the node below it. Each node stores a value and information about column number and row number. A single node:



Matrix:



When reading a value from input, a new node and pointers to the right and down are created, then let left node point to the new node and above node point to new node. Followed a sequence of right to left, up to down, a matrix is successfully loaded.

In the determinate calculation process, I used recursion. Recursion is a sound choice to a determinate problem because the process to get the determinant of a defined matrix is a recursive process. Matrix's determinant is calculated through its minor matrix's determinant; the minor matrix's determinant is getting from its own minor matrix's determinant. Until it hit stopping point matrix 1*1, it returns x. Minor matrix is created through the following steps:

1. Initiate a new node

2. Find left top corner node for minor matrix from the original matrix
3. Copy value of current node to new node
4. Move to next right node for minor matrix from the original matrix
5. Repeat 3-4 until all values are copied to new node
6. Note: to skip the next right node, do node.right = node.right.right; similarly to skip next row of the node, do node.down = node.down.downO(g(n))

Determinant calculation contains two major cost, one is minor matrix generation, another is function of `power(-1, i + j) * a[i, j] * det(minor(a[i, j]))` .

For a n*n matrix (n^2 datasize), one minor matrix generation is (n-1)*(n-1). There are n minor matrix. Minor matrix generation cost is: (n-1)(n-1)n + (n-2)(n-2)(n-1) + … + 1 = f(n^4)

Calculation cost equals number of minor matrix: n+(n-1)+(n-2)+(n-3)+…+1 = n(n+1)/2=f(n^2)

Because data size is n^2, Therefore, overall cost is O(n^2)

Enhancement:

1. Error messages are very specific descriptions of what type of error was encountered for which matrix, which is written in the output file.
2. User-friendly print out in output.
3. 15 supplement testing cases
4. Stopping cases are matrix 1*1 and matrix 2*2.
5. include deep copy constructor
6. skips determinant when entry is 0 or negative integer
7. track of program executing timing
8. Allows for Real numbers as data (not just integers)

Learnings:

Lab3 is a good practice to implement a linked list data structure and utilizing recursion function. By applying the linked list data structure to solve a real question, I had a better understanding of how linked list works and the benefit it brings to an efficiency of the programming process. In the later development work, I'm more confident in applying linked list to solve issues.

Something I can improve is:

1. How row and column information gets stored. In Lab3, I add row and column information on every node. It could be more memory efficient by adding header and use a circular for columns and rows.
2. How minor matrix is created. In lab3, I initiated a new matrix, copy value and linked nodes to develop the new matrix by moving going through the original matrix. An alternative way is to do a deep copy on the original matrix, cut-off pointers of certain nodes. For example, to cut off node(3,4), let node(2,4).down = nod(3,4).down and node(3,3).right = node(3,4).right.