TASK 2:

Consider the task of multiplying two integer matrices A and B, each of size 500 × 500. Each matrix can have at most 500 non-zero entries. Any row of A or B that contains at least one non-zero element must have no less than 50 non-zero elements. (10 Marks).

a. Propose a space-efficient data structure for this task.
b. Based on your proposed data structure, design an efficient algorithm to multiply A and B. The resulting matrix should be stored separately.
c. What is the maximum number of integer multiplications needed to complete this task as per your implementation?

SOLUTION:

Considering Integer to be of X bytes, to store 500 x 500 matrix using orthodox method of using a 2D array would consume around 250000 * X bytes just of one matrix which is not space efficient. I would use a 2D map(Hash table) for storing the elements. This would take space in order O(n) where n is the number of non zero elements(max 500). The data structure would also help in performing the matrix multiplication which is the primary reason to choose such arrangement.

Let us see how the Matrix is represented in the data structure. Let us take the 5x5 sparse matrix for the example.

$$A = \begin{bmatrix} 0 & 0 & 0 & 5 & 4 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 16 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 22 & 12 \end{bmatrix}$$

```
matA = {
         0:{
             3: 5,
             4:4,
           },
         2:{
             2:16,
             3:10
           },
         4:{
             3:22,
             4:12
          }
         };

// Note : The map is not necessarily sorted.
```

The representation of a non-zero element of a matrix lets say A would be A[i][j] where i is represented by row index and j as column index in the map. The complexity to extract the element at position <i,j> is O(1).
Formula for Matrix multiplication of two matrices is given by

$$C_{ij} = \sum_{k=0}^{n-1} A_{ik} * B_{kj}$$

$$A = \begin{bmatrix} A_{00} & A_{01} & A_{02} \\ A_{10} & A_{11} & A_{12} \\ A_{20} & A_{21} & A_{22} \end{bmatrix} B = \begin{bmatrix} B_{00} & B_{01} & B_{02} \\ B_{10} & B_{11} & B_{12} \\ B_{20} & B_{21} & B_{22} \end{bmatrix}$$

$$C = \begin{bmatrix} A_{00}.B_{00} + A_{01}.B_{10} + A_{02}.B_{20} & A_{00}.B_{01} + A_{01}.B_{11} + A_{02}.B_{21} & A_{00}.B_{02} + A_{01}.B_{12} + A_{02}.B_{22} \\ A_{10}.B_{00} + A_{11}.B_{10} + A_{12}.B_{20} & A_{10}.B_{01} + A_{11}.B_{11} + A_{12}.B_{21} & A_{10}.B_{02} + A_{11}.B_{12} + A_{12}.B_{22} \\ A_{20}.B_{00} + A_{21}.B_{10} + A_{22}.B_{20} & A_{20}.B_{01} + A_{21}.B_{11} + A_{22}.B_{21} & A_{20}.B_{02} + A_{21}.B_{12} + A_{22}.B_{22} \end{bmatrix}$$

Thus as we can see we only need to multiply the elements of A (A[i][j]) with elements of B (B[k][l]) only if j==k;

Algorithm:

```
// Let us say two 500 x 500 matrix are given named A and B
int A[500][500] = [[0,0,...,0],[0,0,...,0,0]...[0,4,5,...,0]];
int B[500][500] = [[0,0,...,0],[0,0,...,11,12]...[0,0,0,...,0]];

typedef unordered_map<int,unordered_map<int,int>> Matrix;

Matrix mtA;
Matrix mtB;
Matrix mtC;
```

```
void matrixTransform(int &X[][], Matrix &M){
  for i= 0 to n-1:
    for j = 0 to n-1:
        if X[i][j]: M[i][j] = X[i][j]
}

matrixTransform(A,mtA);
matrixTransform(B,mtB);

for i in keys of mA;
  for j in keys of mA(i):
    for k in keys of mB(j):
     mtC[i][k] += mA[i][j] * mB[j][k];
```

The number of multiplications for 2 Matrices A(m x n) and B(n x p) is (m x n x p). In the given scenario, the values of m, n and p are 500. But not all elements have a value. There are at least 249500 zeros which we need not multiply. So we can consider an array of
10 x 50 at most, this is so, because, the problem statement says, if a row has elements, it has at least 50 elements, so if we consider 50 elements to be represented.
While multiplying matrices of 10 x 50, the number of columns from left matrix A will only be multiplied if that many rows are available in the right matrix B. Therefore, in this scenario, values of m x n (10 x 50) and n x p (10 x 50) the n from the left Matrix will only be used for 10 rows of right matrix. Which means for number of multiplications we can say values for:

m = 10,
n = min(nLeft, nRight) [Only for a sparse matrix] = min(10, 50) = 10 and
p = 50

Therefore, maximum number of integer multiplications needed are

m x n x p = 10 x 10 x 50 = 5000