Crack a hack

Matrix Layer Rotation

Algorithm Problem Solving
17ECSE309

USN:01FE15BEC138
Name : Rahul Jain

Problem statement

- You are given a 2D matrix of dimension m x n and a positive integer r. You have to rotate the matrix r times and print the resultant matrix. Rotation should be in anti-clockwise direction.
- Rotation of a matrix is represented by the following figure. Note that in one rotation, you have to shift elements by one step only.

$$\begin{tikzcd}
a_{11} \arrow[d] & a_{12} \arrow[l] & a_{13} \arrow[l] & a_{14} \arrow[l] & a_{15} \arrow[l] \\
a_{21} \arrow[d] & a_{22} \arrow[d] & a_{23} \arrow[l] & a_{24} \arrow[l] & a_{25} \arrow[u] \\
a_{31} \arrow[d] & a_{32} \arrow[r] & a_{33} \arrow[r] & a_{34} \arrow[u] & a_{35} \arrow[u] \\
a_{41} \arrow[r] & a_{42} \arrow[r] & a_{43} \arrow[r] & a_{44} \arrow[r] & a_{45} \arrow[u]
\end{tikzcd}$$

Matrix Rotation

Algorithm

- Find minimum of Number of rows and column
- Compute the number of layers
- Convert 2D array into 1D
- Rotate it by r times and assign back it to 2D array

Code:

```
#include <math.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <assert.h>
#include <limits.h>
#include <stdbool.h>



int main() {
   long long m;
   long long n;
   long long r;
   scanf("%lli %lli %lli", &m, &n, &r);
   long long matrix[m][n];
   for (long long matrix_i = 0; matrix_i < m; matrix_i++) {
      for (long long matrix_j = 0; matrix_j < n; matrix_j++) {

         scanf("%lli",&matrix[matrix_i][matrix_j]);
      }
   }
```

```
long long count = 0; // number of layers
   long long temp[n*m];
   long long k;
   if(m>n)
       k=n;
   else
       k=m;

   if(k%2 == 0){
       count = k/2;
   }
   else
   {
       count = k/2 +1;
   }
```

```
for(long long k =0;k<count;k++){  //converting 2-D matrix into 1-D array
        long long g =0;
        for(long long i =k;i<n-k;i++){
          temp[g] = matrix[k][i];
            g++;
        }
        for(long long i = k+1;i<m-k;i++){
            temp[g] = matrix[i][n-1-k];
            g++;
        }
        for(long long i = n-k-2;i>=k;i--){
            temp[g] = matrix[m-1-k][i];
            g++;
        }
        for(long long i = m-k-2;i>=k+1;i--){
            temp[g] = matrix[i][k];
            g++;
        }
long long *a = malloc(sizeof(long long) * g); //1-D array used for rotating

        for(long long i=0; i<g; i++){
      long long j = ((i - r)% g + g) % g;
            a[j] = temp[i] ;
      }
      g =0;
      for(long long i =k;i<n-k;i++){  //coverting to 2-D array
        matrix[k][i] = a[g];
          g++;
      }
      for(long long i = k+1;i<m-k;i++){
          matrix[i][n-1-k] = a[g];
          g++;
      }
      for(long long i = n-k-2;i>=k;i--){
          matrix[m-1-k][i] = a[g];
          g++;
      }
      for(long long i = m-k-2;i>=k+1;i--){
          matrix[i][k] = a[g];
          g++;
      }

   }
```

```
 for(long long i =0;i<m;i++){
        for(long long j =0;j<n;j++){
    printf("%lld ",matrix[i][j]); //print the rotated matrix
        }
        printf("\n");
    }
    return 0;
}
```