Research & Reviews: Journal of Engineering and Technology e-ISSN:2319-9873

KareemNaaz Matrix Divide and Sorting Algorithm

Shaik Kareem Basha*

Department of Computer Science and Engineering, HITAM, India

Review Article

Received date: 18/11/2016

Accepted date: 13/12/2016

Published date: 19/12/2016

***For Correspondence**

Shaik Kareem Basha, HOD of Computer Science and Engineering, HITAM, India, Tel: +91-8712273844.

E-mail: s.kareem.basha@gmail.com

Keywords: Algorithm, Columns, Matrix, Elements

ABSTRACT

Sorting is a process of arranging data in a specific order according to requirements of Application. Efficient sorting brings orderliness in the data. Many sorting algorithms are proposed. Each algorithm has its own advantages and disadvantages. In this paper an efficient matrix sorting algorithm called as KareemNaaz Matrix Divide and Sorting Algorithm is proposed. The proposed Matrix Sorting Algorithm takes a matrix of order (mxn) as input. It recursively divides the given matrix of order (mxn) into two groups of rows of matrix, based on the middle row. Each group of rows consists of m/2 rows of matrix. First group contains rows from first row to middle row of matrix and second group contains rows from middle+1 row to last row of matrix. Recursively groups of rows are selected and divided into sub groups of rows. If sub group of rows contains only one row, then it is divided into two groups of columns, based upon the middle column of row, each group of columns consists of n/2 columns. First group contains columns from first column to middle column of row, second group contains columns from middle+1 column to last column of row. Recursively groups of columns of row are selected and divided into sub groups of columns. If sub groups of columns contains single column then combine and sorting of those columns takes place. I followed the various stages of Software Development Life Cycle to demonstrate the proposed sorting algorithm. Section I will give introduction about proposed sorting algorithm. In section II, proposed Algorithm is Analyzed and Designed. In section III, proposed algorithm is Implemented using C programming Language. In section IV, I tested the implementation of proposed algorithm using different test cases. In section V, I concluded the proposed Algorithm.

INTRODUCTION

Sorting is a process of arranging data in a specific order according to requirements of application. In complex software applications, sorting plays a major role to reduce searching time of data items [1]. Many sorting algorithms are developed. Each algorithm has its own advantages and disadvantages. Often a developer is puzzled as to which algorithm is best and efficient. The efficiency can be measured in terms of memory usage, time taken to sort, cpu usage etc. In this paper an efficient matrix sorting algorithm called as KareemNaaz Matrix Divide and Sorting Algorithm is proposed. The proposed KareemNaaz Matrix Divide and Sorting Algorithm takes a matrix of order (mxn). It recursively divides the given matrix of order (mxn) into two groups of rows of matrix, based on the middle row. Each group of rows consists of m/2 rows of matrix. First group contains rows from first row to middle row of matrix and second group contains rows from middle+1 row to last row of matrix. Recursively groups of rows are selected and divided into sub groups of rows. If sub group of rows contains only one row, then it is divided into two groups of columns, based upon the middle column of row, each group of columns consists of n/2 columns. First group contains columns from first column to middle column of row, second group contains columns from middle+1 column to last column of row. Recursively groups of columns of row are selected and divided into sub groups of columns. If sub groups of columns contains single column then combine and sorting of those columns takes place. The proposed algorithm is mainly based upon Divide and Conquer strategy and it is an extension of Merge Sort Algorithm, which is used to sort n elements of linear list by dividing the list into two sub lists, based upon the mid element. Left sub list contains elements from first element to mid element of list and right sub list contains elements from mid+1 element to last element of list. Its overall time complexity is O(n log n), whereas the time taken by proposed KareemNaaz Matrix Divide and Sorting Algorithm is O(mn log mn), if linked list is used to combine sorted rows of matrix of order mxn. This is derived in section II of this paper.

Analysis and Design of Proposed Algorithm

Figure 1 shows the division of given matrix of order (mxn) into sub matrices and combining sorted sub matrices at the end. Let us assume (m=3) is the number of rows of a matrix and (n=3) is the number of columns per row of a given matrix. The given matrix of order mxn is divided into two groups of rows, based on the middle row of matrix. Each group of rows consists of (m/2=3/2) rows, recursively each group of rows is selected and divided into sub groups of rows, if sub group of rows consists of only one row, then divide the columns of row into two groups of columns, each group of columns of a row consists of (n/2=3/2) columns, recursively select each group of columns and divide it into sub groups of columns. If sub groups of columns consists of only one column then combine and sort elements of columns of a row.

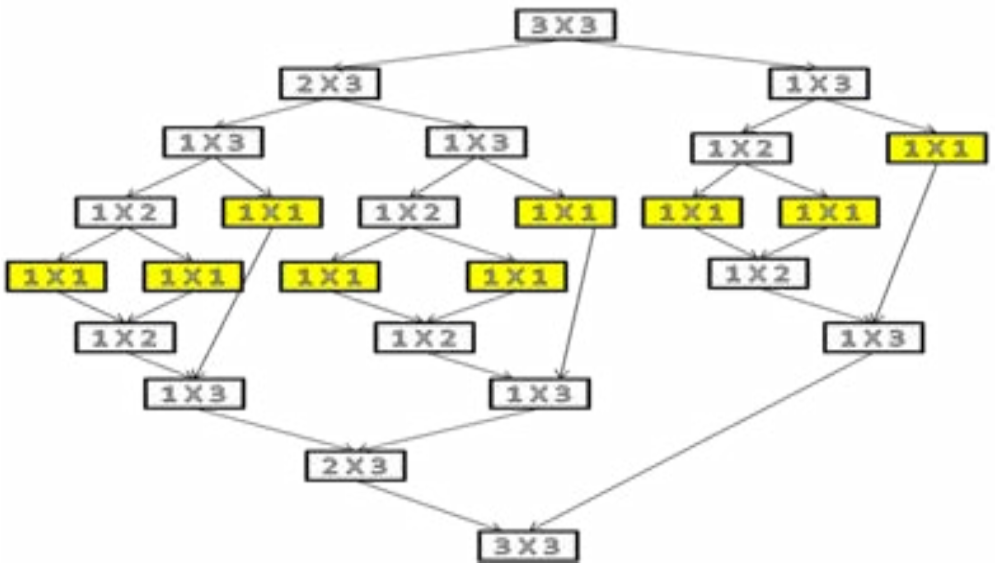

Figure 1. Division of matrix into sub matrices and combining sorted sub matrices.

Figure 2 shows a matrix of order (3x3), which contains 9 elements {{1 4 3}, {7 8 6}, {2 5 9}}. It is divided into two groups of rows, based on the middle row of matrix. First group A of rows contains 2 rows {{1 4 3} { 7 8 6 }} and second group B of rows contains 1 row {2 5 9}. Recursively first group A of rows is selected and is divided into two sub groups of rows, first sub group C of rows contains only 1 row {1 4 3} and second sub group D of rows contains only 1 row {7 8 6}. Recursively sub group C is selected, since it consists of only 1 row, so it is divided into 2 groups of columns, based on the middle column of row. First group C1 of columns contains 2 columns {1 4} of row and second group C2 of columns contains 1 column {3} of row. Recursively first group C1 of columns is selected and is divided into two sub groups of columns each contain only one column. This is a recursive process, which continues until all the elements of matrix of order (mxn) are sorted in increasing order from first column of first row to last column of last row.

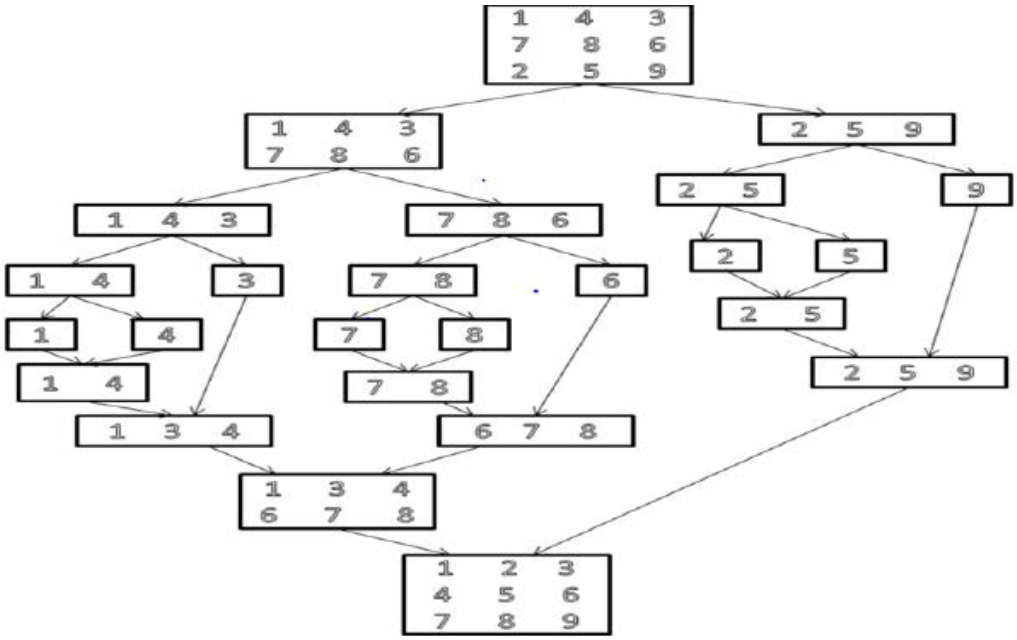

Figure 2. Dividing 3X3 matrix into sub matrices and Combining sorted sub matrices.

Algorithm 2.1 combines the sorted rows of matrix of order mxn from first row to last row, each row of matrix consists of n columns. This algorithm [2] takes index of first row of matrix, index of last row of matrix, index of first column of each row of matrix, index of last column of each row of matrix, external matrix a[][] of an order nxn as input. It places the elements of each row of matrix into linear list, sorts all the elements of linear list and places the sorted elements of linear list into columns of each row of matrix of order mxn.

```
Algorithm KNCombineRows (row1, row2, col1, col2) {
// row1 is the index of first row of matrix a[][]
// row2 is the index of last row of matrix a[][]
// col1 is the index of first column of each row of matrix a[][]
// col2 is the index of last column of each row of matrix a[][]
// a[][] is a matrix of order mxn
 for(i=row1;i<=row2;i=i+1) { // loop repeats from first row to last row of matrix
 for(j=col1;j<=col2;j=j+1) { // loop repeats from first column to last column of each row of //matrix
 k=k+1;
 temp[k]=a[i][j]; // placing elements of matrix into linear list temp
 for(m=k;m>1;m=m-1) { // sorting of elements of linear list temp
 if(temp[m]<temp[m-1]) // if element at index m of temp is smaller than element at index m-1 //of temp
 swap(temp[m],temp[m-1]); // swap elements of temp at indices m and m-1
 else break;
 } } }
 k=0;
 for(i=row1;i<=row2;i=i+1) { // loop repeats from first row to last row of matrix
 for(j=col1;j<=col2;j=j+1) { // loop repeats from first column to last column of each row of //matrix
k=k+1;
 a[i][j]=temp[k]; placing elements of sorted linear list temp into matrix
} } }
```

Algorithm 2.1 Combining Sorted Rows of a matrix

In Algorithm 2.1, the time taken to place elements of each row of matrix into linear list is mn units, since matrix consists of m rows and each row consists of n columns. If the linear list is Array List then the time taken to sort elements of linear list is $(mn)^2$ and if the linear list is Linked List then the time taken to sort elements of linear list is mn, since linear list consists of mn elements of matrix. After sorting, once again the sorted elements of linear list are placed into matrix of order mxn, which takes mn units of time. Let $T_{cr}(m,n)$ is the time taken to combine sorted m rows of matrix of order mxn, each row consists of n columns.

If Linear List is Array List then the overall time taken by Algorithm 2.1 is as follows:

$T_{cr}(m,n)=mn + (mn)^2 + mn$

$T_{cr}(m,n)=2mn + (mn)^2$

$T_{cr}(m,n)=O((mn)^2)$

If Linear List is Linked List then the overall time taken by Algorithm 2.1 is as follows:

$T_{cr}(m,n)=mn + mn + mn$

$T_{cr}(m,n)=3mn$

$T_{cr}(m,n)=O(mn)$

Algorithm 2.2 combines the sorted columns of a row of matrix of order mxn. It takes index of row, whose sorted columns are combined, index of first column of row, index of last column of row, index of middle column of row of matrix as input. Actually it combines elements of 2 sorted group of columns of a row, group 1 contains columns from first column to middle column of row, group2 contains columns from middle+1 column to last column of row, by comparing elements of columns of each group. Smaller

element of column from any group is selected and placed into linear list. After placing all the elements of columns of two groups into linear list in sorted order, the sorted elements are again replaced into columns of row.

```
Algorithm KNCombineCols(row, col1, midcol, col2) {
// row is the index of a row of matrix a[][]
// col1 is the first column of row of matrix a[][]
// col2 is the last column of row of matrix a[][]
// midcol is the middle column of row of matrix a[][]
// a[][] is a matrix of order mxn
i=col1; j=midcol+1;
while(i<=midcol && j<=col2) { //index i repeats from first column to middle column of row of //matrix and index j repeats from middle+1 column to last column of row of matrix
if(a[row][i]<a[row][j]) {// if element at column i of row is smaller than element at column j of //row
b[k]=a[row][i]; //place element at column i of row into linear list b
i=i+1;
} else {
b[k]=a[row][j]; // place element at column j of row into linear list b
j=j+1; }
k=k+1; }
while(i<=midcol) { // place elements from column i to mid column of row into linear list b
b[k]=a[row][i];
k=k+1;
i=i+1; }
while(j<=col2) {// place elements from column j to last column of row into linear list b
b[k]=a[row][j];
k=k+1;
j=j+1; }
for(i=col1;i<=col2;i=i+1) { // place all the sorted elements of linear list b into row of a matrix
a[row][i]=b[i];
} }
```

Algorithm 2.2 Combining Sorted Columns of a Row

Algorithm 2.2 compares elements of columns of two groups, and places into linear list in sorted order. Each group of columns consists of n/2 columns, since all the columns of a row are exactly divided into two groups, based on the middle column. So the time taken to place elements of columns into linear list in sorted order is n units, since n/2 elements of first group of columns and n/2 elements of second group of columns are placed into linear list in sorted order. At last all the sorted elements of linear list are placed into columns of a row. Let $T_{cc}(1,n)$ is the time taken to combine n sorted columns of a row of matrix of order mxn. So the overall time taken by Algorithm 2.2 is as follows:

$T_{cc}(1,n)=(n/2) + (n/2) +n$

$T_{cc}(1,n)= 2(n/2) + n = 3n$

$T_{cc}(1,n)= O(n)$

Algorithm 2.3 divides the columns of a row into two groups of columns, based on the index of mid-column of a row. Each group of columns consists of (n/2) columns. Group1 consists of columns from first column to middle column of row, group2 consists of columns from middle+1 column to last column. Recursively the group of columns are selected and divided into smaller sub groups. If sub groups consists of only one column, then these sub groups are combined and sorted.

```
Algorithm KNDivideCols( row, col1, col2) {
// row is the index of row of a matrix
// col1 is the first column of row of a matrix
// col2 is the last column of row of a matrix
 if(col1<col2) { // if index of first column is smaller than index of second column of row
 midcol=(col1+col2)/2; // find index of middle column of a row
 KNDivideCols(row,col1,midcol); // divide columns from first column to middle column of row
 KNDivideCols(row,midcol+1,col2);// divide columns from middle+1 column to last column of //row
KNCombineCols(row,col1,midcol,col2); // combine all the sorted columns of a row
 } }
```

Algorithm 2.3 Dividing UnOrdered Columns of a Row

Let $T_{dc}(1,n)$ is the time taken to divide n columns of a row into two groups of columns, based upon the mid column of a row. Each group of columns consists of n/2 columns. The time taken by Algorithm 2.3 is as follows:

$T_{dc}(1,n)=2T_{dc}(1,n/2) + T_{cc}$

$T_{dc}(1,n)=2T_{dc}(1,n/2) + n$

$T_{dc}(1,n)=2(2T_{dc}(1,n/4) + (n/2)) + n$

$T_{dc}(1,n)=2^2T_{dc}(1,n/4) + 2n$

$T_{dc}(1,n)=2^2(2T_{dc}(1,n/8) + (n/4)) + 2n$

$T_{dc}(1,n)=2^3T_{dc}(1,n/2^3)+ 3n$

$T_{dc}(1,n)=2^kT_{dc}(1,n/2^k)+ kn$

Let $n=2^k$ $k=\log n$

$T_{dc}(1,n)=n\ T_{dc}(1,1) + n \log n$

$T_{dc}(1,n)=n + n \log n=n \log n$

$T_{dc}(1,n)=O(n \log n)$

Algorithm 2.4 divides the rows of matrix into two groups of rows, based upon the middle row of matrix, each group consists of m/2 rows. First group consists of rows from first row to middle row of matrix and second group consists of rows from middle+1 row to last row of matrix. Recursively all the groups of rows are selected and divided into sub groups of rows. If sub group consists of only one row, then that row is further divided into groups of columns by using DivideCols() Algorithm. At last all the sorted rows of matrix are combined.

```
Algorithm KNDivideRows(row1, row2, col1, col2) {
// row1 is the index of first row of matrix
//row2 is the index of last row of matrix
//col1 is the index of first column of each row of matrix
//col2 is the index of last column of each row of matrix
 if(row1<row2) { //if index of first row is smaller than index of last row of matrix
 midrow=(row1+row2)/2; // find index of middle row of matrix
 KNDivideRows(row1,midrow,col1,col2); // divide rows from first row to middle row of matrix
 KNDivideCols(row1,col1,col2); // divide columns from first column to last column of row of //matrix
 KNDivideRows(midrow+1,row2,col1,col2); // divide rows from middle+1 row to last row of //matrix
 KNDivideCols(row2,col1,col2);//divide columns from first column to last column of row2 of //matrix
 KNCombineRows(row1,row2,col1,col2); // combine sorted rows of matrix
 } }
```

Algorithm 2.4 Dividing UnOrdered Rows of a Matrix

Let $T_{dr}(m, n)$ is the time taken to divide m rows of given matrix of order mxn into two sub groups of rows, based upon the middle row of matrix. Each group of rows consists of m/2 rows, each row of a group consists of n columns. The time taken by Algorithm 2.4 is as follows:

$T_{dr}(m, n)=2T_{dr}(m/2, n) + T_{cr}$

$T_{dr}(m, n)=2T_{dr}(m/2, n) + mn$ (assume linear list as Linked List)

$T_{dr}(m, n)=2 (2T_{dr}(m/4, n) + (m/2)n) +mn$

$T_{dr}(m, n)=2^2 T_{dr}(m/4, n) + 2mn$

$T_{dr}(m, n)=2^2(2 T_{dr}(m/8, n) + (m/4)n)+ 2mn$

$T_{dr}(m, n)=2^3 T_{dr}(m/8, n) + 3mn$

$T_{dr}(m, n)=2^k T_{dr}(m/2^k, n) + kmn$

Let $m=2^k$ $k=\log m$

$T_{dr}(m, n)=m T_{dr}(1, n) + mn \log m$

$T_{dr}(m, n)=m T_{dc}(1, n) + mn \log m$ (since $T_{dr}(1, n)= T_{dc}(1, n)$)

$T_{dr}(m, n)=mn \log n + mn \log m$

$T_{dr}(m, n)=mn (\log n + \log m)=mn \log mn$

$T_{dr}(m, n)=O(mn \log mn)$

Implementation of Proposed Algorithm using C

The Proposed Algorithm is implemented by using C Programming Language, which is a robust, structure oriented programming language, which provides different concepts like functions, pointers, structures, arrays and so on to solve complex problems.

```
/* Implementation of KareemNaaz Matrix Divide and Sorting Algorithm*/
# include <stdio.h>
# include <conio.h>
int a[10][10]; // UnOrdered matrix with m rows and n columns per row
void swap(int *a,int *b) { // to swap elements of matrix with m rows and n columns per row
 int temp=*a;
 *a=*b;
 *b=temp;
 }
void CombineRows(int row1,int row2,int col1,int col2) {
//row1 is the index of first row of matrix
//row2 is the index of last row of matrix
//col1 is the index of first column of each row of matrix
//col2 is the index of last column of each row of matrix
 int temp[50],k=0; // a temporary linear array list to store elements of matrix
 int i,j,m;
 for(i=row1;i<=row2;i++) // outer loop repeats from first row to last row of matrix
 for(j=col1;j<=col2;j++) {// inner loop repeats from first column to last column of each row
 temp[++k]=a[i][j]; // placing elements of matrix into linear list temp
 for(m=k;m>1;m--) // sorting of elements of linear list temp in Increasing Order
 if(temp[m]<temp[m-1]) // if element at position m is smaller than element at position m-1 of temp
```

```
 swap(&temp[m],&temp[m-1]); // swap elements at positions m and m-1 of temp
 else
 break; }
 k=0;
 for(i=row1;i<=row2;i++) // outer loop repeats from first row to last row of matrix
 for(j=col1;j<=col2;j++) // inner loop repeats from first column to last column of each row
 a[i][j]=temp[++k]; // placing sorted elements of linear list temp into matrix of order mxn
}
void CombineCols(int row,int col1,int midcol,int col2) {
// row is the index of row of a matrix
// col1 is the index of first column of row of matrix
// col2 is the index of last column of row of matrix
//midcol is the index of middle column of row of matrix
int i=col1,j=midcol+1,k=col1;
int b[20];
while(i<=midcol && j<=col2) // index i repeats from first column to middle column of row and //index j repeats from middle+1 column to last column of row of matrix of order mxn
if(a[row][i]<a[row][j]) // if element at column i of row is smaller than element at column j of row
b[k++]=a[row][i++]; // place element at column i of row into linear list b
else
b[k++]=a[row][j++]; // place element at column j of row into linear list b
while(i<=midcol) // loop repeats from column i to middle column of row of matrix
b[k++]=a[row][i++]; // place element at column i of row into linear list b
while(j<=col2) // loop repeats from column j to last column of row of matrix
b[k++]=a[row][j++]; // place element at column j of row into linear list b
for(i=col1;i<=col2;i++) // loop repeats from first column to last column of row of matrix
a[row][i]=b[i]; // place sorted elements of linear list b into row of matrix
}
void DivideCols(int row,int col1,int col2) {
// row is the index of row of matrix of order mxn
//col1 is the index of first column of row of matrix
//col2 is the index of last column of row of matrix
 int midcol;
 if(col1<col2) {// if index of first column is smaller than index of last column of row
 midcol=(col1+col2)/2; // find index of middle column of row of matrix of order mxn
 DivideCols(row,col1,midcol); // divide columns of row from first column to middle column
 DivideCols(row,midcol+1,col2); // divide columns of row from middle+1 column to last column
 CombineCols(row,col1,midcol,col2); // combine sorted columns of row of matrix
 } }
void DivideRows(int row1,int row2,int col1,int col2) {
```

```
//row1 is the index of first row of matrix
//row2 is the index of last row of matrix
//col1 is the index of first column of each row of matrix
//col2 is the index of last column of each row of matrix
 int midrow; // index of middle row of matrix
 if(row1<row2) {// if index of first row is smaller than index of last row of matrix
 midrow=(row1+row2)/2;// find index of middle row of matrix
 DivideRows(row1,midrow,col1,col2); // divide rows of matrix from first row to middle row
 DivideCols(row1,col1,col2); // divide columns of row1 from first column to last column
 DivideRows(midrow+1,row2,col1,col2); // divide rows of matrix from middle+1 row to last row
 DivideCols(row2,col1,col2);//divide columns of row2 from first column to last column
 CombineRows(row1,row2,col1,col2);//combine sorted rows of matrix
 } }
void DisplayMatrix(int n) { // displaying elements of sorted matrix of order nxn
int i,j;
for(i=1;i<=n;i++) {
printf("\n");
for(j=1;j<=n;j++)
printf("%d ",a[i][j]);
} }
void main() {
 int n,i,j;
 clrscr();
 printf("\nImplementation of KareemNaaz Matrix Divide and Sorting Algorithm");
 printf("\mEnter n value (matrix of order nxn):\n"); // let us assume number of rows of matrix is //equal to number of columns per row of matrix
 scanf("%d",&n);
 printf("\nEnter %d elements of matrix:\n",n*n);
 for(i=1;i<=n;i++) // placing elements into matrix of order nxn
 for(j=1;j<=n;j++)
 scanf("%d",&a[i][j]);
 DivideRows(1,n,1,n); // sorting the elements of matrix of order nxn
 printf("\nAfter Sorting:");
 DisplayMatrix(n); // displaying elements of matrix of order nxn
 getch();
 }
```

Program 3.1 Implementation of KareemNaaz Matrix Divide and Sorting Algorithm [3]

Testing of Proposed Algorithm

Different Test cases are considered to test the result of Program 3.1, which takes a matrix of order 3x3 {{1 4 3} {7 8 6} {5 9 2}} as input and displays the sorted matrix of order 3x3 { {1 2 3} {4 5 6} {7 8 9}} as output. **Figure 3** shows the output screen shot of program 3.1.

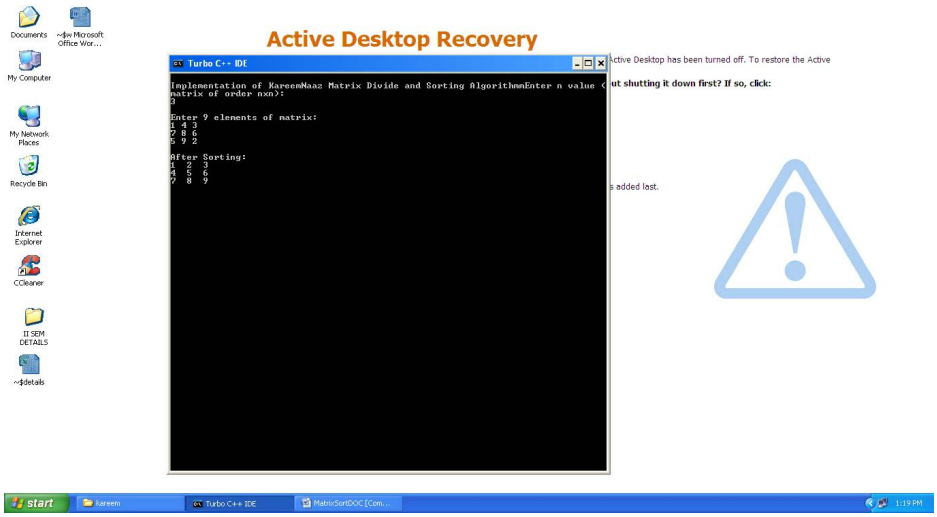

Figure 3. Screen shot of Program 3.1.

CONCLUSION

I conclude that the proposed KareemNaaz Matrix Divide and Sorting Algorithm take a matrix of order (mxn). It recursively divides the given matrix of order (mxn) into two groups of rows of matrix, based on the middle row. Each group of rows consists of m/2 rows of matrix. First group contains rows from first row to middle row of matrix and second group contains rows from middle+1 row to last row of matrix. Recursively groups of rows are selected and divided into sub groups of rows. If sub group of rows contains only one row, then it is divided into two groups of columns, based upon the middle column of row, each group of columns consists of n/2 columns. First group contains columns from first column to middle column of row, second group contains columns from middle+1 column to last column of row. Recursively groups of columns of row are selected and divided into sub groups of columns. If sub groups of columns contains single column then combine and sorting of those columns takes place. The proposed algorithm is mainly based upon Divide and Conquer strategy. Its overall time complexity is O(mn log mn) if linked list is used to combine sorted rows of matrix of order mxn.

REFERENCES

1. Aho AV and Jeffrey DU. Data Structures and Algorithms. AddisonWesley, Reading, Massachusetts; 1983.
2. Cormen TH, et al. Introduction to Algorithms. McGraw-Hill, New York; 1990.
3. Knuth DE. The Art of Computer Programming. Sorting and Searching. Addison-Wesley, Reading, Massachusetts; 1998.