

# **Lab Assignment :2**

## **CSN-261:Data structures Lab**

Submitted to: Dr Sudip Roy

*Mahak Bansal*

*Btech II year*

*18112039*

*Sub Batch:O2*

*Tools used : C(language),Linux(os),  
Github(csv),Doxygen(documentation),GDB(debugger)*

## **PROBLEM 1:**

In this Problem, you have to implement a simple transposition cipher, where this cipher encrypts and decrypts a sequence of characters by dividing the sequence into blocks of size  $n$ , where  $n$  is specified by the encryption key. If the input text has a length that is not a multiple of  $n$ , the last block is padded with null characters (`\0`).

In addition to  $n$ , the key also specifies two parameters  $a$  and  $b$ . For each block, the  $i$ -th output character, starting from 0 as usual, is set to the  $j$ -th input character, where  $j = (ai + b) \bmod n$ . For appropriate choices of  $a$  and  $b$ , this will reorder the characters in the block in a way that can be reversed by choosing a corresponding decryption key  $(n, a', b')$ .

For example, if  $n = 5$ ,  $a = 3$ , and  $b = 2$ , the string Hello, world! would be encrypted like this:

```
in: H e l l o , w o r l d ! \0 \0
i: 0 1 2 3 4 0
   1 2 3 4 0 1 2 3 4
j: 2 0 3 1 4 2 0 3 1 4 2 0 3 1 4
out: l H l e o w , o r ! l \0 d \o
```

## **Task to Perform**

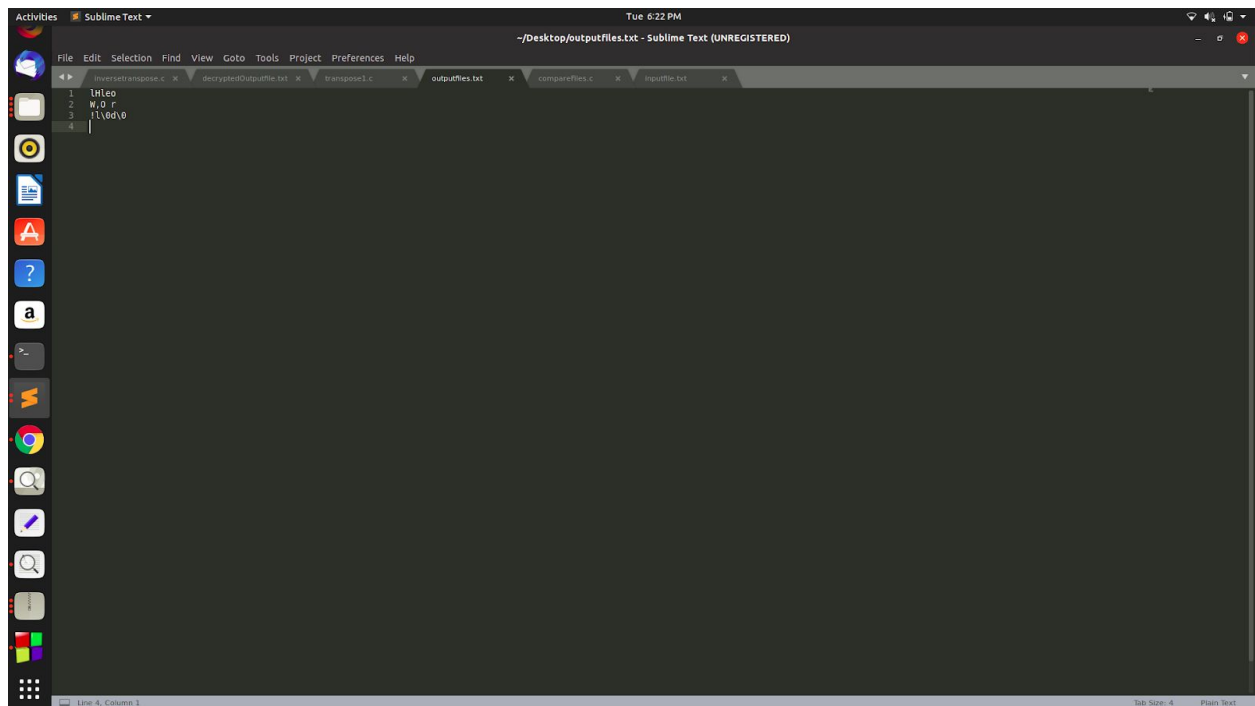
Write a program `transpose.c` that takes  $n$ ,  $a$ ,  $b$ , `inputfile.txt` in `argv[1]`, `argv[2]`, `argv[3]`, and `argv[4]`, respectively, applies the above encryption; and writes the result to `outputfile.txt`. Further, write a program `inverseTranspose.c` that decrypt the `outputfile.txt` and result in a new file named `decryptedOutputfile.txt`. Finally, write a program `compareFiles.c` to find the equivalence between the `inputfile.txt` and `decryptedOutputfile.txt` files.

You may assume that  $n$ ,  $a$ , and  $b$  are all small enough to fit into variables of type `int`. Your program should exit with a nonzero exit code if  $n$  is not at least 1 or if it is not given exactly four arguments, but you do not need to do anything to test for badly-formatted arguments. You should not make any other assumptions about the values of  $n$ ,  $a$ , or  $b$ ; for example, either of  $a$  or  $b$  could be zero or negative.

## **DATA STRUCTURES AND ALGORITHM:**

I have used basic file handling operations to read inputfile.txt and command line arguments in c. Simple function like atoi to convert char to int.

## **OUTPUTS :**



The screenshot shows a Sublime Text editor window titled "Sublime Text" with a menu bar (File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, Help) and a toolbar. The editor is open to a file named "outputfiles.txt" located at "~Desktop/outputfiles.txt". The file contains the following C code:

```
1 #include <stdio.h>
2 #include <string.h>
3 #include <stdlib.h>
4 #include <unistd.h>
```

The editor's status bar at the bottom indicates "Line 4, Column 1" and "Tab Size: 4, Plain Text". The left sidebar shows a file explorer with a list of files: "inputfile.txt", "outputfiles.txt", "comparefiles.c", "transpose1.c", "decryptedOutputfile.txt", and "inversetranspose.c". The "outputfiles.txt" file is selected and its content is displayed in the editor.

```
Activities Terminal Tue 6:22 PM mahak@nsipiron: ~/Desktop
File Edit View Search Terminal Help
mahak@nsipiron:~/Desktop$ gcc transpose1.c -o transpose1
mahak@nsipiron:~/Desktop$ time ./transpose1 5 3 2 inputfile.txt
real    0m0.003s
user    0m0.000s
sys     0m0.000s
mahak@nsipiron:~/Desktop$ gcc inversetranspose.c -o inversetranspose
mahak@nsipiron:~/Desktop$ time ./inversetranspose 5 3 2 outputfiles.txt
real    0m0.001s
user    0m0.001s
sys     0m0.000s
mahak@nsipiron:~/Desktop$ gcc comparefiles.c -o comparefiles
mahak@nsipiron:~/Desktop$ time ./comparefiles
total number of positions where the two files donot match are 0
real    0m0.003s
user    0m0.000s
sys     0m0.000s
mahak@nsipiron:~/Desktop$
```

## **CPU TIME:**

### **Transpose.c:**

```
real    0m0.002s
user    0m0.002s
sys     0m0.000s
```

### **Comparefiles.c**

```
real    0m0.003s
user    0m0.003s
sys     0m0.000s
```

### **Inversetranspose.c:**

```
real    0m0.001s
user    0m0.001s
sys     0m0.000s
```

## **Problem2: Medial axis transformation (MAT)**

A region can be represented either by its interior or by its boundary. Here we represent the region by its interior using one of the most common methods called image array. In this case we have a collection of pixels. Since the number of elements in the array can be quite large, the main objective is to reduce its size by aggregating equal-valued pixels.

A general approach is to treat the region as a quadtree, where the region is represented as a union of maximal non-overlapping square blocks whose sides are in power of 2. The quadtree can be generated by successive subdivision of the image array into four equal sized quadrants. If the sub-array does not consist entirely of 1s or entirely of 0s, it is then further subdivided into quadrants and sub-quadrants, etc.

### **Task to perform:**

Write a C program, MAT.c to represent any region (in image array representation), into its quadtree form.

Input:

Sample region is represented as  $n \times n$  array (as shown in Fig. 1 using  $6 \times 6$  matrix).

The format of the input file should be as follows:

the pixel values in the input file are separated by a single space and rows are separated by a newline character (refer to the sample L2\_P2\_inputsample.txt file shared in Piazza).

(Note: The  $6 \times 6$  region array should be mapped at the bottom-left corner of a  $8 \times 8$  binary array as shown in Fig. 2(b))

### **Output:**

1. Print the Maximal square array where it should be filled in the following order: top-right, top-left, bottom-right and bottom-left quadrant, this should be done recursively for all the sub-quadrants. All the cells within a maximal square block should be filled with its corresponding block number. For example, with respect to Fig. 2(c) maximal array should be represented as
2. Print the quadtree in the following manner, labels of leaf nodes, corresponding bit value and their level information (assuming the level of the root node to be 0), while traversing the quadtree in postorder. For example, in Fig. 2(d) the leaf node 3 having bit value 0 at level 2 and should be printed as (3,0,2).

## DATA STRUCTURES AND ALGORITHM USED

I have used structures to represent a point and a node. Quadtree is implemented using trees .Basic file handling is used to read the input file.

## OUTPUT

```

Activities Terminal Tue 3:34 PM mahak@nsprion: ~/Desktop
File Edit View Search Terminal Help
mahak@nsprion:~/Desktop$ gcc mat.c -o mat -lm
mahak@nsprion:~/Desktop$ time ./mat
enter your choice :how would you like to view the quadtree
1.in maximal array representation
2.in the form(label,bit value,level)
3. enter -1 to exit1
printing tree in maximal array form
1 1 1 1 2 2 3 3
1 1 1 1 2 2 3 3
1 1 1 1 4 4 5 5
1 1 1 1 4 4 5 5
6 6 7 8 13 13 14 14
6 6 9 10 13 13 14 14
11 11 12 12 15 16 19 19
11 11 12 12 17 18 19 19
enter your choice :how would you like to view the quadtree
1.in maximal array representation
2.in the form(label,bit value,level)
3. enter -1 to exit2
1 0 1
2 0 2
3 0 2
4 1 2
5 1 2
6 0 2
7 0 3
8 1 3
9 1 3
10 1 3
11 0 2
12 1 2
13 1 2
14 1 2
15 1 3
16 1 3
17 1 3
18 0 3
19 0 2
enter your choice :how would you like to view the quadtree
1.in maximal array representation
2.in the form(label,bit value,level)
3. enter -1 to exit-1
real    0m3.738s
user    0m0.001s
sys     0m0.003s
mahak@nsprion:~/Desktop$

```

**CPU TIME:**

```
real    0m3.738s
user    0m0.001s
sys     0m0.003s
```

