HACETTEPE UNIVERSITY COMPUTER SCIENCE ENGINEERING DEPARTMENT

BBM203 SOFTWARE LABORATORY I

PROGRAMMING ASSIGNMENT 1

Advisor: R.A. Alaettin UÇAN

Subject : Data Structures and Algorithms
Due Date : 04/11/2018 (23:59:59)

Student ID : 21228817
Name : Ali Utku ÜNLÜ
E-Mail : aliutkuunlu@gmail.com

1.Introduction

In this assignment this topic titles expected from us.

- Read-Write Files
- Arrays
- Matrices
- Recursion
- Dynamic Memory Allocation

In order to do that we have a simulation like this:

Imagine there is a place which has a hidden treasure. We can only find the treasure with the key. By using key we would know the direction that we must go.

The key comes from “keymatrix.txt” which shape is square. Keymatrix must have a size that odd numbers such as 3,5 or 7.

The place comes from “mapmatrix.txt” which shape is rectangular.

Arguments:

```
./findtreasure 18x18 3 mapmatrix.txt keymatrix.txt output.txt
```

--Size of mapmatrix will be given at command line argument with separated via”x” or “X”. This is the first argument of the program. Sizes will be determine by splitting this argument respect to x or X.

```
char* token = strtok(firstInput, "x|X");
while (token != NULL)
{
    if(token != "")
        strcpy(sizes[l],token);
    l++;
    cont++;
    token = strtok(NULL, "x|X");
}
rowSize = atoi(sizes[0]);
columnSize = atoi(sizes[1]);
```

--Size of keymatrix will be given at command line argument For example: “3” or “5”. This is the second argument of the program.

NOTE: Sizes of the mapmatrix must be divisible by keymatrix’s size.
“18%3 = 0”

-- mapmatrix is a 2D array we will create this array by using “mapmatrix.txt” file. This file path is third argument of the program.

--keymatrix is also 2D array we will create this array by using “keymatrix.txt” file. This file path is fourth argument of the program.

-- The results of each step that we done will be write to “output.txt” file which will be created on the program.

This file path is fifth argument of the program.

2.Solution

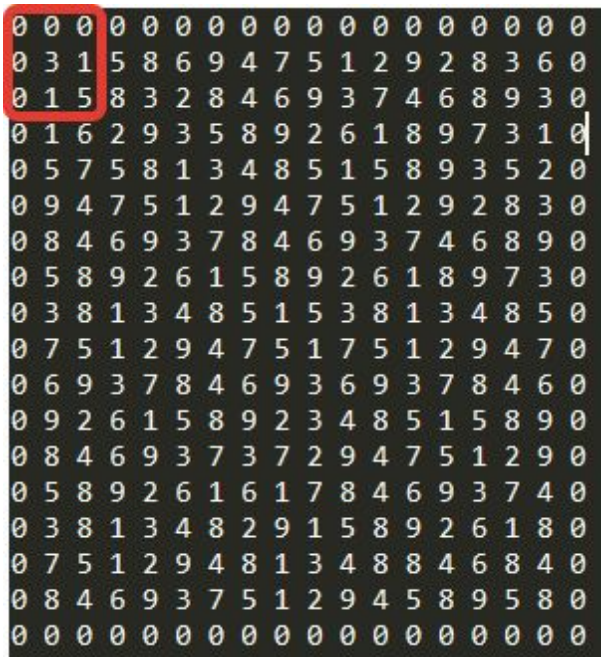
The treasure hunter will be start for searching at specific index of mapmatrix. This index will be determined by using keymap size.

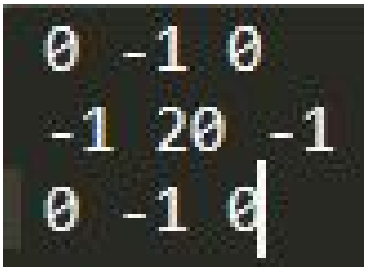
```
int keyMapSize = atoi(argv[2]);/*size of key map*/
int shift = (keyMapSize-1)/2;/*shift amount on the map while searching*/
rowIndex=shift;
columnIndex=shift;
```

If size is 3 than starting index must be 1. or if size is 5 than starting index must be 2.

The treasure hunter can go four ways. “up, down, right and left.”

S/he will learn the next way that s/he needs to go by multiplication of two matrices. One of the matrices is keymatrix. The other one is a submatrix of mapmatrix which size is equal to keymatrix. S/he need to multiply each value with the same index and add add all results to a single number. For beginning progress will be like this:





submatrix of beginning. keymatrix.

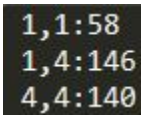
The result is: (0*0)+(0*-1)+(0*0)+(0*-1)+(3*20)+(1*-1)+(0*0)+(1*-1)+(5*0)=58.

This result tell her/him to next way. S/he take mod five the coming number which is “58” in this case. The result of mod operations are:

0: Treasure is found.
1: Go up
2: Go down
3: Go right
4: Go left

**If there is no way to go on the matrix s/he have to go opposite of chosen direction.

After each step program must write the current indexes and result of multiply.



Example output.

3. Functions

- Read_File Function:

It takes two parameters, one of them is our first command line argument and the other one is an empty two dimensional char array. It reads the argument and pass to the array.

Prototype is:
void Read_File(char* s, char** matrix); s is our command line argument which is given in the command line. Matrix is our 2D dynamic array which is stores the input data

Using:
Read_File(argv[3], tempTreasureMap);

- split Function:

This function split the given 2D array according to white spaces and fill the given integer array. It takes three parameters one of them our input matrix, next one is our empty integer array and the last one is row number of the mapmatrix.

Prototype is:
void split(char** matrix, int** result, int lineCount);

Using:
split(tempTreasureMap, map, rowSize);

- search Function:

  This function does the calculations and choose the next way recursively.

  It takes 9 parameters.
  --First one is mapmatrix type of int array.
  --Second one is key matrix type of int array.
  --Third one is current position of row index type of int pointer.
  --Fourth one is current position of column index type of int pointer.
  --Fifth one is row size of map matrix type of int.
  --Sixth one is column size of map matrix type of int.
  --Seventh one is size of keymap matrix type of int.
  --Eighth one is shift amount type of int.
  --Ninth one is output file type of FILE pointer.

  Prototype is:
  void search(int** map, int** key, int* rowIndex, int* columnIndex, int rowSize, int columnSize, int keyMapSize, int shift, FILE* output);

  Using:
  search(map, key, &rowIndex, &columnIndex, rowSize, columnSize, keyMapSize, shift, output);

- goUp Function:

  This function modified the indexes according to chosen way. In that case this function increment the row index as keymap size. If there is no way then decrement the row index.

  It takes 2 parameters.
  --First one is current position row index type of int pointer.
  --Second one is size of keymatrix type of int.

  Prototype is:
  void goUp(int* rowIndex, int keyMapSize);

  Using:
  goUp(rowIndex, keyMapSize);

- goDown Function:

This function modified the indexes according to chosen way. In that case this function decrement the row index as keymap size.If there is no way then increment the row index.

It takes 4 parameters.
--First one is current position row index type of int pointer.
--Second one is size of keymatrix type of int.
--Third one is rowsize of mapmatrix type of int.
--fourth one is shift amount type of int.

Prototype is:
void goDown(int* rowIndex, int keyMapSize, int rowSize, int shift);

Using:
goDown(rowIndex, keyMapSize, rowSize, shift);

- goLeft Function:

This function modified the indexes according to chosen way. In that case this function decrement the column index as keymap size. If there is no way then increment the column index.

It takes 2 parameters.
--First one is current position column index type of int pointer.
--Second one is size of keymatrix type of int.

Prototype is:
void goLeft(int* columnIndex, int keyMapSize);

Using:
goLeft(columnIndex, keyMapSize);

- goRight Function:

This function modified the indexes according to chosen way. In that case this function increment the column index as keymap size.If there is no way then decrement the column index.

It takes 4 parameters.
--First one is current position column index type of int pointer.
--Second one is size of keymatrix type of int.
--Third one is columnsize of mapmatrix type of int.
--fourth one is shift amount type of int.

Prototype is:
void goRight(int* columnIndex, int keyMapSize, int columnSize, int shift);

Using:
goRight(columnIndex, keyMapSize, columnSize, shift);