CSE 331 HW2 REPORT

EMRE SEZER

1901042640

EXPLANATION OF ASSEMBLY CODE:

My program, reads all of the items in the txt file and stores them in the buffer. Then, it checks each character 1 by 1. Until it reads ‘,’ character, stores the numbers at the stack after substracting 48 from number (48 is ascii code of ‘0’). When it reads ‘,’ it multiplies each number in the stack with correct multiply of 10 (This part is for reading multi digit numbers). Result is stored in the actv_arr which is the array that holds elements that operations will be executed on.

My algorithm works with 3 nested loops. First loop is for determining which element to start from looking for sequences. Third loop iterates and looks for bigger numbers than sub_arr’s last element (sub_arr is the array that holds temporary sequences). In the second loop, after each third loop checks if a sequence is found with bigger length than current sequence’s length. If found, new sequence is the array that recently found. Finally it writes elements in fin_arr(Array that holds the sequence with the biggest length) to the output.txt. I also added descriptions to the asm file.

PSEUDO CODE :

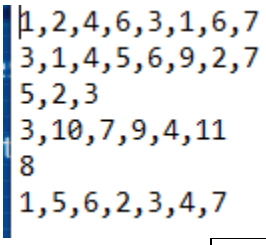
```
const int MAX_LENGTH  = 6;  // Max array size

int sub_arr[MAX_LENGTH];    // Holds integers of each line in input file

int sub_arr[MAX_LENGTH];    // Holds temporary sequences

int sub_arr_len = 0;        // sub_arr’s length

int fin_arr[MAX_LENGTH];    //  Holds biggest sequence for array, getting updated if bigger sequence if found

int fin_arr_len = 0;        // fin_arr’s length

Open file ”input.txt” for reading        // Opens input file

Open file ”output.txt” for writing       // Opens output file

buffer = read(test.txt)                  // Reads input file and stores entire file in an array

While(file pointer hasn’t reached EOF)  // Loop continues until it reaches end of the file

{

        While(current character of buffer != ‘\n’) Store integers between ‘,’ ’s to arr     /*

        Stores each integer in the currently reading line to arr                          */

        Call part3(arr)                  // Calls part3 function in order to find biggest sequence for each line in input file

        Write fin_arr to “output.txt”            // Writes biggest sequence to output file for each line in input file

        Write fin_arr_len to “output.txt”        // Writes biggest sequence’s size to output file for each line in input file

        Close “test.txt”                         // Closes input file

        Close “output.txt”                       // Closes output file

}


void part3(int arr [])
```

```
{
        for(int i = 0; i < MAX_LENGTH; i++)
        {
                for(int l = 0; l < MAX_LENGTH; l++)       sub_arr[l] = 0;       // Resets sub_arr
                sub_arr_len  = 0;                                               // Sets sub_arr' length to 0
                sub _arr[sub_arr_len] = arr[i];                         // Adds first element of arr to sub_arr's first element
                sub_arr_len++;                                                  // Increases sub_arr's length by 1
                for(int j = i + 1; j < MAX_LENGTH; j
                {
                        for(int l = 0; l < sub_arr_len; l++)        sub_arr[l] = 0;       // Resets sub_arr
                        sub_arr_len  = 0;                                               // Sets sub_arr' length to 0
                        sub_arr[sub_res_len] = arr[i];              // Adds first element of arr to sub_arr's first element
                        sub_len++;                                  // Increases sub_arr's length by 1
                        if(arr[j] < sub_arr[0])             continue;   // If current element is smaller than sub_arr's 1st skips
                        for(int k = j; k < MAX_LENGTH; k++) ++)             /* Iterates through arr and looks for a bigger
                        element than sub_arr's last element                 */
                        {
                                if(arr[k] > sub_arr[sub_arr_len - 1])               // If a bigger element is found
                                {
                                      sub_arr[sub_arr_len] = arr[k];        // Adds that element to the end of the array
                                            sub_arr_len++;                          // Increases sub_arr's length by 1
                                }
                        }
                        if(sub_len > fin_len)                   // If current sub_arr' size is bigger than current fin_arr's size
                        {
                                fin_arr_len = sub_arr _len;                 // Sets fin_arr'size to sub_arr's size
                                for(int l = 0; l < MAX_LENGTH; l++) fin_arr[l] = sub_arr[l];    // Replaces sub_arr to fin_arr
                        }
                }
        }
```

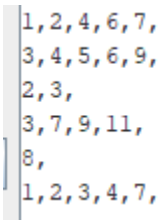
My algorithm works with 3 nested loops. First loop is for determining which element to start from looking for sequences. Third loop iterates and looks for bigger numbers than sub_arr's last element (sub_arr is the array that holds temporary sequences). In the second loop, after each third loop checks if a sequence is found with bigger length than current sequence's length. If found, new sequence is the array that recently found.

Time complexity of my program is O(n^4). There are 3 nested loops and these loops will work for n lines of input. Space complexity is O(1) since . I set space for memory a constant numbers.

TEST CASES AND SCREENSHOTS:



(Input screenshots)



(Result screenshots)

INPUT	OUTPUT
1,2,4,6,3,1,6,7	1,2,4,6,7
3,1,4,5,6,9,2,7	3,4,5,6,9
5,2,3	2,3
3,10,7,9,4,11	3,7,9,11
8	8
1,5,6,2,3,4,7	1,2,3,4,7

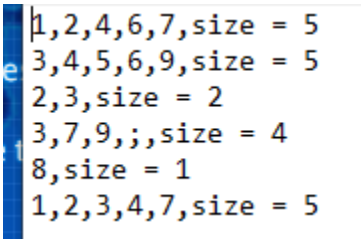
I get the results as expected. My program works fine for those cases.

MISSING PARTS:

Can't write multi digit numbers to the output file. As you can see

On the right side: At 4'th line it failed to write 11 to the file.

From table at the top there should have been 11 instead of ';' .



BONUS PARTS:

My program works with multi digit numbers and shows inner results.