Notes on Assignment 1

CSCI 2270: Data Structures
Section: 202
TA: Sanskar Katiyar

Problem 1: Insert into Sorted Array

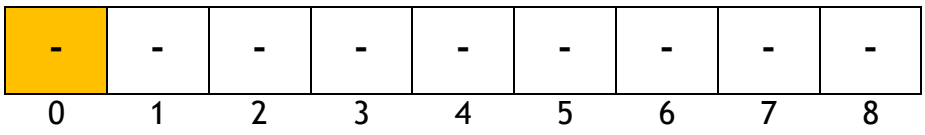
The primary objective is to write the `insertIntoSortedArray` function correctly. Below I have provided you an example run with some interesting observations that will give you hints through guiding questions.

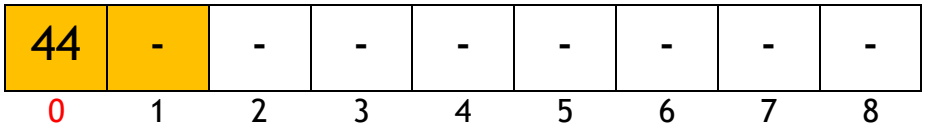
The idea is to insert an element into a sorted array, such that the array remains sorted after the insertion of the element. Let's look at an example run.

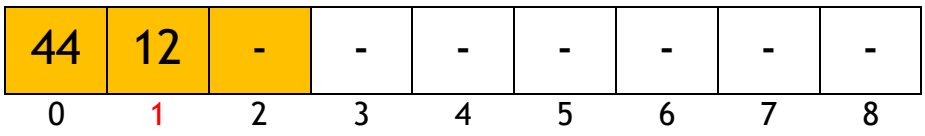
Example Run

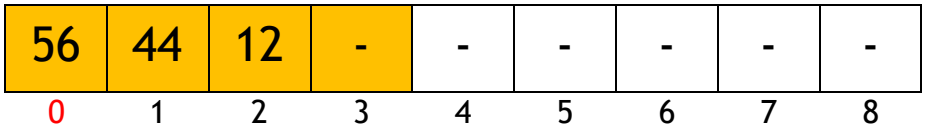
Dashes (-) indicate garbage values in the array.

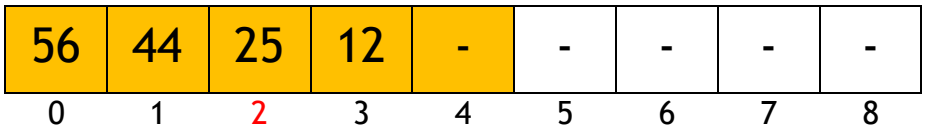
```
numEntries = insertIntoSortedArray(myArray, numEntries, 44);
numEntries = insertIntoSortedArray(myArray, numEntries, 12);
numEntries = insertIntoSortedArray(myArray, numEntries, 56);
numEntries = insertIntoSortedArray(myArray, numEntries, 25);
```











The index highlighted in red is the index where the element needs to go to. The highlighted cells are the index range `[0, numEntries]`.

OBSERVATIONS

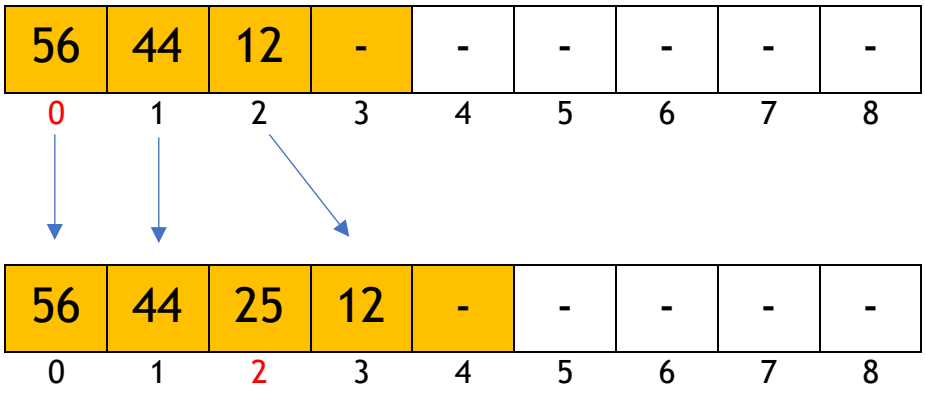
numEntries

This variable doesn't just help you determine the number of elements in the array but also the extent of your sorted array (the portion of the array that you need to work with).

So, you don't need to worry about the complete array (`n = 9`, here) but only the range of indices `[0, numEntries]`. This includes the extra space for inserting the new element.

Index

Take a closer look around the index where the `newValue` gets inserted (highlighted in red above).



Notice that the elements towards the left of the inserted element stay at the same index `[i]` while the elements to the right are now at index `[i+1]` where `i` is the index of an element before the current function call.

APPROACH(ES)

There are a couple of ways you can approach this problem:

1) **Insert first, Sort second**

   You can insert the element towards at end of the **sorted** array and then sort the entire array.

   The algorithm to sort the array is of your choice. However, I would request you to look at the **Insertion Sort** algorithm. Go through the algorithm and see if you can exploit the principle that this algorithm uses.

   Note that even though this is an acceptable solution, this is not an efficient solution. The reason being: this approach makes you sort the entire array just for adding one element. The sorting algorithm would make multiple passes over the array when this can be solved in just one-pass.

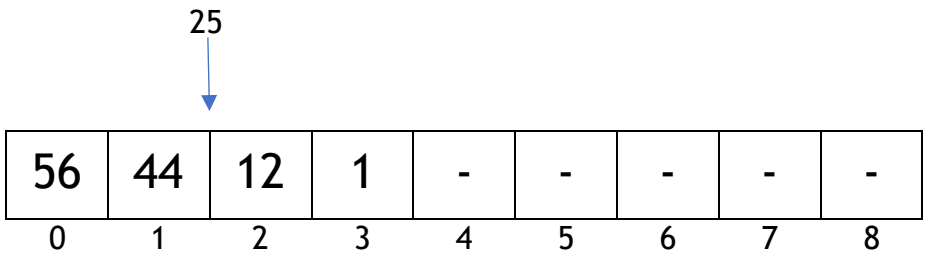
2) **Sort *and* Insert**

   This is the one-pass algorithm I was hinting at during the recitation. This approach is heavily inspired from the **Insertion Sort** algorithm.

   The key questions here are:

   a. **At which index to insert the element newValue?**

      Given the current state of the array (as below) and `newValue = 25`, we see that 25 should go between 44 and 12, but since 44 > 25, thus 25 should go at `index = 2` and the numbers at `index > 2` should shift by 1 unit to the right.



But how do we find this index computationally?

Note that you know the index range of the sorted array `[0, numEntries] = [0, 4]` here. In this sub-array you will need to compare each element with `newValue` and find the index which would satisfy the condition for descending order if `newValue` is inserted there.

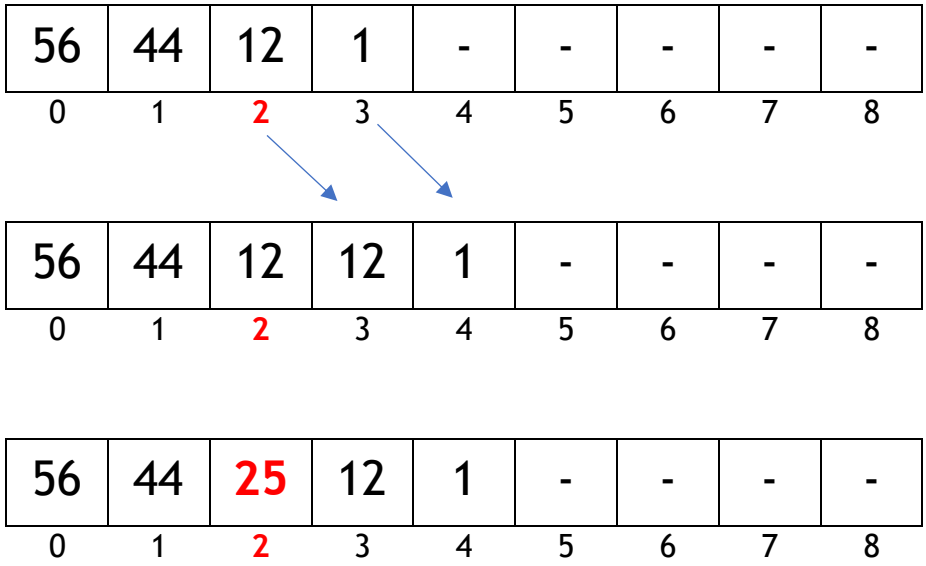
b. How to make room for the new element?

From above, let's say we now know the index where we need to insert the `newValue = 25`, which is, `index = 2`. But we ***can't*** do:

`myArray[2] = newValue; //` **Why not?**

At the same time, we know that the values at index < 2 don't need to worry since they are already occupying their correct place. The only way we can make space is by shifting the elements at index >=2 to the right by 1 unit. So, how do we do that? And more importantly**, in which order**?

You basically need something like this:



Finally, do you think you can combine both of (a) and (b) above?

COMMON MISTAKES

- Not reading the filename from command line arguments. You will need to get the filename from `argv[]`.

- Not updating/returning numEntries variable after adding an element into the array.

- Not typecasting the string read from the file. Need to use `stof()` to typecast to a float

- Printing within the function `insertIntoSortedArray()`. Don't print anything within the function.

- Don't read the file within the `insertIntoSortedArray` function. This should be done in `main()`.

- When testing the complete program in coderunner, make sure you are not printing extra commas or newlines. Coderunner is finnicky about its format.

Problem 2: Parks and Structures

This is a very straightforward question. You just need to be familiar with the syntax for structures and accessing the elements of a structure. (Refer to Recitation 2 slides).

Remember: structure just defines a new data type.

Another thing you need to be familiar, perhaps, is reading a file line by line. In order to split the line through commas, you should look at the `getline` function.

COMMON MISTAKES

- Need to use `argv[]` to get command line arguments. Do appropriate typecasting for the upper and lower bounds using the `stoi()` or `stof()` function.

- Check the Assignment writeup for the output format (Don't need to replace underscores with spaces).

- Use `-Wall` flag during compilation to check the warnings for any uninitialized variables that might end up accessing a bad memory location resulting in a segmentation fault. (Check Maciej's post on Moodle)