Sorting and Merging Arrays

Abstract

In this project you will implement

1. An array sorting problem, using a specified pattern
2. Merging of sorted arrays, using exhaustive search algorithm or Heap Sort

Algorithm 1: Pattern Sorting

Assume that you are given two arrays of items. The first array contains series of items (sometimes integers) that may appear in the second array. The second array represents a desired pattern or order of integers in the first array.
Example: If a second array is in the following format [a, b, c], it implies that the first array should be sorted in order of a items, followed by b and the c items [a, a, ..., a, b, b, . . . , b, c, c, . . . , c]

Sample input:
array 1 = [1, 0, -1, 0, 0, 1,1, -1, 0, 1]
array 2 = [0, 1, -1]

Sample output: =[0, 0, 0, 0, 1, 1, 1, 1, -1, -1]

To Do:

a. Develop a complete and clear pseudocode for an algorithm to solve this problem. Your algorithm should sort the given array in place. As discussed in class, in-place sorting mutates the given array in the output, ensuring the creation of no new storage space. The desired sorting may be done in any order, not necessarily in ascending or descending order.
b. Mathematically analyze your pseudocode and state the efficiency class
c. Implement your algorithm in either Python or C++
d. Using the given sample files (in3A.txt), print resulting lists for the 3 input arrays. Include them in your PDF report

Algorithm 2: Merging Sorted Arrays

You are provided with grades of students from various sections. The grading system allows the use of negative points. The grades appear to be sorted: the first item in each array is the least element in the respective array. You decide to merge the various lists into a single sorted array. Sample input arrays and the desired output are presented below:

Sample input:

```
all_lists =[ [2, 5, 9, 21],
             [-1, 0, 2],
             [-10, 81, 121]
             [4, 6, 12, 20, 150] ]
```

Sample output:

```
[ -10, -1, 0, 1, 2, 2, 4, 5, 6, 9, 12, 20, 21, 81, 121, 150]
```

There are different ways of merging the given lists. Given that the first element of each array is the smallest integer, you can build a list of smallest items. Pick the smallest out of the list of all smallest items. This will become the first item of the merged sorted list. You may then proceed to check all items in parent array. Another method of achieving this is through the use of Heap Sort. A Min heap can be used to store the smallest elements at any point in your algorithm.

To Do:

a. Develop a complete and clear pseudocode to merge the arrays
a. Mathematically analyze your pseudocode and state the efficiency class
b. Implement your algorithm in either Python or C++
c. Using the given sample files (in3B.txt), print resulting merged lists for the 3 input arrays. Include them in your PDF report

To Do

1. Develop the pseudocodes and implement your algorithms in Python or C++
2. Produce a readme file, describing how to execute your program.
3. Submit all files, including your codes separately. (Do not zip).

Grading Rubric

The suggested grading rubric is below.

Algorithm 1 and 2 (50 each)

a. Clear and complete Pseudocode = 10 points
b. Mathematical analysis and correct Big *O* efficiency class = 5 points

c. Successful compilation of codes= 20 points
d. Produces accurate results (3 output lists) = 15 points