CS3610 Project 4

Due: Tuesday Nov. 14, 11:59 pm

In this project, you will write a program to simulate an external sorting algorithm. External sorting is used in scenarios in which the size of the dataset to be sorted exceeds the capacity of the main memory. In the first stage of external sorting, the dataset is divided into a set of smaller, unordered sublists. Each sublist is then transferred from disk to memory and sorted individually using an efficient internal sorting algorithm, such as quicksort. Finally, the sorted sublists are efficiently merged together on disk into a single sorted dataset.

To begin simulating external sort, you will first read an unsorted list of integers stored in file into an nxm matrix, where n represents the number of unordered sublists and m is the number of integers comprising each sublist. Next, you will sort each of the n sublists individually using quicksort. To ensure that your quicksort runs in $O(mlgm)$ time in the worst case, you must implement a partition function with the pivot element set as the median of the first, middle, and last elements of the sublist given as input. In the final stage, you will merge together the n sorted sublists into a single output list using a multiway merge that runs in $O(mlgn)$ time. To achieve a $O(mlgn)$ runtime, first create a min heap from the first elements of each sorted sublist. (Consequently, you will have a size n min heap.) Next, extract the min element from the min heap and place it into a 1D, sorted output list. Return back to the min heap and replace the previously extracted min element with its successor from their originating, sorted sublist. If the min element was the largest in its sublist, move directly on to extracting the next min element in the min heap. Repeat this process until all sublists have been merged into the 1D, sorted output list. Rather than implementing a min heap from scratch, you may instead use the standard heap functions in the C++ algorithm library (http://en.cppreference.com/w/cpp/algorithm) or you may use the priority queue from the C++ queue library (http://en.cppreference.com/w/cpp/container/priority_queue).

You have been provided a driver program that reads in the test input, makes calls to the quicksort and multiway merge functions, and outputs the final sorted results. Do not modify the main function and do not rewrite any of the existing function headers. You may however implement extra helper functions or data structs if you feel it necessary.

Input

Input is read from the keyboard. The first line of input is the number of unordered sublists n. The second line is the number of integers m comprising each unordered sublist. The third line is an unordered, space-delimited list of $n * m$ integers.

Output

Output the externally sorted list of integers using space as a delimiter.

Sample Test Case

Use input redirection to redirect commands written in a file to the standard input, e.g.
`$ ./a.out < input1.dat`.

Input 1

```
4
5
100 28 83 22 3 4 1 0 222 425 42 49 599 58 79 934 41 23 444 422
```

Output 1

```
0 1 3 4 22 23 28 41 42 49 58 79 83 100 222 422 425 444 599 934
```

Turn In

Email your source code to yy471014@ohio.edu with the subject "CS3610 Project 4".
If you have multiple files, package them into a zip file.

Grading

Total: 100 pts.

- **10**/100 - Code style, commenting, general readability.

- **05**/100 - Compiles.

- **05**/100 - Follows provided input and output format.

- **80**/100 - Successfully implemented external sort.