Bookmarks

- How To?
- Week 1
- Week 2
- Week 3
  - Sorting And Search Algorithms
  - 3rd Week Problems
    Laboratory due Nov 20, 2016 at 21:00 UTC
- Week 4
- Week 5

Sorting

Bookmark this page

Sorting

2.0 points possible (graded)

Input file:	sort.in
Output file:	sort.out
Time limit:	2 seconds
Memory limit:	256 megabytes

Given a sequence of integer numbers. Your task is to sort it in a non-decreasing order using the mergesort algorithm.

To persuade us that you really use the mergesort algorithm, we ask you, after performing a merge of a certain interval, to print the endpoint indices and endpoint values of this interval.

Input
In the first line of the input file there is an integer n ($1 \leq n \leq 100\,000$), the number of elements in the sequence. The second line contains the sequence itself: n integer numbers not exceeding 10^9 by their absolute value.

Output
The output file consists of several lines.

In the **last line** of the output file print the sequence after sorting. Separate the numbers by single white spaces.

All the previous lines should describe the results of merges, one per line. Every such line should contain four numbers: l_f l_l V_f V_l, where l_f is the starting index of the just-merged interval, l_l is its ending index, V_f is the value of the first element of the interval, V_l is the value of the last element.

Indices start with one, that is, $1 \leq l_f \leq l_l \leq n$. **Please print the result of merges for the interval of size 1 as well**.

If you print l_f l_l X Y , this means you have just merged a subsequence which corresponds to the $[l_f;l_l]$ subsequence of the original sequence. In your particular implementation, the real indices for the beginning and ending of this subsequence may vary, but we nevertheless ask you to use this numbering scheme.

The merge descriptions can go in an arbitrary order, not necessary coinciding with the order they are performed. However, to improve performance, we recommend to print these descriptions as soon as possible, not storing them in memory. This is the reason why we ask to print the resulting array at the very end.

The correctness of the sorting scenario which you printed will be performed by a special checker program. Every correct mergesort, which splits a subsequence into two smaller subsequences (not necessary equal!), will be accepted if it manages to perform sorting within time and memory limits.

Note that every correct output will have 2n − 1 descriptions of merges, and for all $1 \leq i \leq n$ there will be a record i i a_i a_i, where a_i is the i-th element of the initial sequence.

Example

sort.in	sort.out
10 1 8 2 1 4 7 3 2 3 6	1 1 1 1 2 2 8 8 1 2 1 8 3 3 2 2 4 4 1 1 3 4 1 2 1 4 1 8 5 5 4 4 6 6 7 7 5 6 4 7 1 6 1 8 7 7 3 3 8 8 2 2 7 8 2 3 9 9 3 3 10 10 6 6 9 10 3 6 7 10 2 6 1 10 1 8 1 1 2 2 3 3 4 6 7 8