ETH Zurich
Institut für Theoretische Informatik
Prof. Dr. Angelika Steger

HS 2015
Exercise Sheet 0

Algorithms Lab

Exercise 1 – *Merge Complexity*

As it sometimes happens in the life of a computer programmer, you find yourself with a large number of sorted lists that need to be merged into a single sorted list.

The first solution you think of is to use the handy `std::merge()` function from the STL. It takes two sorted lists, and outputs one merged sorted list. You quickly realize that there is a better way than to keep on merging two lists until you have just one left. However before you think of that you notice that under the header "Complexity" the documentation for `std::merge()` says: "At most, performs length1 + length2 -1 assignments and comparisons, where length1 and length2 are the lengths of the input sequences."

Thinking about this a little more you realize that in your original solution the merge order would have mattered a lot in terms of amount of worst case assignments and comparisons (cost for short). If for example you have lists of length 1, 2 and 4. Then you could on one hand merge the first two with cost 1+2-1=2 then, the new list with the last one for 3+4-1 = 6. In total you do at most 2 + 6 = 8 assignments and comparisons. If on the other hand you merge the two longer ones first you get cost 2+4-1=5 for the first operation and 1+6-1=6 for the second, i.e. cost 11 in total.

Given m such lists of lengths $\ell_1, \ldots, \ell_m$, what is the minimum worst case cost to merge them all into one using `std::merge()`? Write a program to compute the answer.

Input The first line of input contain $n \leq 100$, the number of test cases you have to process.

Then n lines follow. The first number on each line is m, $0 \leq m \leq 100'000$. Then m integers $\ell_1, \ldots, \ell_m$ follow on the same line, all separated by one space. The lengths are at least 1 and at most 1'000.

Output For each test case output one line containing the minimum total worst case cost to merge all lists. If the input contains only one list, the cost is 0.

Sample input	**Sample output**
3 3 1 2 4 4 5 4 5 4 1 1	8 33 0