Mustafa Haluk AYDIN 2011400327 haluk.aydin@boun.edu.tr

Firstly, I took the input and divided into small parts which are lately written in temp files named as in1, in2, in 3… Before the writing, I stored and sorted the small inputs with using make_heap(arr1[],arr1[]+int) in double array arr1. make_heap() makes the array heap. And it takes the max value and sends it to the arr1 [first].

For merging, I use an algorithm that looks the both of the temp files first element and takes the smaller one and writes it in new temp file which is “intemp”. Right after writing this input it takes new input from the temp file which had the smaller. It does that until there is no element in both temp files. Then it erases two temp files and create new temp file which is going to be read later. After this, it takes new two temp files and so on…

I have a int variable named as ”xx” which determines how many elements a temp file has. For first sorting I have a number of total elements named num. Assume xx is n. I divide it with n. I have (n/xx)+1 files. Each files makes n times make_heap() whose complexity 3 *interval between first and last. Total complexity for first sorting of all elements in one file is 3*xx*(xx-1)/2. For merging, the total elements are processed log(n/xx) times. Total compexity is 3*xx*(xx-1)/2 * ((n/xx)+1) + n*log(n/xx). Nearly 3/2*n*xx + n*logn - n*logxx . Because of n>>xx and n>>constants, we can say n*logn. O(nlogn).

If we consider extra external memory, there is a file whose name is” intemp”. Its size changes with size of temp files it gets while merging. At the end it has all elements. So extra external memory usage is O(n)