# CS F211
# Data Structures and Algorithms
# Assignment - 6
## Sorting and Heaps

Allowed Language: **C**

February 14, 2024

## General Tips

- Try to use functions as much as possible in your code. Functions increase reusability and the pass-by-value feature provides a significant help sometimes. Modularizing your code also helps you to debug efficiently.
- Use `scanf` to read characters/strings from STDIN. Avoid using `getchar`, `getc` or `gets`. Try to read up about character suppression in `scanf` as it will be very helpful in some of the problems.
- Use `printf` instead of `putc`, `putchar` or `puts` to print character/string output on STDOUT.
- Indent your code appropriately and use proper variable names. These increase readability and writability of the code. Also, Use comments wherever necessary.
- Use a proper IDEs like Sublime Text or VSCode as they help to run and test your code on multiple test-cases easily.
- **Note:** Kindly try to do all of these questions by yourself at least once. Spend some time thinking about it, or trying to code it instead of directly asking help of your friends or searching it up online. This helps you understand the question, allowing you to solve further questions which are not in the scope of this Assignment yourself.

# A: Merge Sort

Merge sort is one of the most popular sorting algorithms. It is a very efficient algorithm and it's time complexity is proven to be the best case time complexity for comparison-based sorting algorithms as you might have seen in class. We want you to implement merge sort.

Given an array of size $n$, you have to sort it in time complexity $O(n \log n)$ using only merge sort algorithm.

## Input

The first line contains one positive integer $n$ ($1 \leq n \leq 10^5$), the size of array $a$.
The second line contains $n$ integers $a_1, a_2, \ldots, a_n$ ($-10^9 \leq a_i \leq 10^9$).

## Output

The only line of output should contain $n$ integers of the sorted array $a$.

---

input
4
3 4 5 2

output
2 3 4 5

---

input
2
6 4

output
4 6

---

input
9
8 5 2 6 2 1 0 0 7

output
0 0 1 2 2 5 6 7 8

---

# B: Quick Sort

Quick sort is one of the most popular sorting algorithms. Worst-case time complexity of quick sort might be $O(n^2)$ but on average its as good as merge sort. The best part about quick sort is that, it doesn't use extra space complexity other than the recursion stack. The C++ inbuilt sort function uses a hybrid of quick sort, heap sort and insertion sort to provide a very fast and efficient sorting algorithm which runs faster than merge sort in practice. We want you to implement quick sort.

Given an array of size $n$, you have to sort it using only quick sort algorithm.

## Input

The first line contains one positive integer $n$ ($1 \leq n \leq 2 * 10^3$), the size of array $a$.
The second line contains $n$ integers $a_1, a_2, \ldots, a_n$ ($-10^9 \leq a_i \leq 10^9$).

## Output

The only line of output should contain $n$ integers of the sorted array $a$.

---

input
4
3 4 5 2

output
2 3 4 5

---

input
2
6 4

output
4 6

---

input
9
8 5 2 6 2 1 0 0 7

output
0 0 1 2 2 5 6 7 8

---

# C: Heap Sort

Heaps are the most efficient way to implement priority queues. Heap sort is a very efficient in-place sorting algorithm which can be thought of as a version of selection sort, where you are selecting the max element in an efficient way. Please use Heapify to turn the given array into a heap and then keep extracting the max element into the same array. The C++ inbuilt sort function uses a hybrid of quick sort, heap sort and insertion sort to provide a very fast and efficient sorting algorithm which runs faster than merge sort in practice. We want you to implement heap sort.

Given an array of size $n$, you have to sort it in time complexity $O(n \log n)$ using only heap sort algorithm (Only the exact algorithm will be tolerated).

## Input

The first line contains one positive integer $n$ ($1 \leq n \leq 10^5$), the size of array $a$.
The second line contains $n$ integers $a_1, a_2, \ldots, a_n$ ($-10^9 \leq a_i \leq 10^9$).

## Output

The only line of output should contain $n$ integers of the sorted array $a$.

---

input
4
3 4 5 2

output
2 3 4 5

---

input
2
6 4

output
4 6

---

input
9
8 5 2 6 2 1 0 0 7

output
0 0 1 2 2 5 6 7 8

---

# D:AA Sort

Although merge sort runs in $O(n \log n)$ worst-case time, and insertion sort runs in $O(n^2)$ worst-case time, the constant factors in insertion sort make it faster for small $n$. Thus, it makes sense to use insertion sort within merge sort when subproblems become sufficiently small. Consider a modification to merge sort in which $n/k$ sublists of length $k$ are sorted using insertion sort and then merged using the standard merging mechanism, if the value of $k$ chosen is good this can be very efficient and can even achieve the same asymptotic time complexity as merge sort.

Given an array of size $n$, you have to sort the array in time complexity $O(nk + n \log (n/k))$ using only DAA sort algorithm. (Kira and Nom have done the TC analysis in DAA, you don't have to worry about it).

## Input

The first line contains two positive integers $n$ and $k$ ($1 \leq n \leq 10^5, 1 \leq k \leq 16$), the size of array $a$ and the value $k$ mentioned in the algorithm.
The second line contains $n$ integers $a_1, a_2, \ldots, a_n$ ($-10^9 \leq a_i \leq 10^9$).

## Output

The only line of output should contain $n$ integers of the sorted array $a$.

---

```
input
20 5
20 15 27 28 20 26 15 23 22 18 11 3 10 5 21 21 5 3 11 21

output
3 3 5 5 10 11 11 15 15 18 20 20 21 21 21 22 23 26 27 28
```

---

```
input
7 1
3456 96452 31846 48843 1234678 124657 85678

output
3456 31846 48843 85678 96452 124657 1234678
```

---

# E: Nomki sort

As you probably know by now, the C++ inbuilt sort function uses a hybrid of quick sort, heap sort and insertion sort to provide a very fast and efficient sorting algorithm which runs faster than merge sort in practice.

Noticing this, Kira and Nom have decided to create a new sorting algorithm. They decided to just use quick sort, but if the recursion depth of the partition exceeds $2 \log n$, you have to use merge sort to sort that partition instead. They decided to name this Nomki sort. Interestingly, Nomki sort is quicker than merge sort since it has better constant factors. This must be true since quick sort is generally faster than merge sort and we avoid the $O(n^2)$ worst case of quick sort by using merge sort for larger partitions.

Given an array of size $n$, you have to sort it in time complexity $O(n \log n)$ using only Nomki sort algorithm (Use $\log 2$ function from math.h to find max depth $(2 \log n)$ and pass it as a parameter to quicksort, if in any recursion step depth=0 sort that partition using merge sort instead).

## Input

The first line contains one positive integer $n$ $(1 \leq n \leq 10^5)$, the size of array $a$.
The second line contains $n$ integers $a_1, a_2, \ldots, a_n$ $(-10^9 \leq a_i \leq 10^9)$.

## Output

The only line of output should contain $n$ integers of the sorted array $a$.

---

```
input
20
20 15 27 28 20 26 15 23 22 18 11 3 10 5 21 21 5 3 11 21

output
3 3 5 5 10 11 11 15 15 18 20 20 21 21 21 22 23 26 27 28
```

---

```
input
7
3456 96452 31846 48843 1234678 124657 85678

output
3456 31846 48843 85678 96452 124657 1234678
```

---

# F: Counting Inversions

Divide and conquer is a class of algorithms where you try to divide your problem into non-overlapping subproblems and then merge the solution of the subproblems to get the original answer. As you may have learnt in class Karatsuba and Merge sort are divide and conquer algorithms. Now we want you to solve another popular problem which can be solved using divide and conquer.

Given an array of size $n$, count the number of pairs of $(i, j)$ such that $i < j$ and $a_i > a_j$.

Since Nom and Kira have already learnt this problem in DAA, they help you by giving you some hints. During the merge step of merge sort, if the next smallest element is coming from the right half (even after checking from first half), it means all the remaining elements in the left half are greater than this element.

**Note:** You are supposed to solve this problem in $O(n \log n)$.

## Input

The first line contains one positive integer $n$ ($1 \leq n \leq 10^5$), the size of array $a$.
The second line contains $n$ integers $a_1, a_2, \ldots, a_n$ ($-10^9 \leq a_i \leq 10^9$).

## Output

A single integer - the total number of inversions.

---

```
input
4
8 4 2 1

output
6
explanation
Given array has six inversions:  (8,4),(4,2),(8,2),(8,1),(4,1),(2,1)
```

---

```
input
5
1 20 6 4 5

output
5
explanation
Given array has five inversions:  (20,6),(20,4),(20,5),(6,4),(6,5)
```

---

# G: Nom's Revenge

After finding out that Kira had not been working for months, and just doing lottery, Nom chained him and told him to solve DSA Assignment problems. While Nom was eating his Peanut Butter, he gave him an array and told him to find the sum of $k$ smallest elements in $O(n \log k)$ time complexity. Assume 1-based indexing.

## Input

The first line contains two positive integers $n$, the size of array $a$; and $k$ $(1 \leq k \leq n \leq 10^5)$.
The second line contains $n$ integers $a_1, a_2, \ldots, a_n$ $(-10^9 \leq a_i \leq 10^9)$.

## Output

A single integer - sum of $k$ smallest elements.

---

input
6 2
8 2 1 0 2 3

output
1

---

input
10 4
92 11 292 28 73 102 82 0 12 1

output
24

---

input
7 6
8237 9832 2199 5362 9382 1372 9932

output
36384

---

# H: Merge K Sorted Arrays

You have seen how to merge two sorted arrays in a previous lab sheet. You have also used it for merge sort in this sheet. Also for the DAA sort you have basically merged $k$ sorted arrays in $O(n \log k)$ time. But surprisingly we can achieve this same time complexity without using a divide and conquer algorithm by using heaps.

Given $k$ sorted arrays of sizes $n_1, n_2, n_3, \ldots, n_k$, merge all these arrays into one sorted array.

You have to solve this problem in time complexity $O(n \log k)$ where $n$ is the summation of $n_i$.

## Input

The first line contains $k$ ($1 \leq k \leq 1000$), the total number of arrays.
Consequent $k$ lines contain a single integer $n_i$ ($1 \leq n_i \leq 1000$) followed by $n_i$ sorted positive integers $\leq 10^9$ in the same line.

## Output

A single array with all the integers merged and sorted.

---

```
input
5
3 1 4 5
2 4 7
2 8 9
4 1 2 3 4
3 10 11 21

output
1 1 2 3 4 4 4 5 7 8 9 10 11 21
```

---

```
input
3
5 1 2 3 4 10
6 4 6 8 10 120 2400
1 1

output
1 1 2 3 4 4 6 8 10 10 120 2400
```

---

# I: Median Millionaire Valentine's

Seeing how rich people were getting in BITS, Donut Didi decided to find herself a guy with the median income in her area which is Jubilee Hills, for valentine's day. She made her bestie go around every house and enquire their income and append it to an array he's storing. But since Donut Didi was very curious she asked her bestie the median many times during his enquiry.

Since Didi's bestie is smarter than her, he used a combination of max-heap and min-heap to solve this problem in $O(n \log n)$ time. Assume that the array is initially empty.

Note: Take the $\lfloor \frac{n}{2} \rfloor$-th element as the median.

## Input

The first line of input contains one integer $q$ ($1 \leq q \leq 10^5$),
The following $q$ lines contain queries in on of the following 2 forms.

- 1 $x$ - add element $x$ to the array.
- 2 - print the current median of the array.

## Output

Print the median of the current array for each query of form 2.

---

```
input
8
1 7
1 14
1 3
2
1 20
2
1 10
2

output
7
7
10
```

---

# J: Bon Voyage

Since Bon Voyage is at its peak, the organisers are concerned as to change the voting timings. They release a form and as for preferred voting time slots, with starting and ending time. After collecting the slots, they decide to merge any overlapping slots and turn them into one single slot until there are no overlapping slots left.

Two slots $sl_1$ and $sl_2$ are said to be overlapping if the ending time of $sl_1$ is $\geq$ the starting time of $sl_2$ and the starting time of $sl_1$ is $\leq$ the ending time of $sl_2$. Merging two overlapping slots produces a single new slot with start time = minimum of start time of both slots, end time = maximum of end time of both slots.

## Input

The first line contains a single integer $n$ ($1 \leq n \leq 2 * 10^3$), the number of preferred voting slots.
Consequent $n$ lines contain two integers each $s_i$ and $e_i$ ($1 \leq s_i < e_i \leq 10^9$).

## Output

The first line should contain $k$, the number of new slots.
Consequent $k$ lines should be sorted and should contain two integers $snew_i$ and $enew_i$ each.

---

```
input
4
1 3
8 10
2 6
15 18

output
3
1 6
8 10
15 18
```

---

```
input
2
4 5
1 4

output
1
1 5
```

---