Data Structure and Algorithm, Spring 2024
Mini-Homework F

TA E-mail: dsa_ta@csie.ntu.edu.tw

Due: 13:00:00, Tuesday, April 16, 2024

Rules and Instructions

- Any form of cheating, lying, or plagiarism will not be tolerated. Students can get zero scores and/or fail the class and/or be kicked out of school and/or receive other punishments for those kinds of misconducts.
- Discussions on course materials and homework solutions are encouraged. But you should write the final solutions alone and understand them fully. Books, notes, and Internet resources (including chatGPT) can be consulted, but not copied from.
- Since everyone needs to write the final solutions alone, there is **absolutely no need to lend your homework solutions and/or source codes to your classmates at any time.** In order to maximize the level of fairness in this class, lending and borrowing homework solutions are both regarded as dishonest behaviors and will be punished according to the honesty policy.
- In each mini-homework, we will have one programming problem that is closely connected to what we have taught in the class. The purpose of the problem is for you to *practice* what you have learned, before conquering the more challenging problems in other homework sets.
- Like other programming problems, you should write your code in C programming language, and then submit the code via the *DSA Judge*:

  https://dsa2024.csie.org/.

  Each day, you can submit up to 5 times for each problem. The judge system will compile your code with

  ```
  gcc main.c -static -O2 -std=c11
  ```

- For debugging in the programming part, we strongly suggest you produce your own test-cases with programs and/or use tools like `gdb` or compile with flag `-fsanitize=address` to help you solve issues like illegal memory usage. For `gdb`, you are strongly encouraged to use compile flag `-g` to preserve symbols after compiling.

  Note: For students who use IDE (VScode, DevC++...) you can modify the compile command in the settings/preference section. Below are some examples:

  - `gcc main.c -O2 -std=c11 -fsanitize=address # Works on Unix-like OS`
  - `gcc main.c -O2 -std=c11 -g # use it with gdb`

- The score of the part that is submitted after the deadline will get some penalties according to the following rule:

$$Late\ Score = \max\left(\left(\frac{5 \times 86400 - Delay\ Time(sec.)}{5 \times 86400}\right) \times Original\ Score, 0\right)$$

- If you have questions about this mini-homework, please go to the the Discord channel and discuss the issues with your classmates. This is *strongly preferred* because it will provide everyone with a more interactive learning experience. If you really need an email answer, please follow the rules outlined below to get a fast response:

  - The subject should contain one tag, `"[hwe]"`, specifying the problem where you have questions. For example, `"[hwe] Will there be a repeated number?"`. Adding these tags allows the TAs to track the status of each email and to provide faster responses to you.
  - If you want to provide your code segments to us as part of your question, please upload it to Gist or similar platforms and provide the link. Please remember to protect your uploaded materials with proper permission settings. Screenshots or code segments directly included in the email are discouraged and may not be reviewed.

Programming Part

Max-Heap Construction by Multiple Insertions (20 pts)

Problem Description

Given an array A that contains N distinct elements, we can call MAX-HEAP-INSERT$(A, A[n])$ for $n = 1, 2, \ldots, N$ to convert A into a max heap. Although constructing the heap by those insertions may not be the fastest approach (check the textbook for faster ways!), it is somewhat simpler for you to understand and implement. :-) The MAX-HEAP-INSERT algorithm, as listed in the textbook, is

MAX-HEAP-INSERT(A, key)
1 $A.heapsize = A.heapsize + 1$
2 $A[A.heapsize] = -\infty$
3 HEAP-INCREASE-KEY$(A, A.heapsize, key)$

The MAX-HEAP-INSERT algorithm calls HEAP-INCREASE-KEY to float the new key up. We strongly believe that you should know how to implement PARENT(i) from our lecture. :-)

HEAP-INCREASE-KEY(A, i, key)
1 $A[i] = key$
2 **while** $i > 1$ **and** $A[\text{PARENT}(i)] < A[i]$
3 SWAP$(A[\text{PARENT}(i)], A[i])$
4 $i = \text{PARENT}(i)$

Please output the resulting max heap after the N insertions.

Input

The first line contains the number N. The second line contains N positive integers separated by spaces, representing the initial $A[1], A[2], \ldots, A[N]$.

Output

Output the array-represented max-heap after N insertions within a line, separating each number by a space.

Constraint

- $1 \leq N \leq 10^6$
- $1 \leq A[n] \leq 10^9$
- All $A[n]$ are distinct.

Sample Testcases

Sample Input 1

```
2
1 3
```

Sample Output 1

```
3 1
```

Sample Input 2

```
4
8 9 6 4
```

Sample Output 2

```
9 8 6 4
```

Sample Input 3

```
4
9 8 4 6
```

Sample Output 3

```
9 8 4 6
```

Sample Input 4

```
4
6 8 4 9
```

Sample Output 4

```
9 8 4 6
```

Hint

- By design, you can pass this homework by simulating the algorithms properly. There is no need for other arithmetic calculations or cuts.