Heap Sort

(heap_sort.cpp/c)

Time Limit： 1 sec , Memory Limit： 262144 KB

There are a number of sorting algorithms which have different characteristics as follows.

Algorithm	Time Complexity (Worst)	Time Complexity (Average)	Stability	Memory Efficiency	Method	Features
Insertion Sort ALDS1_1_A	$O(N^2)$	$O(N^2)$	○	○	Insertion	Can be fast for an almost sorted array
Bubble Sort ALDS1_2_A	$O(N^2)$	$O(N^2)$	○	○	Swap	
Selection Sort ALDS1_2_B	$O(N^2)$	$O(N^2)$	×	○	Swap	
Shell Sort ALDS1_2_D	$O(N^2)$	$O(N \log N)$	×	○	Insertion	
Merge Sort ALDS1_5_A	$O(N \log N)$	$O(N \log N)$	○	×	Divide and Conqure	Fast and stable. It needs an external memory other than the input array.
Counting Sort ALDS1_6_A	$O(N + K)$	$O(N + K)$	○	×	Bucket	Fast and stable. There is a limit to the value of array elements.
Quick Sort ALDS1_6_B	$O(N^2)$	$O(N \log N)$	×	△	Divide and Conqure	Fast and almost-in-place if it takes a measure for the corner cases. Naive implementation can be slow for the corner cases.
Heap Sort ALDS1_9_D (This problem)	$O(N \log N)$	$O(N \log N)$	×	○	Heap structure	Fast and in-place. It is unstable and performs random access for non-continuous elements frequently.

The Heap Sort algorithms is one of fast algorithms which is based on the heap data structure. The algorithms can be implemented as follows.

```
1  maxHeapify(A, i)
2      l = left(i)
3      r = right(i)
4      // select the node which has the maximum value
5      if l ≦ heapSize and A[l] > A[i]
6          largest = l
7      else
8          largest = i
9      if r ≦ heapSize and A[r] > A[largest]
10         largest = r
11
12     if largest ≠ i
13         swap A[i] and A[largest]
14         maxHeapify(A, largest)
15
16 heapSort(A):
17     // buildMaxHeap
18     for i = N/2 downto 1:
19         maxHeapify(A, i)
20     // sort
21     heapSize ← N
22     while heapSize ≧ 2:
23         swap(A[1], A[heapSize])
24         heapSize--
25         maxHeapify(A, 1)
```

On the other hand, the heap sort algorithm exchanges non-continuous elements through random accesses frequently.

Your task is to find a permutation of a given sequence A with N elements, such that it is a max-heap and when converting it to a sorted sequence, the total number of swaps in maxHeapify of line 25 in the pseudo code is maximal possible.

Input (heap_sort.in)

In the first line, an integer N is given. In the next line, A_i ($i = 0, 1, \ldots, N - 1$) are given separated by a single space character.

Output (heap_sort.out)

Print N integers of the permutation of A separated by a single space character in a line.

This problem has multiple solutions and the judge will be performed by a special validator.

Constraints

- $1 \leq N \leq 200000$
- $0 \leq$ each element of $A \leq 1000000000$
- The elements of A are all different

Sample Input and Output

Sample Input 1

```
8
1 2 3 5 9 12 15 23
```

Sample Output 1

```
23 9 15 2 5 3 12 1
```