1 Problem: Convert array into heap

Problem Introduction

In this problem you will convert an array of integers into a heap. This is the crucial step of the sorting algorithm called HeapSort. It has guaranteed worst-case running time of $O(n \log n)$ as opposed to QuickSort's average running time of $O(n \log n)$. QuickSort is usually used in practice, because typically it is faster, but HeapSort is used for external sort when you need to sort huge files that don't fit into memory of your computer.

Problem Description

Task. The first step of the HeapSort algorithm is to create a heap from the array you want to sort. By the way, did you know that algorithms based on Heaps are widely used for external sort, when you need to sort huge files that don't fit into memory of a computer?

Your task is to implement this first step and convert a given array of integers into a heap. You will do that by applying a certain number of swaps to the array. Swap is an operation which exchanges elements a_i and a_j of the array a for some i and j. You will need to convert the array into a heap using only $O(n)$ swaps, as was described in the lectures. Note that you will need to use a min-heap instead of a max-heap in this problem.

Input Format. The first line of the input contains single integer n. The next line contains n space-separated integers a_i.

Constraints. $1 \le n \le 100\,000$; $0 \le i, j \le n - 1$; $0 \le a_0, a_1, \ldots, a_{n-1} \le 10^9$. All a_i are distinct.

Output Format. The first line of the output should contain single integer m — the total number of swaps. m **must satisfy conditions** $0 \le m \le 4n$. The next m lines should contain the swap operations used to convert the array a into a heap. Each swap is described by a pair of integers i, j — the 0-based indices of the elements to be swapped. After applying all the swaps in the specified order the array must become a heap, that is, for each i where $0 \le i \le n - 1$ the following conditions must be true:

1. If $2i + 1 \le n - 1$, then $a_i < a_{2i+1}$.
2. If $2i + 2 \le n - 1$, then $a_i < a_{2i+2}$.

Note that all the elements of the input array are distinct. Note that any sequence of swaps that has length at most $4n$ and after which your initial array becomes a correct heap will be graded as correct.

Time Limits. `C`: 1 sec, `C++`: 1 sec, `Java`: 3 sec, `Python`: 3 sec. `C#`: 1.5 sec, `Haskell`: 2 sec, `JavaScript`: 3 sec, `Ruby`: 3 sec, `Scala`: 3 sec.

Memory Limit. 512Mb.

Sample 1.

Input:

```
5
5 4 3 2 1
```

Output:

```
3
1 4
0 1
1 3
```

Explanation:
After swapping elements 4 in position 1 and 1 in position 4 the array becomes 5 1 3 2 4.

After swapping elements 5 in position 0 and 1 in position 1 the array becomes 1 5 3 2 4.
After swapping elements 5 in position 1 and 2 in position 3 the array becomes 1 2 3 5 4, which is already a heap, because $a_0 = 1 < 2 = a_1, a_0 = 1 < 3 = a_2, a_1 = 2 < 5 = a_3, a_1 = 2 < 4 = a_4$.

Sample 2.

Input:

```
5
1 2 3 4 5
```

Output:

```
0
```

Explanation:
The input array is already a heap, because it is sorted in increasing order.

Starter Files

There are starter solutions only for C++, Java and Python3, and if you use other languages, you need to implement solution from scratch. Starter solutions read the array from the input, use a quadratic time algorithm to convert it to a heap and use $\Theta(n^2)$ swaps to do that, then write the output. You need to replace the $\Theta(n^2)$ implementation with an $O(n)$ implementation using no more than $4n$ swaps to convert the array into heap.

What to Do

Change the `BuildHeap` algorithm from the lecture to account for min-heap instead of max-heap and for 0-based indexing.

Need Help?

Ask a question or see the questions asked by other learners at this forum thread.