Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2283 - Diseño y Análisis de Algoritmos - 2do semestre 2022

TAREA 1

Publicación:	Lunes 29 de agosto.
GitHub Classroom:	https://classroom.github.com/a/q0Od5iym
Entrega:	**Miércoles 7 de septiembre 23:59 horas**.

Indicaciones

- La tarea es estrictamente individual.
- La solución debe ser entregada en el archivo `t1.py` del repositorio privado asignado mediante GitHub Classroom para esta tarea. Se revisará el último *commit* subido antes de la entrega al repositorio. Se usará Python 3.10.X para la revisión.
- El *input* para el programa debe ser obtenido desde *standard input*. El *output* debe ser entregado mediante *standard output*.
- La corrección se realizará mediante *tests* automatizados acordes al formato de *input* y *output* especificado. Cada *test* tendrá un *timeout* según lo que se especifica como tiempo esperado.
- Un *test* se considerará **reprobado** en caso de que 1) dado el *input* el *output* sea incorrecto, 2) exista un error de *runtime* durante su ejecución, o 3) el *timeout* se cumpla durante su ejecución. En otro caso, el *test* se considerará **aprobado**.
- No se permite el uso de librerías externas a la librería estándar de Python *a priori*. Consultar en las *issues* del repositorio oficial del curso en caso de requerir una.
- **Para esta tarea sí aplica la política de atrasos descrita en el programa del curso.**

Problema

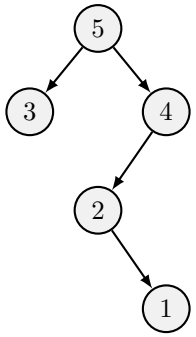
Una **permutación** es una secuencia $[a_1, a_2, \ldots, a_n]$ de números naturales en la que todos los números del conjunto $\{1, \ldots, n\}$ aparecen exactamente una vez. Por ejemplo, [1], $[3, 5, 2, 1, 4]$, $[1, 3, 2]$ son permutaciones, y $[2, 3, 2]$, $[4, 3, 1]$, [0] no lo son.

A Policarpo le han entregado recientemente una permutación $a = [a_1, ..., a_n]$ de longitud n. A Policarpo le gustan más los árboles que las permutaciones, así que quiere transformar la permutación a en un árbol binario. Su idea es transformar un arreglo de diferentes números naturales en un árbol de la siguiente manera:

- El elemento máximo del arreglo se convierte en la raíz del árbol.
- Todos los elementos a la izquierda del máximo forman un subárbol izquierdo (que se construye según las mismas reglas pero aplicadas a la parte izquierda del arreglo), pero si no hay elementos a la izquierda del máximo, entonces la raíz no tiene un hijo izquierdo.

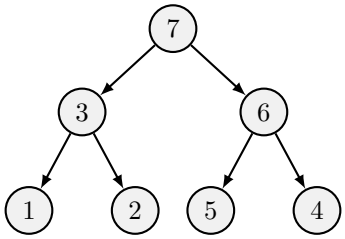
- Todos los elementos a la derecha del máximo forman un subárbol derecho (que se construye según las mismas reglas pero aplicadas a la parte derecha del arreglo), pero si no hay elementos a la derecha del máximo, entonces la raíz no tiene un hijo derecho.

Por ejemplo, si construye un árbol a partir de la permutación $a = [3, 5, 2, 1, 4]$, entonces la raíz será el elemento $a_2 = 5$, y el subárbol izquierdo será el árbol que se construirá para el sub-arreglo $a[1...1] = [3]$, y el derecho para el subarreglo $a[3...5] = [2, 1, 4]$. Como resultado, se construirá el siguiente árbol:



Árbol correspondiente a la permutación $[3, 5, 2, 1, 4]$

Otro ejemplo: que la permutación sea $a = [1, 3, 2, 7, 5, 6, 4]$. En este caso, el árbol tiene el siguiente aspecto:



Árbol correspondiente a la permutación $[1, 3, 2, 7, 5, 6, 4]$

Denotemos por d_v la profundidad del vértice a_v, es decir, el número de aristas del camino desde la raíz hasta el vértice numerado a_v. Nótese que la profundidad de la raíz es cero. Dada la permutación a, para cada vértice, se debe encontrar el valor de d_v.

Input

La primera línea contiene un número entero t ($1 \leq t \leq 10^5$) que corresponde a el número de casos de prueba. Luego siguen t casos de prueba. La primera línea de cada caso de prueba contiene un número entero n ($1 \leq n \leq 10^5$) que corresponde a la longitud de la permutación. A continuación aparecen n números $a_1, a_2, ..., a_n$ que corresponden a la permutación a.

Sean $n_1, \ldots, n_t$ los tamaños de las permutaciones para cada uno de los t casos de prueba. Se define N como la suma de los tamaños de las permutaciones

$$N = \sum_{i=1}^{t} n_i.$$

Se puede asumir que siempre se cumple la restricción $1 \leq N \leq 10^5$.

Output

Para cada caso de prueba, una lista separada por espacios de las profundidades de los n valores en el árbol binario: $d_1, d_2, ..., d_n$.

Tiempo esperado

Se espera que la solución se ejecute en un tiempo **menor o igual a 4 segundos** para cualquier instancia de input según las restricciones dadas.

Complejidad esperada

Se espera que la solución posea una complejidad de $\mathcal{O}(N \log(N))$ siendo $N = \sum_{i=1}^{t} n_i$.

Hint: Use dividir para conquistar. Luego para optimizar cada llamada recursiva investigue una estructura de datos que permita realizar consultas de rango de manera eficiente.

Ejemplo

Los siguientes tests están ya cargados a GitHub Classroom con corrección automática mediante GitHub Actions. **Los tests para la corrección serán distintos a estos.**

Input
5 5 3 5 2 1 4 1 1 4 4 3 1 2 10 1 2 3 4 5 6 7 8 9 10 7 4 2 7 1 5 3 6
Output
1 0 2 3 1 0 0 1 3 2 9 8 7 6 5 4 3 2 1 0 1 2 0 3 2 3 1