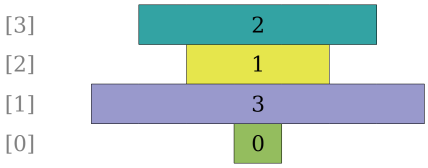
Quase-Hanoi

Um problema clássico estudado em cursos de Ciência da Computação é a torre de Hanoi. A torre é bastante utilizada para explicar os conceitos de pilha e complexidade assintótica.

Na torre de Hanoi existem vários discos de tamanhos diferentes e cada disco possui um furo em seu centro. Os discos são empilhados ao redor de um pino. O problema envolvendo a torre é deslocar todos os discos de um pino para um segundo pino, utilizando um terceiro pino como espaço de armazenamento temporário, de modo que nunca haja mais de um disco fora dos pinos e que um disco jamais fique abaixo de um disco maior que ele.

O problema que você terá que resolver agora lembra a torre de Hanoi. Existe uma coleção de discos, cada um de um tamanho distinto dos demais, empilhados ao redor de um pino. Diferentemente da torre de Hanoi, não existe restrição de que um disco não pode ficar abaixo de um disco maior que ele, por isso essa torre é chamada torre de quase-Hanoi. Um exemplo pode ser visto na figura abaixo:

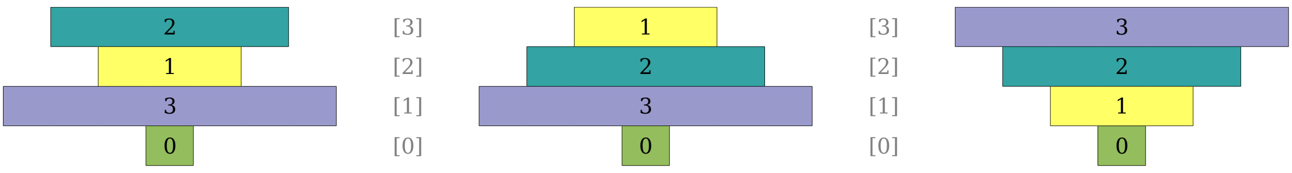


Exemplo de torre de quase-Hanoi

Cada bloco representa um disco. Os números interiores aos blocos representam os tamanhos dos blocos. Note que um bloco de tamanho 0 (zero) não é um bloco inválido, mas o menor bloco possível. O bloco na base da torre está na posição 0 e o bloco no topo da torre está na posição 3.

A única operação que você pode fazer com essa torre é girar uma subtorre do topo. A subtorre S_t é a subtorre cuja base está na posição t da torre original. Uma operação de giro na subtorre S_t, `giro(`t`)`, inverte a ordem dos discos nas posições t, t+1, t+2, ..., N-1 (sendo N o número de discos da torre).

Por exemplo, na torre da figura abaixo à esquerda, a subtorre S_2 é a subtorre formada pelos discos de tamanhos 1 e 2. Uma operação `giro(2)` inverte a ordem dos discos 1 e 2, gerando a torre no centro da figura abaixo. Uma operação de `giro(1)` nessa torre inverte a ordem dos discos 3, 2 e 1, gerando a torre à direita da figura abaixo.



Torre original *Após giro(2)* *Após giro(1)*

Coincidentemente, a torre resultante das operações `giro(2)` e `giro(0)` é uma torre de quase-Hanoi em ordem inversa, isto é, na qual cada disco está apoiado sobre um disco menor. Uma nova operação de `giro(0)` produziria uma torre de quase-Hanoi em ordem.

O seu trabalho é encontrar uma sequência de operações `giro(`t_1`)`, `giro(`t_2`)`, ..., `giro(`t_k`)` que faça uma torre de quase-Hanoi qualquer virar uma torre de quase-Hanoi em ordem. Qualquer sequência válida servirá como resposta.

Descrição da Entrada

A entrada é composta por vários casos de teste. Cada caso de teste é uma sequência de números inteiros N, T_0, T_1, T_2, .., $T_{N\text{-}1}$ (`0` <= T_i <= `1000`) sendo que N é o número de discos na torre e cada T_i é o tamanho é do disco na posição i. Não haverá dois discos de mesmo tamanho em um caso de teste.

A entrada termina quando N = `0`. Nenhum caso conterá mais que cem discos.

Descrição da Saída

Para cada caso de teste, imprima o número do caso de teste conforme exemplificado a seguir, seguido da sequência de índices t_1, t_2, ..., t_k que define uma sequência de operações `giro`(t_1), `giro`(t_2), ..., `giro`(t_k) conforme exemplificado a seguir. Atente para o espaçamento, para a pontuação e para caso especial em que k = `0`.

Lembre-se que a saída esperada apresentada abaixo não é a única saída válida. Não se esqueça que há um limite de tempo e um limite de saída para a execução do seu programa; evite passos redundantes nas sequências, como `giro`(x) imediatamente seguido pelo mesmo movimento `giro`(x) ou `giro`(N-1) em qualquer situação.

Exemplo de Entrada e Saída

<table>
<tr><th>Entrada</th><th>Saída esperada</th></tr>
<tr><td><pre>4
3 2 1 0

4
0 1 2 3

4
3 1 2 0

0</pre></td><td><pre>Caso #1: em ordem.
Caso #2: 0.
Caso #3: 2 1 2.</pre></td></tr>
</table>