

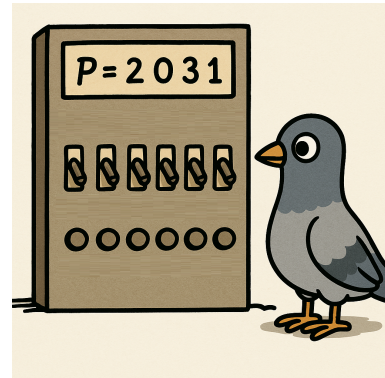


PROBLEMA INDOVINA LA PERMUTAZIONE

James Bambù si è ritrovato in una situazione piuttosto cervelotica!

James ha trovato un'antica macchina, al cui interno sono presenti N interruttori nascosti con stati $s[0], s[1], s[2], \dots, s[N-1]$. In particolare, l' i -esimo interruttore è spento (se $s[i] = 0$), o acceso (se $s[i] = 1$). Inizialmente, tutti gli interruttori sono spenti. La macchina ha anche N pulsanti numerati da 0 a $N-1$, uno schermo, e una permutazione nascosta $P = p[0], p[1], p[2], \dots, p[N-1]$ di $[0, 1, 2, \dots, N-1]$.

Un grande cartello sulla macchina dice: “se riesci a indovinare la permutazione P , vinci un grande premio!” Dato che James vuole il premio, ti chiede aiuto.



Situazione cervelotica

Per determinare P , puoi effettuare alcune operazioni. Ogni operazione consiste nel premere un pulsante della macchina. Quando premi il pulsante i -esimo, accade quanto segue:

1. Sia k il numero di operazioni effettuate strettamente prima di quella attuale. Ad esempio, nella prima operazione, $k = 0$.
2. L'interruttore con indice $p[k \% N]$ viene invertito (formalmente, $s[p[k \% N]] := 1 - s[p[k \% N]]$);
3. Lo schermo mostra lo stato attuale dell'interruttore con indice i (il valore di $s[i]$).

PROBLEMA Ti viene dato l'intero N . Devi determinare la permutazione nascosta P , premendo i pulsanti della macchina un numero sufficientemente piccolo di volte.

Poiché la macchina è vecchia, si romperà se premi i pulsanti troppe volte; quindi, se effettui più di $50 \cdot N$ operazioni, non ottieni il premio.

Il tuo punteggio dipende dal numero di operazioni effettuate. Per vincoli più dettagliati, fai riferimento alla sezione Scoring.

IMPLEMENTAZIONE Il file inviato deve contenere `#include "permutation.h"`.

Devi implementare la seguente funzione:

```
std::vector<int> solve(int N);
```

Questa funzione riceve N come parametro e deve restituire un vettore q di dimensione N tale che per ogni $0 \leq i \leq N-1$ valga $q[i] = p[i]$. La funzione è chiamata esattamente una volta per ogni test.

All'interno della tua soluzione puoi chiamare la seguente funzione:

```
int press_button(int i);
```

Questa funzione restituisce il valore di $s[i]$ (ovvero il valore sullo schermo) dopo aver effettuato un'operazione premendo il pulsante i -esimo.



- ASSUNZIONI**
- ◆ $N \leq 10^4$
 - ◆ P è una permutazione di $[0, 1, 2, \dots, N - 1]$.

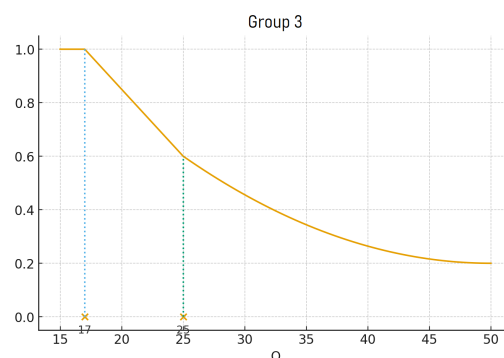
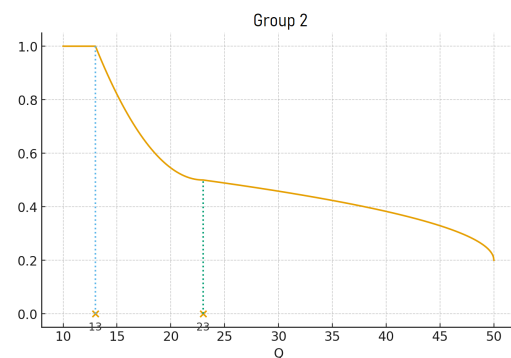
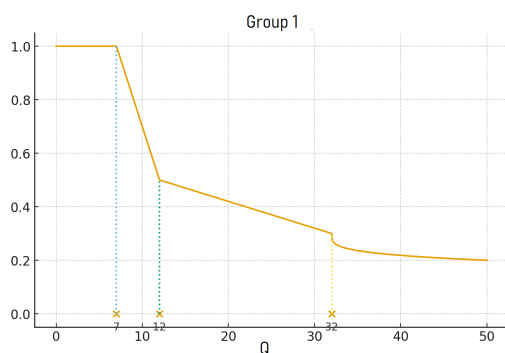
#	Punti	Assunzioni
1	20	$N = 32$
2	40	$N = 1\,000$
3	40	$N = 10\,000$

- ASSEGNAZIONE DEL PUNTEGGIO**
- Ogni test case viene valutato come segue. Sia K il numero di volte in cui la chiamata alla funzione `press_button()` effettuate dalla tua soluzione. Sia $Q = \frac{K}{N}$. Se la permutazione è indovinata in modo errato oppure se $Q > 50$, il punteggio è 0. Altrimenti, il punteggio è dato da S moltiplicato per il punteggio massimo del gruppo, dove S è definito come segue:

$$\text{Gruppo 1. } S = \begin{cases} 1, & Q \leq 7 \\ 1 - \frac{1}{10}(Q - 7), & 7 < Q \leq 12 \\ \frac{1}{2} - \frac{1}{100}(Q - 12), & 12 < Q \leq 32 \\ \frac{3}{10} - \frac{1}{10} \sqrt[4]{\frac{Q-32}{18}}, & 32 < Q \leq 50 \end{cases}$$

$$\text{Gruppo 2. } S = \begin{cases} 1, & Q \leq 13 \\ \frac{1}{2} + \frac{1}{2} \left(\frac{23-Q}{10} \right)^2, & 13 < Q \leq 23 \\ \frac{1}{5} + \frac{3}{10} \sqrt{\frac{50-Q}{27}}, & 23 < Q \leq 50 \end{cases}$$

$$\text{Gruppo 3. } S = \begin{cases} 1, & Q \leq 17 \\ \frac{3}{5} + \frac{25-Q}{20}, & 17 < Q \leq 25 \\ \frac{1}{5} + \frac{2}{5} \left(\frac{50-Q}{25} \right)^2, & 25 < Q \leq 50 \end{cases}$$





■ **ESEMPIO DI INTERAZIONE** Considera un test case con $N = 4$ e $P = [1, 3, 0, 2]$, e le seguenti chiamate a `press_button()`.

Azione	S	Valore restituito
<code>press_button(0)</code>	0100	
		0
<code>press_button(1)</code>	0101	
		1
<code>press_button(2)</code>	1101	
		0
<code>press_button(3)</code>	1111	
		1
<code>press_button(1)</code>	1011	
		0
<code>press_button(3)</code>	1010	
		0
<code>press_button(2)</code>	0010	
		1
<code>return {1,3,0,2}</code>		

■ **GRADER DI ESEMPIO** Ti viene fornito grader di esempio (`sample-grader.cpp`). Il grader legge l'input nel seguente formato:

- riga 1: intero N
- riga 2: N interi $p[0], p[1], p[2], \dots, p[N - 1]$.

Poi chiama la funzione `solve(N)` e stampa l'esito della tua soluzione, insieme al numero di operazioni effettuate o una spiegazione dell'errore.