# D. Garden Decorations

| Nome del problema | Garden Decorations |
|---|---|
| Limite di tempo | 7 secondi |
| Limite di memoria | 1 gigaottetto |

Ogni giorno, quando va a scuola e torna a casa, Detje cammina lungo una strada con $N$ case, numerate da $0$ a $N-1$. Attualmente, la casa $i$ è abitata dalla persona $i$. Per cambiare aria, i residenti hanno deciso di scambiarsi le case. La persona che si trasferirà nella casa $i$ è la persona $a_i$ (che attualmente vive nella casa $a_i$).

Ogni casa ha una statua di un pinguino nel giardino. Le statue hanno due possibili stati: le loro ali sono *aperte* (come se il pinguino stesse volando) o *chiuse* (come se fosse in piedi a terra). I residenti hanno preferenze molto forti su come dovrebbero apparire le loro statue di pinguini. Attualmente, il pinguino di fronte alla casa $i$ è nello stato preferito del residente $i$. I residenti si rifiutano di trasferirsi in una casa a meno che la statua non sia impostata nella posizione che preferiscono.

Detje vuole aiutarli a sistemare le statue dei pinguini in modo che possano muoversi.

Per questo, fa quanto segue: ogni volta che cammina lungo la strada (sia mentre va a scuola che quando torna a casa), osserva i pinguini che incontra uno per uno e, eventualmente, sistema alcune delle statue (aprendo o chiudendo le ali). Dato che le sue giornate a scuola e a casa sono molto impegnative, **non ricorda gli stati dei pinguini che ha visto nelle sue precedenti passeggiate**. Fortunatamente, ha scritto la lista $a_0, a_1, ..., a_{N-1}$, quindi sa quale residente si sta trasferendo e dove.

Aiuta Detje a progettare una strategia che le dica come cambiare lo stato dei pinguini per adattare le statue ai gusti dei residenti. Può camminare lungo la strada al massimo $60$ volte, ma per ottenere un punteggio più alto, deve camminare lungo la strada meno volte possibili.

## Implementazione

Questo è un problema multi-run, il che significa che il tuo programma verrà eseguito più volte.

In ogni esecuzione, dovrai prima leggere una riga con due numeri interi, $w$ e $N$, l'indice della passeggiata e il numero di case. Nella prima esecuzione del tuo programma $w = 0$, nella seconda $w = 1$ e così via (ulteriori dettagli sono spiegati di seguito).

Nella seconda riga di input ci sono $N$ numeri interi $a_0, a_1, ..., a_{N-1}$, il che significa che la persona che si trasferirà nella casa $i$ vive attualmente nella casa $a_i$. Gli $a_i$ formano una *permutazione*: ovvero, ogni numero da $0$ a $N-1$ appare esattamente una volta nell'elenco degli $a_i$. Nota che un residente può scegliere di non trasferirsi; ovvero, è consentito $a_i = i$.

I residenti cambiano casa solo una volta. Ciò significa che per un testcase fisso, il valore di $N$ e l'elenco di $a_i$ saranno gli stessi per tutte le esecuzioni del programma.

## Prima esecuzione.

Per la prima esecuzione del tuo programma, $w = 0$. In questa esecuzione, dovrai semplicemente stampare un singolo intero $W$ ($0 \leq W \leq 60$), il numero di volte in cui vuoi che Detje passi davanti alle case. Il tuo programma dovrà quindi terminare. Dopodiché, il tuo programma verrà eseguito di nuovo $W$ altre volte.

## Esecuzioni successive.

Nella successiva esecuzione del tuo programma, $w = 1$; in quella successiva $w = 2$; e così via fino all'esecuzione finale in cui $w = W$.

Dopo aver letto $w$, $N$ e $a_0, a_1, ..., a_{N-1}$, Detje inizia a camminare lungo la strada.

- Se $w$ è dispari, Detje cammina da casa a scuola e passerà davanti alle case nell'ordine $0, 1, ..., N-1$.

  Il tuo programma deve ora leggere una riga con $b_0$, $0$ (chiuso) o $1$ (aperto), lo stato attuale della statua di fronte alla casa $0$. Dopo aver letto $b_0$, devi stampare una riga con $0$ o $1$, il nuovo valore a cui vuoi impostare $b_0$.

  Quindi il tuo programma deve leggere una riga con $b_1$, lo stato della statua di fronte alla casa $1$; e stampare il nuovo valore di $b_1$. Questo continua per ciascuna delle $N$ case. Dopo che Detje supera l'ultima casa (ovvero, hai letto e scritto $b_{N-1}$), **il tuo programma deve terminare**.

  *Nota che il tuo programma può leggere solo il valore successivo $b_{i+1}$ dopo aver scritto il nuovo valore di $b_i$.*

- Se $w$ è pari, Detje cammina da scuola a casa e passerà invece davanti alle case nell'ordine inverso $N-1, N-2, ..., 0$. Ovvero, inizi a leggere e scrivere $b_{N-1}$, poi $b_{N-2}$ e così via fino a $b_0$.

Quando $w = 1$, i valori di input $b_0, b_1, ..., b_{N-1}$ sono gli stati originali delle statue dei pinguini (che sono anche gli stati preferiti dei residenti). Quando $w > 1$, i valori di input $b_0, b_1, ..., b_{N-1}$ per il tuo programma saranno quelli impostati dall'esecuzione precedente del tuo programma.

Alla fine, dopo l'esecuzione finale del tuo programma, il valore di $b_i$ deve essere uguale al valore originale di $b_{a_i}$ per tutti gli $i$, altrimenti otterrai il verdetto Wrong Answer.

### Dettagli.

Se la *somma* dei tempi di esecuzione delle $W+1$ esecuzioni separate del tuo programma supera il limite di tempo, la tua sottoposizione riceverà il verdetto Time Limit Exceeded.

Assicurati di eseguire flush dello standard output dopo aver stampato ogni riga, altrimenti il tuo programma potrebbe ricevere il verdetto Time Limit Exceeded. In Python, questo avviene automaticamente se usi `input()` per leggere le righe. In C++, `cout << endl;` fa flush oltre a stampare una nuova riga; se usi printf, usa `fflush(stdout)`.

## Limiti e punteggio

- $2 \le N \le 500$.
- Puoi usare al massimo $W \le 60$ round.

La tua soluzione verrà testata su un set di subtask, ognuno dei quali vale un certo numero di punti. Ogni subtask contiene un set di testcase. Per ottenere i punti per un subtask, devi risolvere tutti i testcase nel subtask.

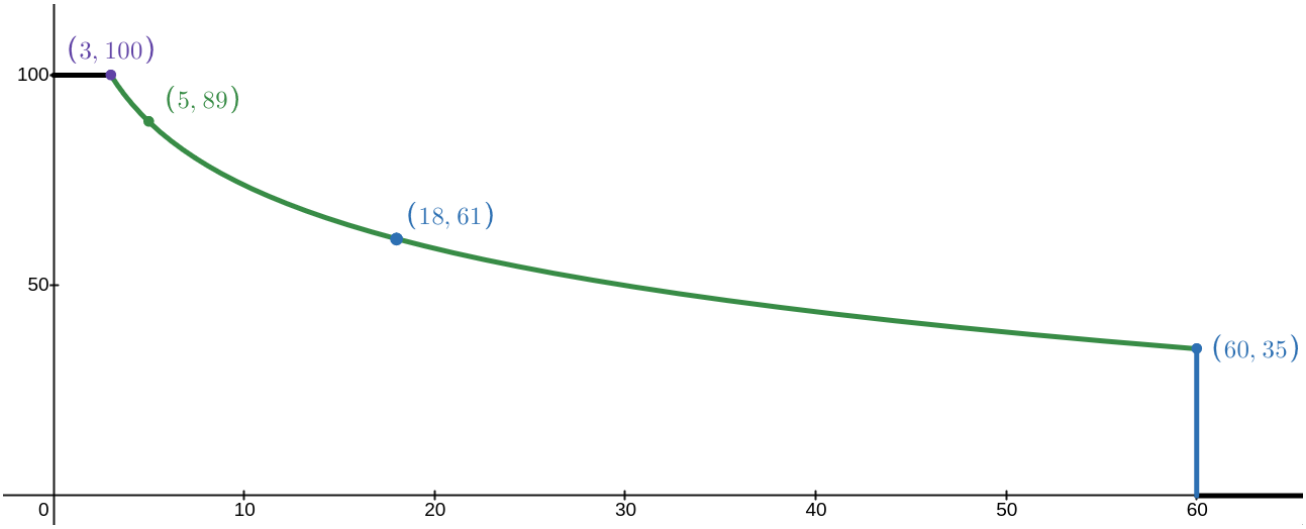
| Gruppo | Punteggio massimo | Limiti |
|---|---|---|
| 1 | 10 | $N = 2$ |
| 2 | 24 | $N \le 15$ |
| 3 | 9 | $a_i = N - 1 - i$ |
| 4 | 13 | $a_i = (i + 1) \bmod N$ |
| 5 | 13 | $a_i = (i - 1) \bmod N$ |
| 6 | 31 | Nessun limite aggiuntivo |

Per ogni subtask che il tuo programma risolve correttamente, riceverai un punteggio basato sulla seguente formula:

$$\text{score} = S_g \cdot \left(1 - \tfrac{1}{2} \log_{10}(\max(W_g, 3)/3)\right),$$

dove $S_g$ è il punteggio massimo per il subtask e $W_g$ è il valore massimo di $W$ utilizzato per qualsiasi tescase nel subtask. Il tuo punteggio per ogni subtask verrà arrotondato all'intero più vicino.

Il grafico seguente mostra il numero di punti, in funzione di $W$, che il tuo programma otterrà se risolve tutti i gruppi di test con lo stesso valore di $W$. In particolare, per ottenere un punteggio di 100 punti su questo problema, devi risolvere ogni testcase con $W \leq 3$.




## Tool di test

Per facilitare il test della tua soluzione, forniamo un semplice tool che puoi scaricare. Vedi "allegati" in fondo alla pagina del problema di Kattis. Il tool è facoltativo da usare. Nota che il programma di valutazione ufficiale su Kattis è diverso dal tool di test.

Per usare il tool, crea un file di input, per esempio "sample1.in", che inizia con un numero $N$ seguito da una riga con $N$ numeri che specificano la permutazione e un'altra riga con $N$ bit ($0$ o $1$) che specificano gli stati iniziali dei pinguini. Ad esempio:

```
6
1 2 0 4 3 5
1 1 0 0 1 0
```

Per i programmi Python, per esempio `solution.py` (normalmente eseguito come `pypy3 solution.py`):

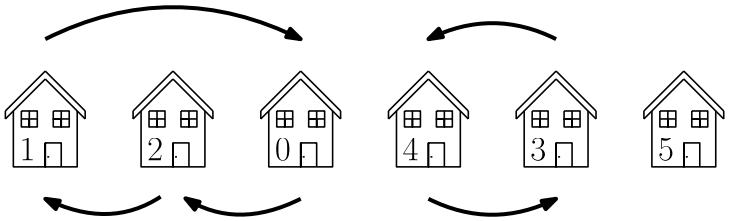
```
python3 testing_tool.py pypy3 solution.py < sample1.in
```

Per i programmi C++, prima compilalo (ad esempio con `g++ -g -O2 -std=gnu++20 -static solution.cpp -o solution.out`) e poi esegui:

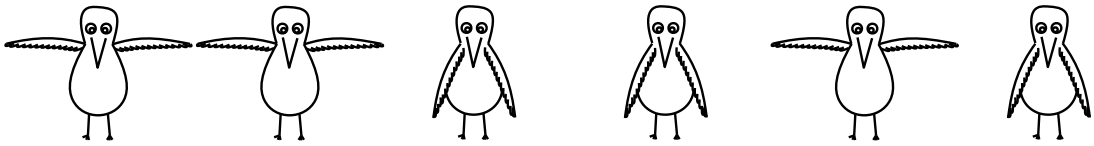
```
python3 testing_tool.py ./solution.out < sample1.in
```

## Esempio

Nell'esempio, ci viene data la seguente permutazione di persone nelle case:

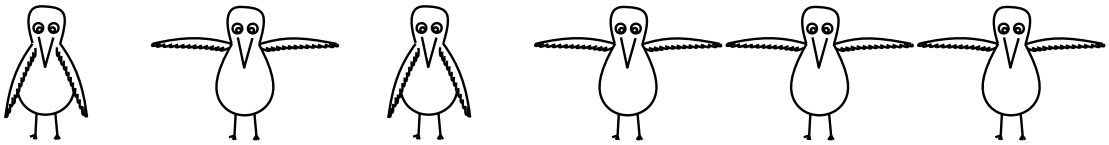



La prima volta che il programma di esempio viene eseguito (con $w = 0$), restituisce $W = 2$, il che significa che Detje camminerà lungo la strada due volte (e il programma verrà eseguito altre due volte). Prima della prima passeggiata, i pinguini nei giardini hanno questo aspetto:



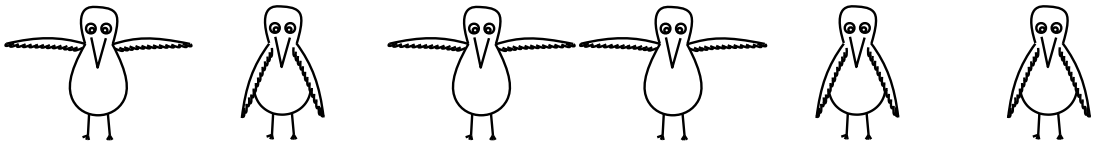
Dopodichè il programma viene eseguito con $w = 1$, che indica la prima passeggiata di Detje. Passa attraverso i pinguini uno alla volta, iniziando da sinistra, e possibilmente cambia il loro stato. Il programma di esempio deve restituire lo stato dell'$i$-esimo pinguino prima che venga visto l'$(i + 1)$-esimo pinguino.

Dopo che Detje è arrivata a scuola, gli stati dei pinguini sono:



Nell'ultima esecuzione del programma (con $w = 2$), Detje torna a casa da scuola. Ricorda che in questo caso, esaminerà i pinguini da destra a sinistra e li elaborerà in questo ordine! Questo significa che deve determinare lo stato dell'$i$-esimo pinguino prima di vedere l'$(i - 1)$-esimo pinguino.

Dopo aver terminato la sua passeggiata, i pinguini ora sono così:



In effetti, questa è la configurazione corretta. Ad esempio, la statua dell'uccello $3$ (ovvero la quarta da sinistra) è aperta (ora $b_3 = 1$), il che è corretto poiché la persona $4$ si sposterà lì ($a_3 = 4$) e originariamente aveva una statua di pinguino aperta (originariamente $b_4 = 1$).

| grader output | your output |
|---|---|
| 0 6 | |
| 1 2 0 4 3 5 | |
| | 2 |

| grader output | your output |
|---|---|
| 1 6 | |
| 1 2 0 4 3 5 | |
| 1 | |
| | 0 |
| 1 | |
| | 1 |
| 0 | |
| | 0 |
| 0 | |
| | 1 |
| 1 | |
| | 1 |
| 0 | |
| | 1 |

| grader output | your output |
|---|---|
| 2 6 | |
| 1 2 0 4 3 5 | |
| 1 | |
| | 0 |
| 1 | |
| | 0 |
| 1 | |
| | 1 |
| 0 | |
| | 1 |
| 1 | |
| | 0 |
| 0 | |
| | 1 |