D. Decoraciones de Jardín

Nombre del Problema	Garden Decorations
Límite de Tiempo	7 segundos
Límite de Memoria	1 gigabyte

Cada día que Detje va a la escuela y de regreso a su casa, camina a lo largo de una calle con N casas, numeradas de 0 a $N-1$. La casa i está habitada actualmente por la persona i. Para tener un cambio de paisaje, los residentes decidieron cambiar casas unos con otros. La persona que se va a mudar a la casa i es la persona a_i (quien actualmente vive en la casa a_i).

Cada casa tiene una estatua de un pájaro en el jardín. Las estatuas tienen dos posiciones posibles, ya sea con las alas *abiertas* (como si el pájaro estuviera volando) o *cerradas* (como si estuviera parado en el piso). Los residentes tienen preferencias muy fuertes en cómo se deben ver sus estatuas y se rehúsan a mudarse a su nueva casa antes de que la estatua en su nuevo jardín se vea tal y como se veía en su jardín de antes. Detje quiere ayudarles a acomodar las estatuas para que se puedan mudar.

Para esto hace lo siguiente: cada vez que camina a lo largo de la calle (ya sea de ida hacia la escuela o de regreso a casa), observa a los pájaros que pasa y posiblemente ajusta algunas de las posiciones (abriéndoles o cerrándoles las alas). Como sus días están muy ocupados, no recuerda la posición que tenían los pájaros que vió en sus caminatas previas. Por suerte, escribió la lista $a_0, a_1, ..., a_{N-1}$, para saber qué residente se va a mudar a dónde.

Ayuda a Dejte a diseñar una estrategia que le diga cuáles pájaros debe maniuplar para ajustar las estatuas al gusto de todos los residentes. Puede caminar a lo largo de la calle a lo más 60 veces, pero para lograr un puntaje mayor, debe caminar menos veces a lo largo de la calle.

Implementación

Tu programa será ejecutado en múltiples ocasiones.

En cada ejecución, debes leer una línea con dos enteros w y N, el índice de la caminata y el número de casas. En la primera ejecución de tu programa $w = 0$, en la segunda $w = 1$ y así sucesivamente (abajo se explican más detalles).

En la segunda línea de la entrada hay N enteros $a_0, a_1, ..., a_{N-1}$, indicando que la persona que se va a mudar a la casa i actualmente vive en la casa a_i. Las a_is forman una *permutación*: esto es, cada número de 0 a $N-1$ aparece exactamente una vez en la lista de a_is. Observa que un residente puede escoger no mudarse; esto significa que $a_i = i$ está permitido.

Los residentes solamente se cambian una vez de casa. Esto significa que para un caso en específico, el valor de N y de la lista de a_is serán las mismas para todas las ejecuciones de tu programa.

Primera Ejecución

Para la primera ejecución de tu programa, $w = 0$. En esta ejecución, simplemente debes imprimir un sólo entero W ($0 \leq W \leq 60$), el número de veces que quieres que Detje camine por las casas. Después, tu programa debe terminar. Después de esto, tu programa será ejecutado otras W veces.

Ejecuciones Subsecuentes

En la siguiente ejecución de tu programa, $w = 1$; en la que sigue, $w = 2$; y así sucesivamente hasta la última ejecución en la que $w = W$.

Después de leer w, N y $a_0, a_1, ..., a_{N-1}$, Detje empieza a caminar a lo largo de la calle.

- Si w es impar, Detje camina desde su casa a la escuela y pasa por las casas en el orden $0, 1, ..., N-1$.

  Después tu programa debe leer una línea con b_0, que puede ser 0 (cerradas) o 1 (abiertas), la posición actual de la estatua en frente de la casa 0. Después de leer b_0, debes imprimir una línea con un 0 o un 1, el nuevo valor que quieres asignar a b_0.

  Después tu programa debe leer una línea con b_1, la posición actual de la estatua en frente de la casa 1 e imprimier el nuevo valor de b_1. Esto continúa para cada una de las N casas. Después de que pasas por la última casa (es decir, lees y escribes b_{N-1}), **tu programa debe terminar**.

  Ten en cuenta que tu programa solo puede leer el siguiente valor b_{i+1} después de que escribiste el valor de b_i.

- Si w es par, Detje camina desde la escuela hacia su casa y va a pasar por las casas en el orden inverso $N-1, N-2, ...0$. El proceso es el mismo que cuando w es impar, a excepción de que empiezas leyendo y escribiendo b_{N-1}, luego b_{N-2}, y así sucesivamente hasta b_0.

Cuando $w = 1$, los valores de entrada $b_0, b_1, ..., b_{N-1}$ son las posiciones originales de las estatuas de pájaros. Cuando $w > 1$, los valores de entrada $b_0, b_1, ..., b_{N-1}$ para tu programa, tendrán los valores asignados en la ejecución anterior de tu programa.

Al final, después de la última ejecución de tu programa, el valor de b_i debe ser igual al valor original de b_{a_i} para toda i, de otro modo obtendrás el veredicto Wrong Answer.

Detalles

Si la *suma* de los tiempos de ejecución de las $W+1$ ejecuciones individuales de tu programa exceden el tiempo límite, tu envío obtendrá Time Limit Exceeded.

Asegúrate de hacer flush del stream de salida después de cada línea que imprimas, de lo contrario tu programa podría obtener el veredicto Time Limit Exceeded. En Python, esto sucede automáticamente si usas `input()` para leer líneas. En C++, `cout << endl;` también hace flush además de imprimir una línea nueva; si usas printf deberás usar `fflush(stdout)`.

Límites y Evaluación

- $2 \leq N \leq 500$.
- Puedes usar a lo más $W \leq 60$ vueltas.

Tu solución se evaluará con un conjunto de grupos de casos de prueba, cada grupo otorga un valor determinado de puntos. Cada grupo contiene un conjunto de casos de prueba. Para obtener los puntos de un grupo, tienes que resolver todos los casos de prueba de ese grupo.

Grupo	Puntaje máximo	Límites
1	10	$N = 2$
2	24	$N \leq 15$
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	Sin restricciones adicionales

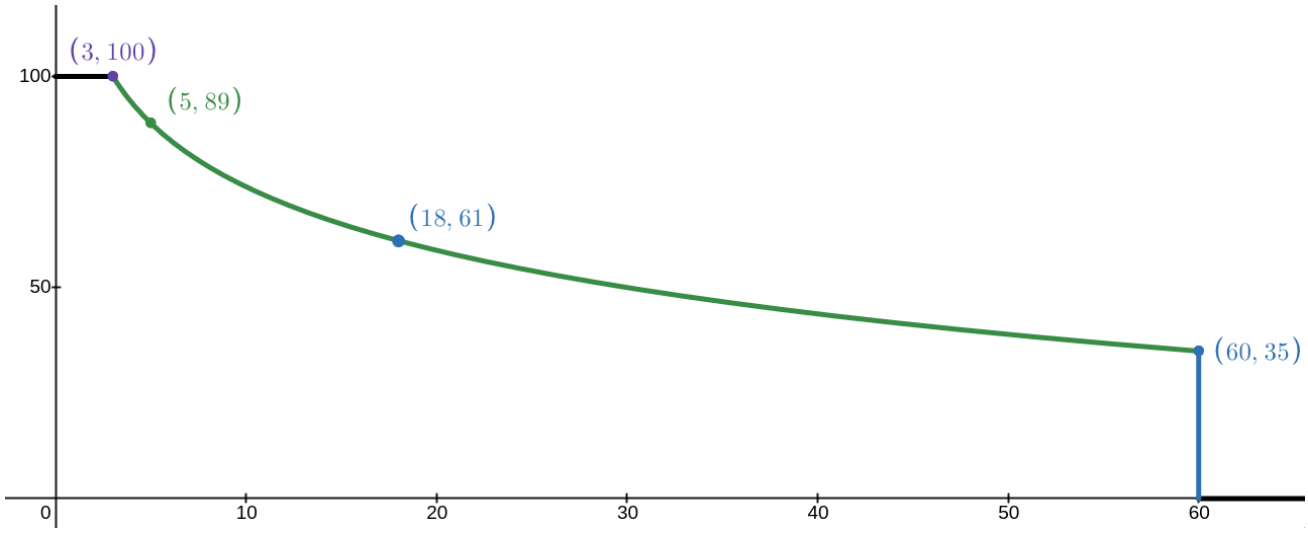
Para cada grupo de prueba que tu programa resuelva correctamente, recibirás un puntaje basado en la siguiente fórmula:

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

donde S_g es el máximo puntaje para el grupo, y W_g es el máximo valor de W usado para cualquier caso de prueba en ese grupo. Tu puntaje para cada grupo será redondeado al próximo entero.

La gráfica de abajo muestra el número de puntos, como una función de W, que tu programa obtendrá si resuelve todos los grupos con el mismo valor de W. Particularmente, para lograr un

puntaje de 100 puntos en este problema, necesitas resolver todos los casos de prueba con $W \leq 3$.



Herramienta de Pruebas

Para facilitarte poder probar tu solución, te damos una herramienta simple que puedes descargar. Ve a la sección "attachments" al final de la página del problema en Kattis. El uso de la herramienta es opcional y puedes cambiarla. Ten en cuenta que el evaluador oficial en Kattis es diferente a esta herramienta de pruebas.

Para usar la herramienta, crea un archivo de entrada, como “sample1.in”, el cual debe empezar con un número N seguido de una línea con N números especificando la permutación y otra línea con N bits (0 o 1) especificando la posición inicial de los pájaros. Por ejemplo:

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

Para programas en python, si tenemos `solution.py` (se corre normalmente `pypy3 solution.py`) corre:

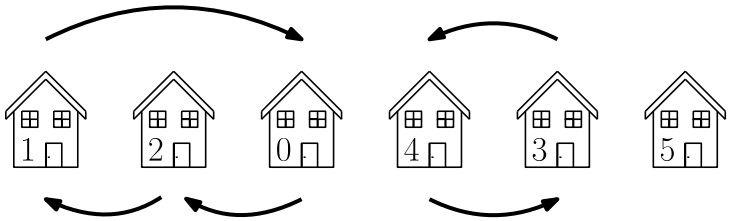
```
python3 testing_tool.py pypy3 solution.py < sample1.in
```

Para programas en C++, primero compílalo (por ejemplo `g++ -g -O2 -std=gnu++20 -static solution.cpp -o solution.out`) y luego corre:

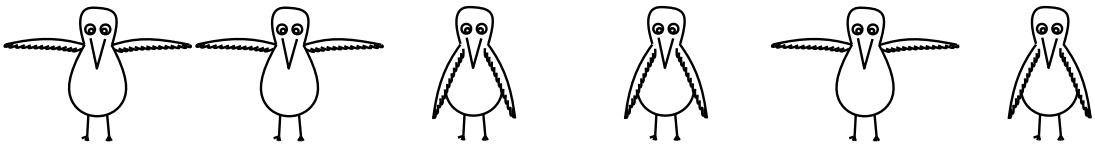
```
python3 testing_tool.py ./solution.out < sample1.in
```

Ejemplo

En el ejemplo, tenemos la siguiente permutación de gente en las casas:

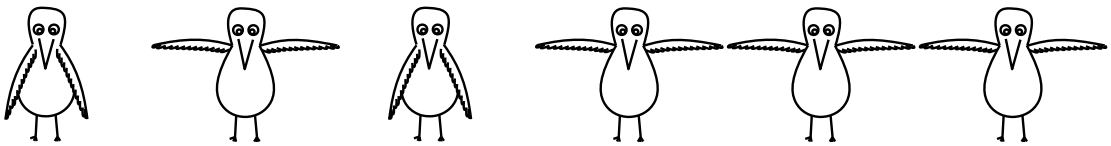


La primera vez que el programa de ejemplo corre (con $w = 0$), imprime $W = 2$, lo que significa que Detje va a caminar a lo largo de la calle dos veces (y el programa se va a ejecutar otras dos veces). Antes de la primer caminata, los pájaros en los jardines se ven de la siguiente manera:



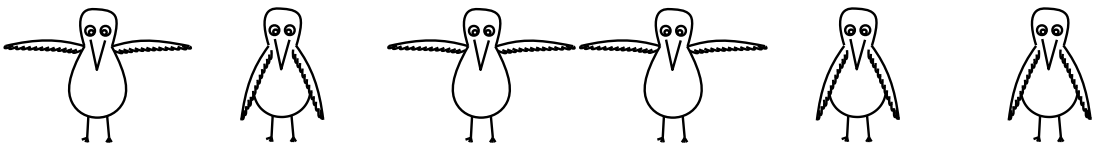
Después el programa se ejecuta con $w = 1$: indicando la primer caminata de Detje. Ella pasa por cada uno de los pájaros, de uno en uno, empezando por la izquierda y posiblemente cambiando su posición. El programa de ejemplo tiene que imprimir la posición del i-ésimo pájaro antes de que veamos el $(i + 1)$-ésimo pájaro.

Después de que Detje llega a la escuela, las posiciones de los pájaros se ven así:



En la última ejecución del programa (con $w = 2$), Detje camina de regreso a su casa. Recuerda que en este caso, ella pasará por cada uno de los pájaros de derecha a izquierda y los procesará en el orden inverso. Esto significa que necesita determinar la posición del i-ésimo pájaro antes de ver al $(i - 1)$-ésimo pájaro.

Después de que termina su caminata, los pájaros ahora se ven así:



Esta es, efectivamente, la configuración correcta. Por ejemplo, la estatua de pájaro 3 (la cuarta desde la izquierda) está abierta (ahora $b_3 = 1$), lo cual es correcto ya que la persona 4 se mudará ahí ($a_3 = 4$) y esta persona tenía originalmente una estatua de pájaro con las alas abiertas (originalmente $b_4 = 1$).

salida del evaluador	tu salida
0 6	
1 2 0 4 3 5	
	2

salida del evaluador	tu salida
1 6	
1 2 0 4 3 5	
1	
	0
1	
	1
0	
	0
0	
	1
1	
	1
0	
	1

salida del evaluador	tu salida
2 6	
1 2 0 4 3 5	
1	
	0
1	
	0
1	
	1
0	
	1
1	
	0
0	
	1