D. Zahradní dekorace

Název úlohy	Garden Decorations
Časový Limit	7 sekund
Paměťový Limit	1 gigabajt

Každý den při chůzi do školy a zpět domů, Detje chodí takovou ulicí, na které je N domů, očíslovaných 0 až $N-1$. Člověka, který bydlí v domě i nazvěme člověkem i. Pro zpestření života si obyvatelé této ulice rozhodli mezi sebou vyměnit své domy. Člověk, který bude bydlet v domě i, je člověk a_i (který před tím bydlel v domě a_i).

Každý dům má před sebou identickou sochu ptáka. Tyto sochy mají dva stavy, a to s rozevřenými křídy (jako kdyby ten pták létal), nebo se složenými křídly (jako kdyby stál na zemi). Každý obyvatel ulice má velmi silné preference o tom, jak by měla vypadat jeho ptačí socha, a odmítá vstoupit do domu, který nemá sochu ve stejném stavu, ve kterém byla v původním domě. Detje jim chce pomoct upravit ptačí sochy tak, aby se mohli všichni spokojeně přestěhovat.

Postupuje následovně: kdykoli jde po ulici (buď do školy, nebo zpátky domů), tak si postupně prohlíží sochy, kolem kterých prochází, a může je upravovat (rozevírat a zavírat křídla soch). Jelikož má hodně práce se školou, nepamatuje, jaký stav měly ptačí sochy při předchozích průchodech. Naštěstí má zapsaný seznam $a_0, a_1, ..., a_{N-1}$, takže ví, který soused se chce přestěhovat kam.

Dejte Detje svoji pomocnou ruku a pomozte jí vymyslet strategii, která jí řekne, které sochy ptáků upravit, aby všechny sochy vyhovovaly svým budoucím obdivovatelům, kteří s ní budou bydlet. Detje projde ulicí maximálně 60krát, ale moc by se jí líbilo, kdyby to zvládla za co nejméně průchodů.

Implementace

V této úloze bude řešení, které odevzdáte, spuštěno vícekrát.

Při každém běhu nejprve načtěte řádek s dvěmi celými čísly w a N, představující číslo průchodu a počet domů. Při prvním běhu programu $w = 0$, v druhém $w = 1$ a tak dále (detaily jsou vysvětleny níže).

Na druhém řádku vstupu je N celých čísel $a_0, a_1, ..., a_{N-1}$, znamenajících, že člověk, který se stěhuje do domu i, doteď bydlel v domě a_i. Čísla a_i tedy tvoří *permutaci*: každé číslo 0 až $N-1$ se

objeví právě jednou v seznamu a_i. Každý obyvatel se může rozhodnout nestěhovat, tedy $a_i = i$ je dovoleno.

Obyvatelé se stěhují pouze jednou. To znamená, že pro daný testovací príklad jsou hodnoty N a a_i v každém běhu vašeho řešení stejné.

První běh

Při prvním běhu vašeho řešení bude $w = 0$. V tomto běhu jednoduše vypište jediné celé číslo W $(0 \leq W \leq 60)$ – kolikrát chcete, aby Detje procházela ulicí. Pak by se vaše řešení mělo ukončit. Vaše řešení se spustí ještě W-krát.

Další běhy

V dalšim běhu bude $w = 1$, pak $w = 2$, a tak dále až po poslední běh, kdy bude $w = W$.

Potom, co načtete w, N a všechna $a_0, a_1, ..., a_{N-1}$, začne Detje chodit ulicí.

- Když je w liché, tak Detje jde směrem ze svého domu ze školy, a to v pořadí $0, 1, ..., N - 1$.

  Vaše řešení by nyní mělo přečíst řádek s jedním číslem b_0, což bude buď 0 (složená) nebo 1 (rozevřená), stav křídel sochy před domem 0. Potom by váše řešení mělo vypsat jeden řádek s novým stavem sochy, u které se nachází Detje, tedy buď 0 (složená) nebo 1 (rozevřená).

  Dále program přečte b_1, stav sochy před domem 1, a vypíše nový stav této sochy. Tohle pokračuje pro každý z N domů. Potom, co Detje přejde poslední dům (tedy načtete stav sochy před tímto domem, a vypíšete její nový stav), by se mělo vaše řešení ukončit.

  Číslo b_{i+1} jde načíst až potom, co vypíšete novou hodnotu pro b_i.

- Pokud je w sudé, Detje jde ze školy domů, a kolem domů prochází v opačném pořadí $N - 1, N - 2, ..., 0$.

  Proces interakce je stejný, jako když je w liché až na to, že nejprve načtete a zapíšete b_{N-1}, pak b_{N-2} a tak dále až po b_0.

Když $w = 1$, tak jsou na vstupu hodnoty $b_0, b_1, ..., b_{N-1}$ tvořící originální stav soch. Když $w > 1$, tak vstup tvoří hodnoty $b_0, b_1, ..., b_{N-1}$, které vaše řešení vypsalo při minulém běhu.

Na konec, po posledním běhu vašeho řešení, musí být hodnota b_i rovna originální hodnotě b_{a_i} pro všechna i, jinak je řešení ohodnoceno jako špatné (Wrong Answer).

Detaily

Pokud součet časů všech $W + 1$ běhů vašeho řešení přesahuje časový limit, tak bude vaše řešení ohodnoceno jako příliš pomalé (Time Limit Exceeded).

Ujistěte se, že po každém vypsání řádku spláchnete standardní výstup, nebo váše řešení možná bude vyhodnoceno jako příliš pomalé (Time Limit Exceeded). V Pythonu se toto děje automaticky, dokud používáte `input()` pro čtení řádků. V C++ `cout << endl;` ukončí řádek a spláchne. Pokud používáte printf, použijte `fflush(stdout)`.

Omezení a bodování

- $2 \leq N \leq 500$.
- Můžete si zvolit maximálně $W \leq 60$.

Vaše řešení bude hodnoceno na několika testovacích sadách. Každá testovací sada obsahuje několik testovacích případů. Pro zisk bodů z testovací sady je potřeba správné vyřešit všechny testovací případy z této sady.

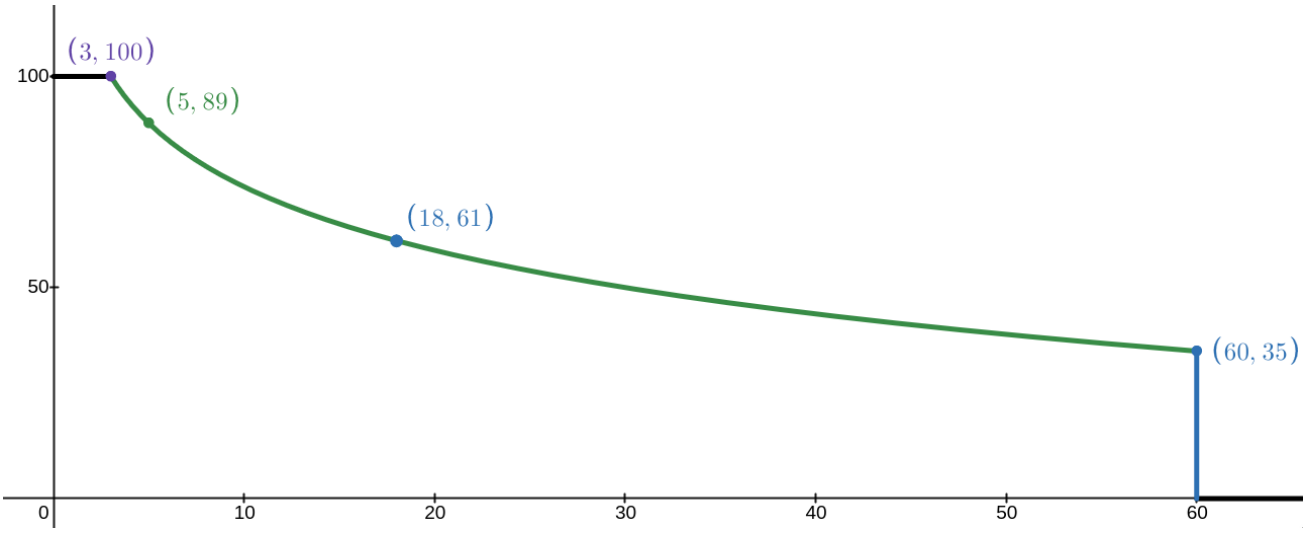
Sada	Maximální body	Omezení
1	10	$N = 2$
2	24	$N \leq 15$
3	9	$a_i = N - 1 - i$
4	13	$a_i = (i + 1) \bmod N$
5	13	$a_i = (i - 1) \bmod N$
6	31	Žádná další omezení

Pro každou testovací sadu, kterou vaše řešení správně vyřeší, dostanete počet bodů podle následujícího vzorce:

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

kde S_g je maximální počet bodů pro danou testovací sadu, a W_g je maximální hodnota W, kterou v této sadě vaše řešení použilo. Tento počet bodů bude zaokrouhlen na nejbližší celé číslo.

Následující graf ukazuje počet bodů jako funkci W, které je maximálním počtem Detjiných přechodů přes všechny testovací příklady. Pro 100 bodů je potřeba vyřešit všechny testovací příklady s $W \leq 3$.



Testovací nástroj

Pro testování vašech řešení máte přístup k jednoduchému testovacímu programu, který si můžete stáhnout. Je v sekci "attachments" dole na stránce úlohy v Kattis. Tento nástroj nemusíte použít. Oficiální testování používá jiný nástroj.

Pro použití nástroje vytvořte nový vstupní soubor, třeba "sample1.in", který začíná řádkem s číslem N a pokračuje řádkem s N čísly, které specifikují permutaci obyvatel, a další řádek s N bity (0 nebo 1), specifikujícími stavy soch. Například:

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

Pro pythonní řešení, například `solution.py` (normálně spouštěno `pypy3 solution.py`):

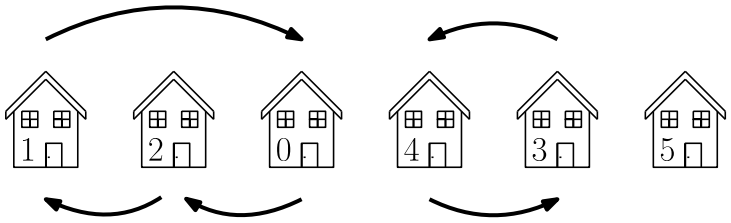
```
python3 testing_tool.py pypy3 solution.py < sample1.in
```

Pro testování řešení v C++ jej nejdřív zkompilujte (třeba pomocí `g++ -g -O2 -std=gnu++20 -static solution.cpp -o solution.out`) a pak spusťte pomocí

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

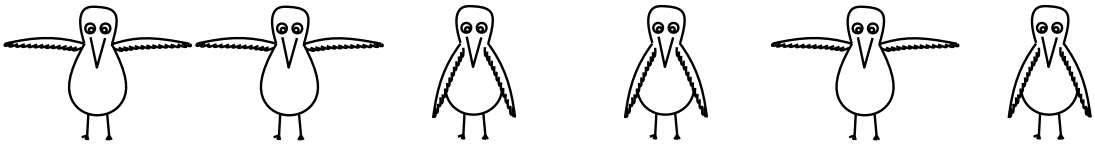
Příklad

V tomto příkladu máme následující permutaci obyvatel domů:



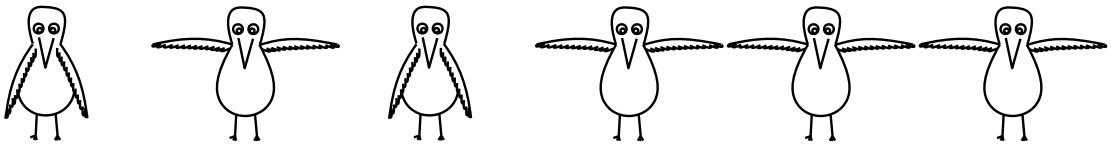
Když běží naše řešení (s $w = 0$), vypíše $W = 2$, takže Detje projde ulici dvakrát, a řešení tedy poběží ještě dvakrát.

Před prvním průchodem vypadájí ptačí sochy následovně:



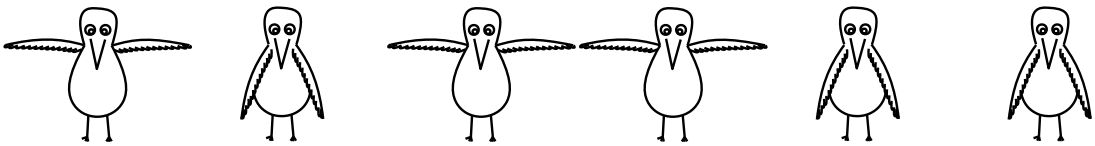
Dále je řešení spuštěno s $w = 1$, takže Detje prochází ulici poprvé. Prochází je po jednom, zleva doprava, a občas mění stav soch. Řešení musí vypsat stav i-tého ptáka, než zjistí stav $(i + 1)$-tého ptáka.

Potom, co Detje dorazí do školy, vypadají sochy následovně:



Při posledním běhu programu ($w = 2$) jde Detje ze školy domů, tedy zprava doleva. To znamená, že řešení musí určit stav i-tého ptáka, než zjistí stav $(i - 1)$-tého ptáka.

Po tomto průchodu vypadají sochy následovně:



Toto je správná konfigurace. Například ptačí socha 3 (tedy čtvrtá zprava) má nyní rozevřená křídla, která člověk 4, který se tam přesouvá, nutně vyžaduje, jelikož na začátku měl ve svém domě sochu s rozevřenými křídly.

výstup hodnotiče	váš výstup
0 6	
1 2 0 4 3 5	
	2

výstup hodnotiče	váš výstup
1 6	
1 2 0 4 3 5	
1	
	0
1	
	1
0	
	0
0	
	1
1	
	1
0	
	1

výstup hodnotiče	váš výstup
2 6	
1 2 0 4 3 5	
1	
	0
1	
	0
1	
	1
0	
	1
1	
	0
0	
	1