D. Zgaduj Zgadula

Nazwa zadania	Zgaduj Zgadula
Limit czasu	4 sekundy
Limit pamięci	1 GB

Na lundzkiej starówce jest ulica, przy której stoi rząd N domów ponumerowanych od 0 do $N-1$. Wiktoria mieszka w jednym z tych domów, a jej koleżanki Jadzia i Ala chcą dowiedzieć się, w którym. Wiktoria, zamiast po prostu powiedzieć swoim koleżankom, gdzie mieszka, zdecydowała, że zagra z nimi w grę. Przed rozpoczęciem gry Jadzia i Ala znają tylko liczbę domów na ulicy. W tym momencie, Jadzia i Ala mogą wybrać liczbę całkowitą K i ustalić strategię. Jakakolwiek komunikacja później jest zakazana.

Gra składa się z dwóch faz. W pierwszej fazie Wiktoria wybiera taką kolejność domów do odwiedzenia, że jej dom będzie ostatnim odwiedzonym. Potem prowadzi Jadzię do domów w tej kolejności, nie ujawniając jej Jadzi wcześniej. Na drzwiach każdego domu, który nie jest domem Wiktorii, Jadzia może napisać kredą jedną liczbę całkowitą o wartości pomiędzy 1 a K włącznie. Na drzwiach ostatniego domu, który odwiedzą (czyli domu Wiktorii), to Wiktoria pisze liczbę pomiędzy 1 a K.

W drugiej fazie Ala idzie po ulicy po kolei mijając domy od 0 do $N-1$ i czyta liczby napisane na drzwiach przez Jadzię i Wiktorię. Chce teraz zgadnąć, w którym domu mieszka Wiktoria. Ma dwie szanse, aby zgadnąć i jeśli jej się uda, to ona i Jadzia wygrywają. W przeciwnym przypadku grę wygrywa Wiktoria.

Czy możesz doradzić Jadzi i Ali, jaką strategię powinny przyjąć, aby zawsze mogły wygrać grę? Twoja strategia będzie oceniana na podstawie wartości K (im mniejsza, tym lepsza).

Implementacja

Twój program będzie uruchamiany wielokrotnie. Przy pierwszym uruchomieniu zaimplementuje strategię Jadzi. Przy drugim - strategię Ali.

Pierwsza linia wejścia zawiera dwie liczby całkowite P i N, gdzie P jest równe 1 lub 2 (pierwsza lub druga faza), a N jest liczbą domów na ulicy. **Poza testem przykładowym (nieuwzględnianym przy ocenianiu) N jest zawsze równe $100\,000$**

Następnie wejście zależy od fazy:

Faza 1

Twój program powinien rozpocząć wykonanie od wypisania liczby K w pojedynczej linii $(1 \leq K \leq 1\,000\,000)$.

Następnie powinien $N-1$ razy wczytać linię zawierającą indeks i $(0 \leq i < N)$ i wypisać linię z liczbą A_i $(1 \leq A_i \leq K)$, gdzie A_i jest liczbą zapisaną przez Jadzię na drzwiach i-tego domu. Każdy indeks i, oprócz indeksu domu Wiktorii, wystąpi dokładnie raz w kolejności, którą wybierze sprawdzaczka.

Faza 2

Twój program powinien wczytać linię z N liczbami całkowitymi $A_0, A_1, ..., A_{N-1}$, gdzie A_i jest liczbą zapisaną na drzwiach i-tego domu.

Następnie powinien wypisać linię z dwiema liczbami całkowitymi s_1 i s_2 $(0 \leq s_i < N)$ - zgadywanymi indeksami. s_1 i s_2 mogą być równe.

Pamiętaj o wyczyszczeniu bufora standardowego wyjścia po każdej linii, którą wypiszesz - w przeciwnym przypadku Twój program może zostać oceniony jako Time Limit Exceeded. W Pythonie `print()` automatycznie czyści bufor. W C++ `cout << endl;` też czyści bufor oprócz wypisywania znaku nowej linii; jeśli używasz printf, użyj `fflush(stdout)`.

Sprawdzaczka do tego problemu może być **adaptywna**, co oznacza, że może zmieniać swoje zachowanie w zależności od wyjścia Twojego programu, aby uniemożliwiać rozwiązaniom heurystycznym przechodzenie testów. Może uruchomić fazę 1, przeanalizować Twoje wyjście, a potem znowu uruchomić fazę 1, używając informacji pozyskanych przy pierwszym uruchomieniu.

Twój program musi być deterministyczny - to znaczy musi zachowywać się tak samo, jeśli zostanie dwa razy uruchomiony na tych samych danych. Jeśli chcesz używać losowości w swoim programie, używaj ustalonego punktu startowego generatora (ang. *random seed*). Możesz to zrobić poprzez przekazywanie stałej zapisanej w kodzie do `srand` (w C++) lub `random.seed` (w Pythonie). Jeśli sprawdzaczka wykryje, że Twój program nie jest deterministyczny, to wyda werdykt Wrong Answer.

Jeśli *suma* czasów wykonania (co najwyżej 3) uruchomień Twojego programu przekroczy limit czasu, Twoje rozwiązanie zostanie ocenione jako Time Limit Exceeded.

Ocenianie

Twoje rozwiązanie będzie uruchamiane na pewnej liczbie testów. Jeśli Twoje rozwiązanie nie zadziała poprawnie na *dowolnym* z tych testów (np. poprzez podanie niepoprawnej odpowiedzi

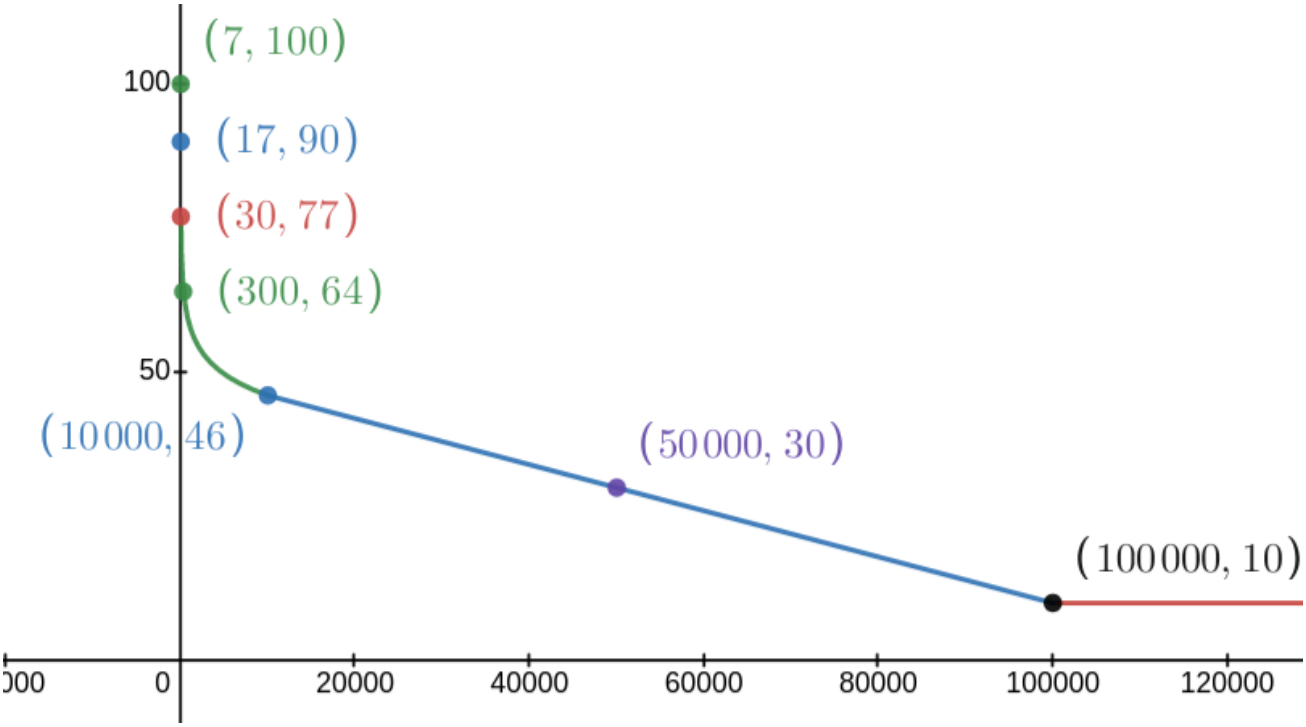
(Wrong Answer), zakończy się błędem wykonania (Run-Time), przekroczy limit czasu (Time Limit Exceeded), itp.), otrzymasz 0 punktów i odpowiedni werdykt.

Jeśli Twój program zakończy się sukcesem i znajdzie tajny indeks we *wszystkich* testach, otrzymasz werdykt Accepted i wynik punktowy obliczany w następujący sposób:

Niech K_{max} będzie maksymalną wartością spośród K używanych we wszystkich przypadkach testowych. W zależności od K_{max}:

	Punktacja
$K_{max} > 99\,998$	10 punktów
$10\,000 < K_{max} \leq 99\,998$	$10 + \lfloor 40(1 - K_{max}/10^5) \rfloor$ punktów
$30 < K_{max} \leq 10\,000$	$46 + \lfloor 31(4 - \log_{10}(K_{max}))/(4 - \log_{10}(30)) \rfloor$ punktów
$7 < K_{max} \leq 30$	$107 - K_{max}$ punktów
$K_{max} \leq 7$	100 punktów

Funkcja punktacji jest przedstawiona na obrazku poniżej.



Test przykładowy jest ignorowany przy ocenianiu, Twoje rozwiązanie nie musi dla niego działać.

Oprogramowanie testowe

Aby ułatwić testowanie Twojego rozwiązania, dostarczamy proste narzędzie, które możesz pobrać. Sprawdź "attachements" na dole strony Kattis. Używanie narzędzia jest opcjonalne i możesz je

modyfikować. Pamiętaj, że oficjalna sprawdzaczka na Kattis jest inna niż oprogramowanie testowe.

Przykładowe użycie (dla $N = 4$, $s = 2$, gdzie s jest liczbą napisaną na ostatnim odwiedzanym domu):

Dla programów w Pythonie np. `solution.py`, (normalnie uruchamianego komendą `pypy3 solution.py`):

```
python3 testing_tool.py pypy3 solution.py <<<"4 2"
```

Programy w C++ należy najpierw skompilować (np. przy użyciu `g++ -g -O2 -std=gnu++17 -static solution.cpp -o solution.out`) a następnie uruchomić:

```
python3 testing_tool.py ./solution.out <<<"4 2"
```

Oprogramowanie testowe następnie odwiedzi domy w losowej kolejności. W celu użycia konkretnej kolejności, zmodyfikuj to oprogramowanie w miejscu oznaczonym “MODIFY HERE”.

Przykładowa interakcja

Test przykładowy jest ignorowany przy ocenianiu i Twoje rozwiązanie nie musi dla niego działać.

Przypuśćmy, że mamy $N = 4$ i Wiktoria mieszka w domu 1. Niech A będzie listą liczb napisanych na domach. Początkowo $A = [0, 0, 0, 0]$, gdzie 0 oznacza, że na drzwiach danego domu nie napisano jeszcze żadnej liczby.

W pierwszym uruchomieniu Twojego programu:

Dane jest $N = 4$. Rozwiązanie odpowiada z $K = 3$.

Żądane jest A_2. Rozwiązanie wypisuje 3. A wynosi teraz $[0, 0, 3, 0]$.

Żądane jest A_0. Rozwiązanie wypisuje 1. A wynosi teraz $[1, 0, 3, 0]$.

Żądane jest A_3. Rozwiązanie wypisuje 2. A wynosi teraz $[1, 0, 3, 2]$.

Na końcu sprawdzaczka ustawia $A_1 = 2$, więc A wynosi teraz $[1, 2, 3, 2]$. To jest koniec pierwszej fazy.

Przy drugim uruchomieniu Twojego kodu programowi jest przekazywana lista `1 2 3 2`.

Odpowiada z `1 3`.

Ponieważ jedna z podanych liczb jest poprawnym indeksem domu Wiktorii (1), Jadzia i Ala wygrywają grę.

Wyjście sprawdzaczki	Twoje wyjście
1 4	
	3
2	
	3
0	
	1
3	
	2

Wyjście sprawdzaczki	Twoje wyjście
2 4	
1 2 3 2	
	1 3