D. Guessing Game

Numele problemei	Guessing Game
Limită de timp	4 secunde
Limite de memorie	1 gigabyte

În centrul vechi al orașului Lund, există o stradă cu N case în șir, numerotate de la 0 la $N-1$. Emma locuiește în una dintre aceste case, iar prietenii ei Anna și Bertil doresc să afle în care anume. În loc să le spună direct prietenilor unde locuiește, Emma decide să joace un joc cu ei.

Înainte de începerea jocului, Anna și Bertil știu doar numărul de case de pe stradă. La acest moment, Anna și Bertil pot alege un număr întreg pozitiv K și pot stabili o strategie. Orice comunicare ulterioară este interzisă.

Jocul în sine constă în două faze. În prima fază, Emma alege o ordine în care să viziteze casele, astfel încât casa ei să fie ultima. Apoi, ea o conduce pe Anna la case una câte una în ordinea aleasă, fără a-i dezvălui în prealabil ordinea. Pentru fiecare casă care nu este casa Emmei, Anna are voie să scrie cu cretă pe ușa casei un singur număr întreg între 1 și K. Pentru ultima casă pe care o vizitează, casa Emmei, Emma va fi cea care scrie un număr întreg între 1 și K pe ușa casei.

În cea de-a doua fază a jocului, Bertil merge de-a lungul străzii de la casa 0 la casa $N-1$ și citește toate numerele scrise pe uși de Anna și Emma. Acum, el dorește să ghicească în care casă locuiește Emma. El are două încercări de a ghici corect, iar el și Anna câștigă jocul dacă reușesc să ghicească casa Emmei. În caz contrar, Emma câștigă jocul.

Poți găsi o strategie prin care Anna și Bertil să câștige jocul în mod garantat? Strategia ta va fi evaluată în funcție de valoarea lui K (cu cât este mai mică, cu atât este mai bună).

Implementare

Sursa voastră va fi rulată de mai multe ori pentru același test. Prima dată când este rulată, va implementa strategia Annei. După aceea, va implementa strategia lui Bertil.

Prima linie va conține două numere întregi, P și N, unde P poate fi 1 sau 2 (prima sau a doua fază), iar N este numărul de case. **Cu excepția exemplelor (care nu este folosită pentru evaluare), N va fi mereu $100,000$.**

Următoarea parte a input-ului depinde de faza în care ne aflăm:

Faza 1

Programul vostru ar trebui să înceapă prin a afișa numărul K, pe o singură linie ($1 \leq K \leq 1\,000\,000$). Apoi, de $N-1$ ori, ar trebui să citească o linie care conține indexul i ($0 \leq i < N$) al unei case și să afișeze un întreg A_i ($1 \leq A_i \leq K$), unde A_i este numărul scris de Anna pe ușa casei i. Fiecare index i, mai puțin cel care corespunde casei Emmei, va apărea exact odată în ordinea aleasă de evaluator.

Faza 2

Programul vostru ar trebui să citească o singură linie cu N numere întregi, $A_0, A_1, \ldots, A_{N-1}$, unde A_i este numărul scris pe cea de-a i-a casă.

Apoi, ar trebui să afișeze, pe o singură linie, două numere întregi s_1 și s_2 ($0 \leq s_i < N$), indicii încercărilor de a ghici. s_1 și s_2 pot fi egale.

Detalii de implementare

Reține că atunci când programul este rulat în Faza 2, acesta este repornit ceea ce înseamnă că nu poți să salvezi informația în variabile între rulări.

După fiecare linie afișată, asigură-te că dai flush la output, altfel programul tău poate lua Time Limit Exceeded. În Python, funcția `print()` face automat flush. În C++, folosirea `cout << endl;` face și flush în plus față de afișarea unui newline; dacă folosești printf, utilizează `fflush(stdout)`.

Evaluatorul pentru această problemă poate fi **adaptiv**, ceea ce înseamnă că poate-și schimba comportamentul în funcție ce afișează programul tău pentru a combate soluțiile euristice. Este posibil chiar ca acesta să ruleze odată programul pentru Faza 1, să analizeze output-ul dat de programul tău, și apoi să ruleze Faza 1 din nou utilizând informații extrase din rularea anterioară.

Programul tău *trebuie* să fie determinist, adică să se comporte la fel dacă este rulat de două ori pe același input. Dacă vrei să folosești valori generate aleator în programul tău, asigură-te că folosești un seed constant. Acest lucru poate fi realizat prin transmiterea unei constante codificate în mod static către funcția `srand` (în C++) sau `random.seed` (în Python), sau, în cazul folosirii generatorului de numere aleatoare din C++11, prin specificarea seed-ului la declararea generatorului de numere aleatoare. În particular, nu poți folosi `srand(time(NULL))` în C++. Dacă evaluatorul detectează că programul tău nu este determinist, vei primi verdictul Wrong Answer.

Dacă *suma* timpilor de execuție a mai multe (până la 3) rulări separate ale programului tău depășește limita de timp, submisia ta va primi verdictul Time Limit Exceeded.

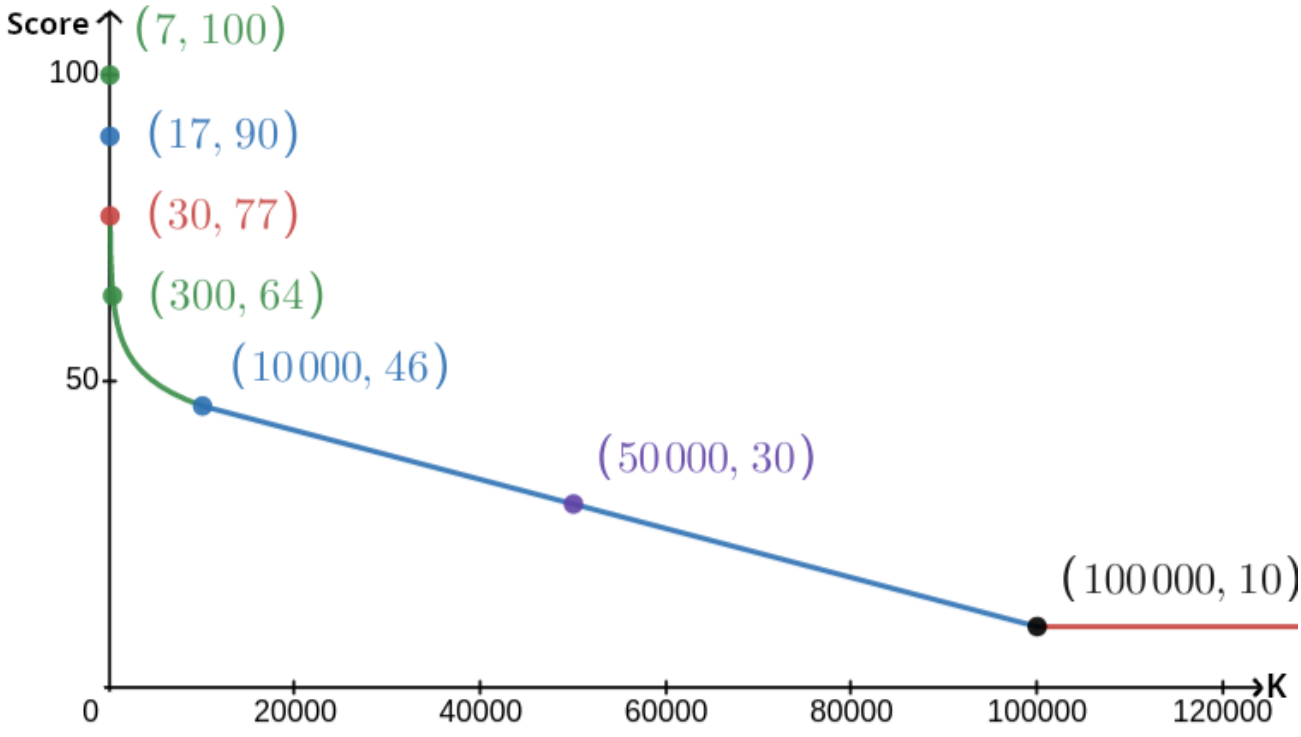
Punctaj

Soluția ta va fi testată pe mai multe teste. Dacă soluția ta greșește pe *cel puțin unul* dintre aceste teste (de exemplu, prin furnizarea de răspunsuri incorecte (Wrong Answer), crăpare (Run-Time Error), depășirea limitei de timp (Time Limit Exceeded), etc.), vei primi 0 puncte și verdictul corespunzător.

Dacă programul tău găsește cu succes indexul casei lui Emma în *toate* pentru toate testele, vei primi verdictul Accepted și un scor calculat în felul următor. Fie K_{max} valoarea maximă a lui K utilizată în oricare caz de testare. În funcție de K_{max}:

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

Funcția de calcul a scorului este reprezentată în figura de mai jos.



Exemplul este ignorat pentru evaluare, iar soluția ta nu trebuie să funcționeze pentru acesta.

Tool de testare

Pentru a facilita testarea soluției , punem la dispoziție un tool simplu pe care il puteți descărca. Vezi secțiunea "Atașamente" în partea de jos a paginii problemei pe Kattis. Utilizarea tool-ului este opțională și acesta poate fi modificat. Programul oficial de evaluare de pe Kattis este diferit de tool-ul de testare.

Exemplu de utilizare (cu $N = 4$, $s = 2$, unde s este numărul scris pe ușa ultimei case vizitate):

Pentru un program în Python, să zicem `solution.py` (care s-ar rula normal folosind comanda `pypy3 solution.py`):

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

Pentru un program în C++, acesta trebuie întâi compilat (ex. folosind `g++ -g -O2 -std=gnu++17 -static solution.cpp -o solution.out`) și apoi tool-ul se rulează folosind comanda:

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

Tool-ul de testare va vizita casele într-o ordine aleatoare. Pentru a folosi o ordine anume, modificați programul acolo unde scrie “MODIFY HERE”.

Exemplu de interacțiune

Exemplul este ignorat pentru evaluare, iar soluția ta nu trebuie să funcționeze pentru acesta.

Să presupunem că avem $N = 4$ și că Emma locuiește în casa 1. Fie A lista de numere scrise pe uși. Inițial, $A = [0, 0, 0, 0]$, unde 0 înseamnă că nu este scris niciun număr pe casa corespunzătoare.

În prima rulare a codului tău:

Se dă $N = 4$. Soluția ta răspunde cu $K = 3$.

Se solicită A_2. Soluția ta răspunde cu 3. A devine $[0, 0, 3, 0]$.

Se solicită A_0. Soluția ta răspunde cu 1. A devine $[1, 0, 3, 0]$.

Se solicită A_3. Soluția ta răspunde cu 2. A devine $[1, 0, 3, 2]$.

În final, evaluatorul setează $A_1 = 2$, astfel $A = [1, 2, 3, 2]$ la sfârșit. Acesta marchează sfârșitul primei faze.

În Faza 2 a codului tău, soluția primește lista `1 2 3 2`.

Răspunde cu `1 3`.

Deoarece una dintre încercări este indexul corect al casei (1), Anna și Bertil câștigă jocul.

grader output	your output
1 4	
	3
2	
	3
0	
	1
3	
	2

grader output	your output
2 4	
1 2 3 2	
	1 3