D. Guessing Game

Feladatnév	Guessing Game
Időkorlát	4 másodperc
Memóriakorlát	1 gigabyte

Lund óvárosában van egy utca, ahol N ház sorakozik egymás mellett, 0-tól $N-1$-ig sorszámoztuk őket. Emma az egyik ilyen házban lakik, és a barátai, Anna és Bertil ki akarják találni, hogy melyik az. Ahelyett, hogy egyszerűen megmondaná a barátainak, hogy melyik az, Emma úgy dönt, hogy játszik velük egyet. A játék kezdete előtt Anna és Bertil csak azt tudja, hogy összesen hány ház van az utcában. Ekkor Anna és Bertil választhat egy pozitív egész K számot és megegyezhetnek egy stratégiában. Ezt követően minden kommunikáció tilos közöttük.

Maga a játék két fázisból áll. Az első fázisban Emma kiválasztja a házak sorrendjét úgy, hogy az ő háza legyen az utolsó, amit meglátogat. Ezután ebben a sorrendben elvezeti Annát a házakhoz, anélkül, hogy a sorrendet előre megmondaná neki. Minden olyan házhoz, amely nem Emma háza, Anna krétával egy 1 és K közötti egész számot írhat a ház bejárati ajtajára. Az utolsó meglátogatott háznál - amely Emma háza - Emma maga ír egy 1 és K közötti egész számot az ajtóra.

A játék második fázisában Bertil végigsétál az utcán a 0 sorszámú háztól az $N-1$ sorszámúig, és elolvassa a játékban szereplő összes számot, amiket Anna és Emma az ajtókra írt. Ki akarja kitalálni, hogy Emma melyik házban lakik. Két esélye van helyes tippre, és ha sikerül, ő és Anna megnyerik a játékot. Ellenkező esetben Emma nyeri a játékot.

Ki tudsz találni egy olyan stratégiát, amellyel Anna és Bertil garantáltan megnyerik a játékot? A stratégiádat a K értéke alapján pontozzuk (minél kisebb, annál jobb).

Megvalósítás

A programod többször is végrehajtásra kerül. Az első futtatáskor Anna stratégiáját hajtja végre. Ezután Bertil stratégiáját fogja végrehajtani.

A bemenet első sora két egész számot tartalmaz: P-t és N-t, ahol P értéke 1 vagy 2 (első vagy második fázis), és N a házak száma.

Az alábbi (pontozáshoz nem használt) minta kivételével **az N mindig $100\,000$.**

A bemenet ezután a fázistól függ:

1. fázis

Ezután a programod egy sorba írja ki a K egész számot, $(1 \leq K \leq 1\,000\,000)$. Ezután $N-1$ alkalommal olvassa be az egyetlen i indexet tartalmazó sort $(0 \leq i < N)$. A programod ezután írja ki egyetlen sorba az A_i $(1 \leq A_i \leq K)$ egész számot, amelyet Anna az i. ház ajtajára ír. Az Emma házára írt sorszám kivételével minden i sorszám pontosan egyszer jelenik meg, az értékelő által meghatározott sorrendben.

2. fázis

A programodnak egy N darab egész számot tartalmazó sort kell beolvasnia: $A_0, A_1, \ldots, A_{N-1}$, ahol A_i az i. ház ajtajára írt szám.

Ezután ki kell írnia egy sort, amely két egész számot tartalmazzon: s_1-t és s_2-t $(0 \leq s_i < N)$, a kitalált indexeket. Az s_1 és s_2 értékek megegyezhetnek.

Végrehajtás részletei

Vedd figyelembe, hogy a program futtatásakor a 2. fázisban a program újraindul. Ez azt jelenti, hogy a futtatások között nem tudsz változókban elmenteni információkat.

Ügyelj arra, hogy a kérdésfeltevés után a standard kimenetet ki kell üríteni, különben a programod időlimit túllépésként (Time Limit Exceeded) értékelhetik. Pythonban a `print()` automatikusan megteszi ezt. C++-ban a `cout << endl;` is kiürít, egy új sor kiírásával; ha `printf()` függvényt használsz, akkor add ki az `fflush(stdout)` parancsot.

A feladat értékelője **adaptív**, ami azt jelenti, hogy a programod kimenetétől függően megváltoztathatja a viselkedését, hogy megakadályozza a heurisztikus megoldások átjutását. Az értékelő lehet, hogy elvégzi az 1. fázis próbafuttatását, majd megnézi az általad írt kimenetet, majd az előző futtatásból kinyert információk felhasználásával újra lefuttatja az 1. fázist.

A **Te programodnak determinisztikusnak *kell* lennie**, azaz ugyanúgy kell viselkednie, ha kétszer ugyanazzal a bemenettel futtatod. **Ha véletlenszerűséget** akarsz használni a programodban, mindenképpen használd az **`srand()`** függvényt (C++-ban) vagy a `random.seed` (Pythonban) vagy, a C++11 véletlenszám-generátor használata esetén a mag megadását a véletlenszám-generátor használatakor. Figyelj arra, hogy az a `srand (time(NULL))` nem használható C++-ban.

Ha az értékelő azt észleli, hogy a programod nem determinisztikus, akkor a programod *Wrong Answer* ítéletet kap.

Ha a programod (legfeljebb 3) különálló futási idejeinek *összege* meghaladja az időkorlátot, akkor a beadás a *Time Limit Exceeded* minősítést kapja.

Pontozás

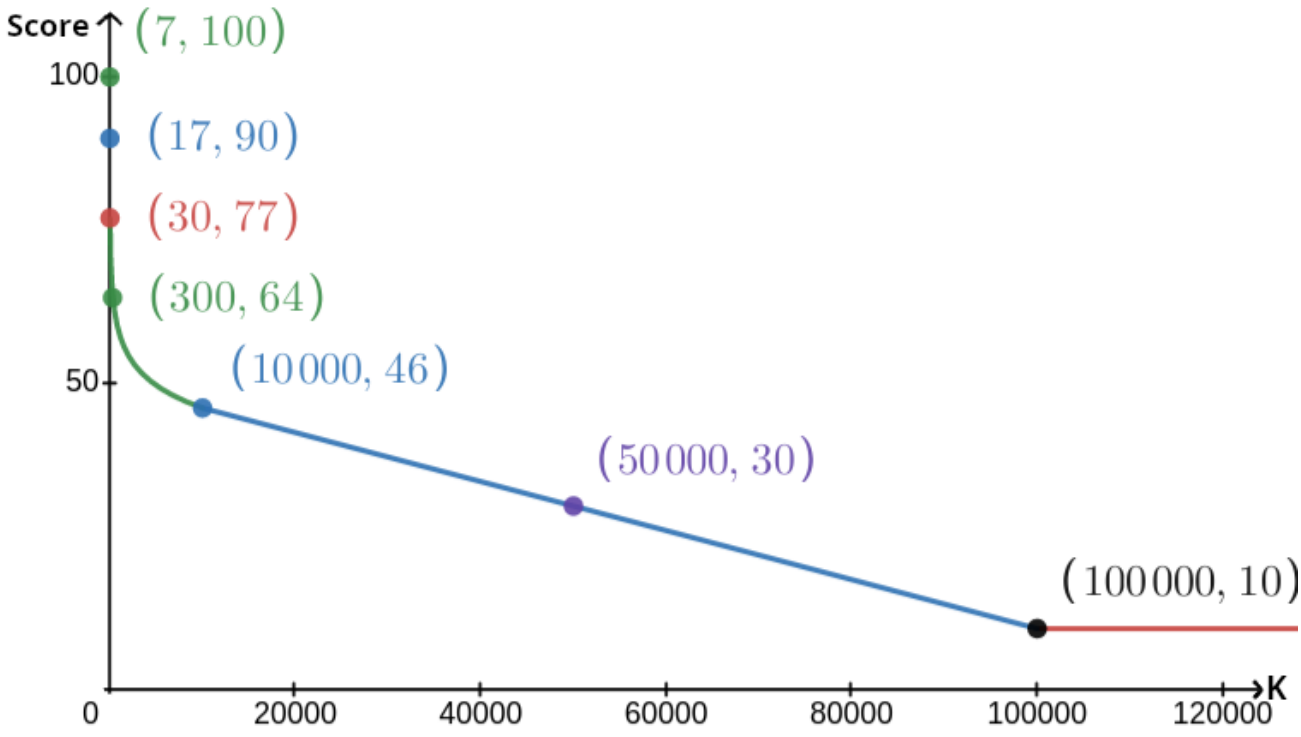
A megoldásodat számos teszteseten teszteli az értékelő. Ha a megoldásod bármelyik tesztesetnél hibázik (pl. rossz választ ír (Wrong Answer), összeomlik (Run-Time Error), túllépi az időkorlátot (Time Limit Exceeded) stb.), akkor 0 pontot kapsz.

Ha a programod sikeresen megtalálta Emma házának sorszámát *minden* tesztesetre, akkor az Elfogadva (Accepted) ítéletet és a következőképpen kiszámított pontszámot kapod.

Legyen K_{max} a tesztesetekhez használt K-k maximális értéke. Ekkor K_{max} függvényében:

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

A pontozási függvényt az alábbi ábra mutatja.



A mintatesztet a pontozásnál figyelmen kívül hagyjuk, és a megoldásnak nem kell működnie rajta.

Tesztelő eszköz

A megoldásod tesztelésének megkönnyítése érdekében egy egyszerű eszközt biztosítunk, amelyet letölthetsz. Ezt a feladatoldal alján, az "attachments" menüpont alatt találod. Az eszköz használata opcionális, és azt szabadon megváltoztathatod. Vedd figyelembe, hogy a végső értékelő eltér a tesztelő eszköztől.

Példahasználat ($N = 4$, $s = 2$, ahol s az utolsónak meglátogatott házra írt szám):

Python-ban a `solution.py` program (általában `pypy3 solution.py` helyett):

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

C++-ban először fordítsd le a programod (például a `g++ -g -g -O2 -std=gnu++17 -static solution.cpp -o solution.out` paranccsal) és utána futtasd:

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

A tesztelőeszköz véletlenszerű sorrendben látogatja meg a házakat. Ha egy meghatározott sorrendet szeretnél használni, módosítsd a tesztelő eszközt ott, ahol a "MODIFY HERE" áll.

Minta interakció

A mintatesztet a pontozásnál figyelmen kívül hagyjuk, és a megoldásnak nem kell működnie rajta.

Tegyük fel, hogy $N = 4$, és Emma az 1 házban lakik. Legyen A a házakra írt számok listája. Kezdetben $A = [0, 0, 0, 0, 0]$, ahol a 0 azt jelenti, hogy a megfelelő házra nincs szám írva.

A kód első futtatásakor:

$N = 4$ adott. A megoldásod $K = 3$ értékkel válaszol.

A_2-t kérdezünk. A megoldásod 3-t ad. A értéke most $[0, 0, 3, 0]$.

A_0-t kérdezzük. A megoldásod 1. A értéke most $[1, 0, 3, 0]$.

A_3-t kérdezzük. A megoldásod válasza 2. A értéke most $[1, 0, 3, 2]$.

Végül az értékelő $A_1 = 2$-t állít be, így A = $[1, 2, 3, 2]$ lesz a végén. Ezzel az első fázis véget ért.

A második fázisban a megoldásod megkapja a `1 2 3 2` listát.

Erre a `1 3` választ adja.

Mivel az egyik válasz a ház (1) helyes sorszáma, Anna és Bertil megnyeri a játékot.

értékelő kimenete	programod kimenete
1 4	
	3
2	
	3
0	
	1
3	
	2

értékelő kimenete	programod kimenete
2 4	
1 2 3 2	
	1 3