D. Guessing Game | Spėliojimo žaidimas

Užduoties pavadinimas	Spėliojimo žaidimas
Laiko limitas	4 sekundės
Atminties limitas	1 gigabaitas

Lundo senamiestyje yra gatvė su N namų eile, namai pažymėti indeksais nuo 0 iki $N-1$. Viename iš šių namų gyvena Ema, o jos draugai Ana ir Bertilas nori išsiaiškinti, kuriame. Užuot tiesiog pasakiusi draugams, kur ji gyvena, Ema nusprendžia su jais pažaisti žaidimą. Prieš prasidedant žaidimui Ana ir Bertilas žino tik namų skaičių gatvėje. Šiuo metu Ana ir Bertilas gali pasirinkti teigiamą sveikąjį skaičių K ir susitarti dėl strategijos. Vėliau bet koks bendravimas draudžiamas.

Žaidimą sudaro du etapai. Pirmajame etape Ema pasirenka namų lankymo tvarką taip, kad jos namas būtų aplankytas paskutinis. Tada ji veda Aną į namus vieną po kito šia tvarka, iš anksto nepasakydama Anai eilės tvarkos. Ant kiekvieno namo, kuris nėra Emos namas, Ana gali kreida užrašyti vieną sveikąjį skaičių nuo 1 iki K ant namo durų. Prie paskutiniojo jų aplankyto namo, Emos namo, Ema pati ant durų užrašo sveikąjį skaičių nuo 1 iki K.

Antrajame žaidimo etape Bertilas eina gatve ir perskaito visus Anos ir Emos ant durų užrašytus skaičius. Dabar jis nori atspėti, kuriame name gyvena Ema. Jis turi dvi galimybes atspėti teisingai ir, jei jam pavyks, jis ir Ana laimės žaidimą. Priešingu atveju žaidimą laimi Ema.

Ar galite sukurti strategiją, pagal kurią Ana ir Bertilas garantuotai laimėtų žaidimą? Jūsų strategija bus vertinama pagal K vertę (kuo mažesnė, tuo geriau).

Užduoties vykdymas

Tai yra kelių paleidimų (multi-run) užduotis, t. y. jūsų programa bus vykdoma kelis kartus. Pirmą kartą paleidus programą bus įgyvendinta Anos strategija, antrą kartą – Bertilo strategija.

Pirmoje įvesties eilutėje bus du sveikieji skaičiai P ir N, kur P yra 1 arba 2 (pirmasis arba antrasis etapas), o N – namų skaičius. **Išskyrus pavyzdines įvestis (nenaudojamas vertinimui), N visada bus lygus** $100\,000$.

Tolimesni pradiniai duomenys priklauso nuo etapo:

1 etapas

Jūsų programa turėtų prasidėti išvedant skaičių K vienoje eilutėje $(1 \leq K \leq 1\,000\,000)$. Tada $N-1$ kartų ji turėtų nuskaityti eilutę su indeksu i $(0 \leq i < N)$ ir išvesti eilutę su skaičiumi A_i $(1 \leq A_i \leq K)$, kur A_i yra skaičius, kurį Ana užrašė ant i namo durų. Kiekvienas indeksas i, išskyrus Emos namo indeksą, bus išvedamas lygiai vieną kartą tam tikra vertinimo programos (grader) nustatyta tvarka.

2 etapas

Jūsų programa turėtų nuskaityti eilutę su N sveikųjų skaičių: $A_0, A_1, ..., A_{N-1}$, kur A_i yra skaičius, užrašytas ant i namo durų.

Tada ji turėtų išspausdinti eilutę su dviem sveikaisiais skaičiais s_1 ir s_2 $(0 \leq s_i < N)$, spėjamais indeksais. Leidžiama, kad s_1 ir s_2 būtų lygūs.

Vykdymo detalės

Atkreipkite dėmesį, kad paleidus programą 2 etape, programa paleidžiama iš naujo. Tai reiškia, kad tarp paleidimų negalėsite išsaugoti informacijos kintamuosiuose.

Po kiekvienos išspausdintos eilutės būtinai išvalykite (flush) standartinę išvestį, kitaip jūsų programa gali būti įvertinta verdiktu Time Limit Exceeded. Python kalboje `print()` tai padaro automatiškai. C++ kalboje `cout << endl;` ne tik perkelia į kitą eilutę, bet ir išvalo standartinę išvestį; jei naudojate printf, naudokite `fflush(stdout)`.

Vertinimo programa gali **prisitaikyti**, t. y. gali keisti savo elgseną priklausomai nuo jūsų programos išvesties, kad užkirstų kelią euristiniams (netiksliems arba spėjimo) sprendimams. Ji gali paleisti 1 etapą, peržiūrėti į jūsų išvestį ir tada vėl paleisti 1 etapą, naudojant informaciją, gautą iš pirmojo paleidimo.

Jūsų programa *turi* būti deterministinė, t. y. abu kartus elgtis taip pat, jei du kartus paleidžiama su ta pačia įvestimi. Jei programoje norite naudoti atsitiktinumą, būtinai naudokite fiksuotą random seed. Tai galima padaryti perduodant įkoduotą (hard-coded) konstantą į `srand` (C++ kalba) arba `random.seed` (Python kalba), arba, jei naudojate C++11 atsitiktinių skaičių generatorius, nurodant seed, kai konstruojamas atsitiktinių skaičių generatorius. Ypač negalima naudoti `srand(time(NULL))` C++ kalboje. Jei vertinimo programa aptiks, kad jūsų programa nėra deterministinė, jai bus suteiktas Wrong Answer verdiktas.

Jei (iki 3) atskirų jūsų programos paleidimų trukmių *suma* viršys laiko limitą, jūsų pateikimas bus įvertintas kaip Time Limit Exceeded.

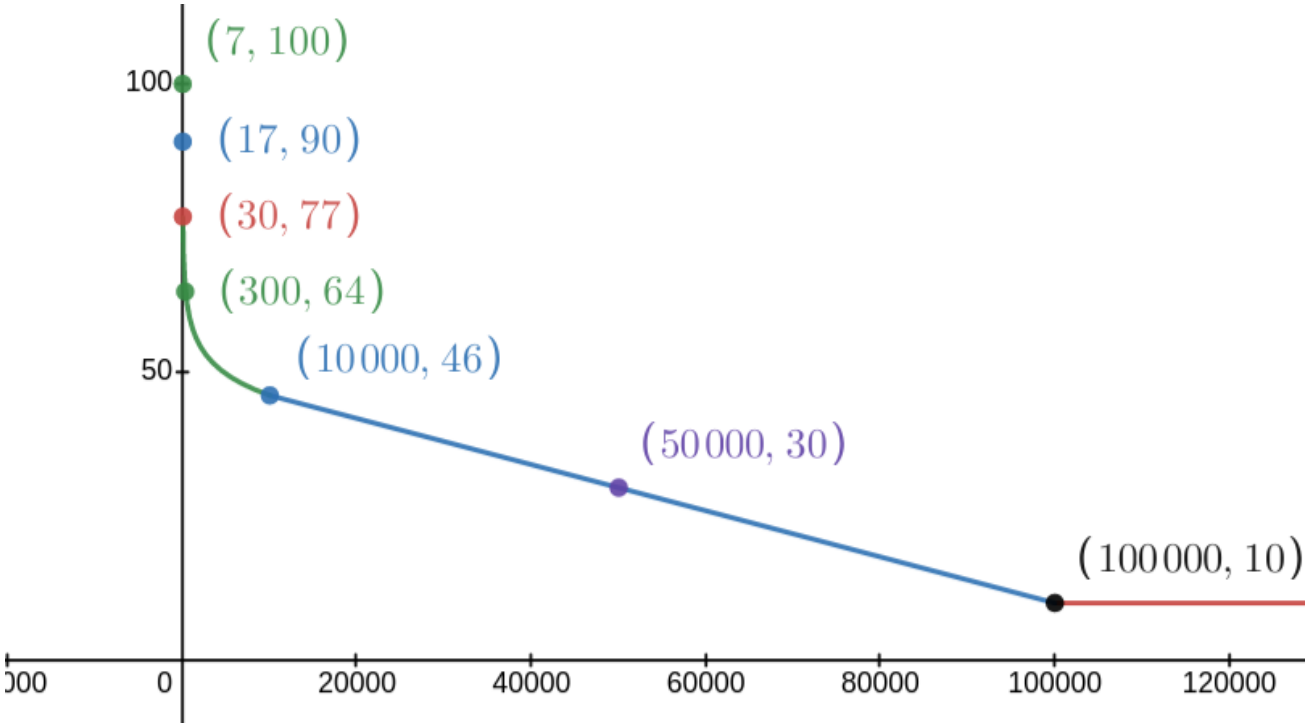
Vertinimas

Jūsų sprendimas bus testuojamas su keliais testavimo atvejais. Jei jūsų sprendimas bus neteisingas *bet kuriame* iš šių testų (pvz., bus pateikti neteisingi atsakymai (Wrong Answer), programa nustos veikti (Run-Time Error), bus viršytas laiko limitas (Time Limit Exceeded) ir t. t.), gausite 0 taškų ir atitinkamą verdiktą.

Jei jūsų programa sėkmingai suras Emos namo indeksą *visuose* testavimo atvejuose, gausite verdiktą Accepted ir rezultatą, apskaičiuotą taip, kaip aprašyta žemiau. Tegul K_{max} yra didžiausia K_{max} reikšmė, naudojama bet kuriam testavimo atvejui. Priklausomai nuo K_{max}:

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

Toliau pateiktame paveikslėlyje pavaizduota vertinimo funkcija.



Į pavyzdinį testavimo atvejį neatsižvelgiama vertinant taškais, ir jūsų sprendimas neprivalo su juo veikti.

Testavimo įrankis

Kad būtų lengviau testuoti savo sprendimą, pateikiame paprastą įrankį, kurį galite atsisiųsti. Žiūrėkite "attachments" Kattis problemos puslapio apačioje. Įrankiu naudotis neprivaloma, ir jums leidžiama jį keisti. Atkreipkite dėmesį, kad oficiali Kattis vertinimo programa skiriasi nuo testavimo įrankio.

Panaudojimo pavyzdys (kai $N = 4$, $s = 2$, kur s yra skaičius, užrašytas ant paskutinio aplankyto namo):

Python programose `solution.py` (paprastai paleidžiama kaip `pypy3 solution.py`):

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

C++ programas pirmiausia kompiliuokite (pvz., naudodami `g++ -g -O2 -std=gnu++17 -static solution.cpp -o solution.out`) ir tada paleiskite:

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

Testavimo įrankis aplankys namus atsitiktine tvarka. Jei norite naudoti konkrečią tvarką, pakeiskite testavimo įrankį ten, kur parašyta "MODIFY HERE".

Pavyzdinė interakcija

Į pavyzdinį testinį atvejį neatsižvelgiama vertinant balais, todėl jūsų sprendimas neturi su juo veikti.

Tarkime, kad turime $N = 4$ ir kad Ema gyvena 1 name. Tegul A būna ant namų užrašytų skaičių sąrašas. Iš pradžių $A = [0, 0, 0, 0, 0]$, kur 0 reiškia, kad ant atitinkamo namo nėra užrašytas nė vienas skaičius.

Pirmame jūsų kodo paleidime:

Paduodama $N = 4$. Jūsų sprendimas atsako su $K = 3$.

Prašoma pateikti A_2. Jūsų sprendimas atsako su 3. A dabar yra $[0, 0, 3, 0]$.

Prašoma pateikti A_0. Jūsų sprendimas atsako su 1. A dabar yra $[1, 0, 3, 0]$.

Prašoma pateikti A_3. Jūsų sprendimas atsako su 2. A dabar yra $[1, 0, 3, 2]$.

Galiausiai vertintojas nustato $A_1 = 2$, todėl galiausiai A = $[1, 2, 3, 2]$. Tai žymi pirmojo etapo pabaigą.

Antrajame kodo paleidimo etape jūsų sprendimui perduodamas sąrašas `1 2 3 2`.

Jis atsako su `1 3`.

Kadangi vienas iš spėjimų yra teisingas namo indeksas (1), Ana ir Bertilas laimi žaidimą.

Vertinimo programos išvestis	Jūsų išvestis
1 4	
	3
2	
	3
0	
	1
3	
	2

Vertinimo programos išvestis	Jūsų išvestis
2 4	
1 2 3 2	
	1 3