D. Игра на погодување

Име на проблемот	Игра на погодување
Временско ограничување	4 секунди
Мемориско ограничување	1 гигабајт

Во стариот дел на градот Лунд, има една улица со N куќи во низа, индексирани од 0 до $N-1$. Ема живее во една од овие куќи, а нејзините пријатели Ана и Бертил сакаат да откријат во која. Наместо да им каже каде живее, Ема одлучува да игра игра со нив. Пред да започне играта, Ана и Бертил знаат само колку куќи има на улицата. Во овој момент, Ана и Бертил можат да изберат позитивен цел број K и да се договорат за стратегија. Секоја комуникација после тоа е забранета.

Самата игра се состои од две фази. Во првата фаза, Ема бира редослед во кој ќе ги посети куќите, така што нејзината куќа ќе биде последна за посета. Потоа ја води Ана по куќите една по една во тој редослед, без однапред да ѝ го каже на Ана редоследот. За секоја куќа која не е Емината куќа, Ана има право да напише еден цел број меѓу 1 и K на вратата на куќата со парче креда. За последната куќа која ќе ја посетат, Емината куќа, Ема самата на вратата запишува цел број меѓу 1 и K.

Во втората фаза на играта, Бертил оди по улицата од куќа 0 до куќа $N-1$ и ги чита сите броеви напишани на вратите од Ана и Ема. Сега Бертил сака да погоди во која куќа живее Ема. Има две шанси да погоди точно, и ако успее, Ана и Бертил победуваат во играта. Во спротивно, Ема победува.

Можете ли да измислите стратегија при која Ана и Бертил се сигурни дека ќе победат во играта? Вашата стратегија ќе биде оценета на база на вредноста на K (што помала, толку подобро).

Имплементација

Ова е проблем со повеќекратно извршување, што значи дека вашиот програма ќе биде извршена повеќе пати. Првиот пат кога ќе биде извршена, ќе ја имплементира стратегијата на (ќе се изврши кодот за) Ана. Потоа ќе биде извршен кодот за Бертил.

Првиот ред од влезот ќе содржи два цели броја, P и N, каде P е или 1 или 2 (првата или втората фаза), а N е бројот на куќи. **Освен кај примерот за влез (не се користи за оценување), N секогаш ќе биде $100\,000$.**

Следниот влез зависи од фазата:

Фаза 1

Вашата програма треба да почне со испишување на бројот K во еден ред $(1 \leq K \leq 1\,000\,000)$. Потоа, $N-1$ пати, треба да прочитате ред кој содржи индекс i $(0 \leq i < N)$, и да испишете ред со цел број A_i $(1 \leq A_i \leq K)$, каде A_i е бројот што Ана го пишува на вратата на куќата i. Секој индекс i, освен индексот на Емината куќа, ќе се појави точно еднаш, во редослед одреден од грејдерот.

Фаза 2

Вашата програма треба да прочита ред со N цели броеви, $A_0, A_1, \ldots, A_{N-1}$, каде A_i е бројот што е напишан на вратата на куќата i.

Потоа, треба да испише ред со два цели броеви, s_1 и s_2 $(0 \leq s_i < N)$, што се индексите - погодувања за куќата. s_1 и s_2 може да бидат и меѓусебно еднакви.

Детали за имплементација

Забележете дека кога ќе се изврши вашата програма во Фаза 2, програмата се рестартира. Ова значи дека не можете да ги зачувате информациите во некои променливи помеѓу две извршувања.

Откако ќе го испишете секој ред, направете flush на излезниот поток (standard output), инаку вашата програма може да добие Time Limit Exceeded. Во C++, `cout << endl;` покрај принтањето на знак за нов ред прави и flush, а ако користите printf, користете `fflush(stdout)`.

Грејдерот за овој проблем може да биде **адаптивен**, што значи дека може да го смени своето однесување во зависност од излезот на вашиот програма за да спречи решенија со хевристики да поминат. Тој може да изведе пробно извршување на Фаза 1, да го анализира вашиот излез, а потоа вистински да ја изврши Фаза 1 користејќи информации добиени од претходното извршување.

Вашата програма мора да биде детерминистичка, што значи, да се однесува исто доколку се изврши двапати врз ист влез. Ако сакате да користите случајни броеви во вашиот програма, осигурете се дека користите фиксирано семе (seed) за случајни броеви. Ова може да се направи со пренесување на фиксирана константа во `srand` (во C++) или random.seed (во Python), или, ако користите генератори на случајни броеви во C++11, со наведување на

семето при создавањето на генераторот на случајни броеви. Особено, не можете да користите srand(time(NULL)) во C++. Ако грејдерот открие дека вашата програма не е детерминистичка, ќе добиете Wrong Answer.

Ако *збирот* на времињата на извршувањето на (до 3) посебни извршувања на вашиот програма го надмине временското ограничување, ќе добиете Time Limit Exceeded.

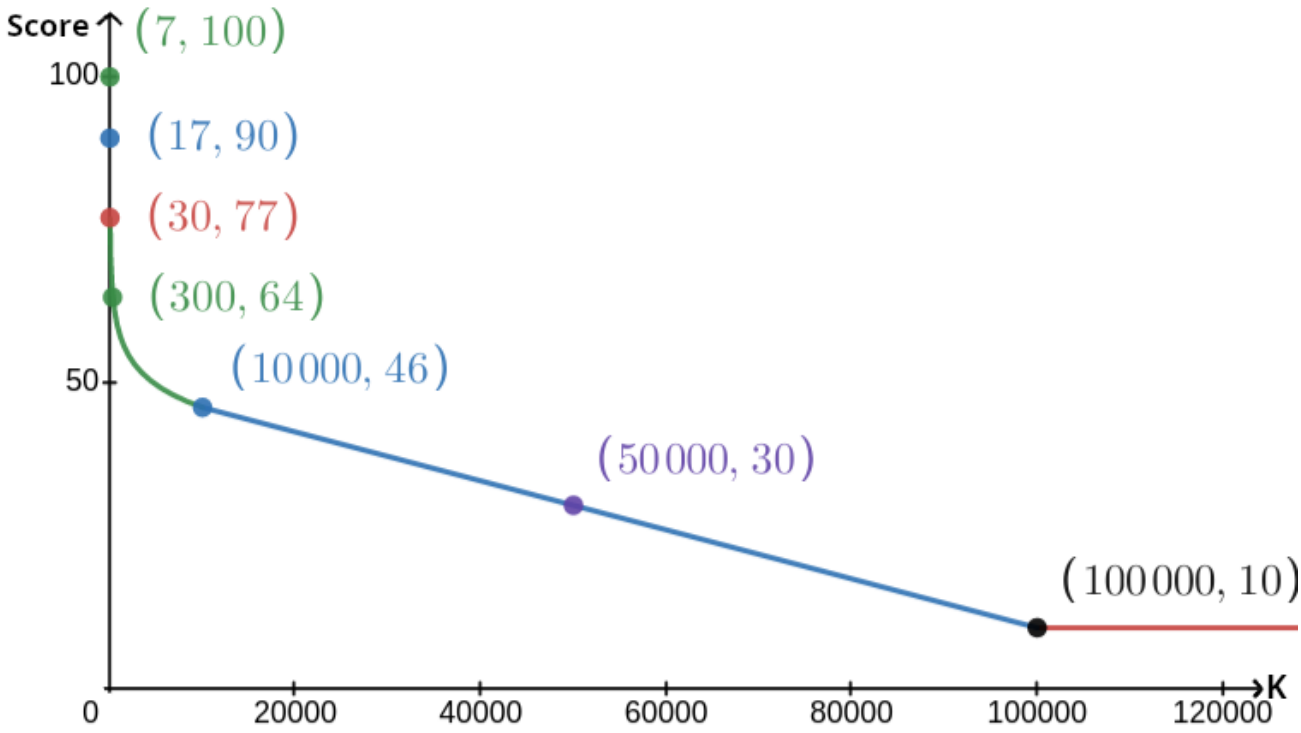
Оценување

Вашето решение ќе биде тестирано на неколку тест случаи. Ако вашето решение падне на *кој било* од овие тест случаи (на пример, дава погрешни одговори (Wrong Answer), се срушува (Run-Time Error), надминува временското ограничување (Time Limit Exceeded), итн.), ќе добиете 0 поени и соодветна одлука.

Ако вашата програма успешно го најде индексот на Емината куќа за сите тест случаи, ќе добиете одлука Прифатено (Accepted), и оценка пресметана како следи. Нека K_{max} биде максималната вредност на K што се користи во кој билотест случај. Во зависност од K_{max}:

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

Функцијата на оценување е претставена на сликата подолу.



Тест случајот од примерот се игнорира при оценувањето и вашето решение не мора да работи за него.

Алатка за тестирање

Како поддршка при тестирањето на вашето решение, ви обезбедивме едноставна алатка којашто можете да си ја преземете. Погледнете го делот “attachments” на дното од проблемската страница на системот Kattis. Алатката е опционална за користење и дозволено е да ја менувате. Да забележиме дека официјалната програма-оценувач на Kattis е различна од оваа алатка за тестирање.

Пример за користење (со $N = 4$, $s = 2$, каде s е бројот запишан на последната посетена куќа):

За C++ програми, прво компајлирате го (e.g. with `g++ -g -O2 -std=gnu++17 -static solution.cpp -o solution.out`) и потоа извршете:

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

Алатката за тестирање ќе ги посети куќите по случаен редослед. За да примените специфичен редослед, модифицирајте ја алатката онаму кадешто пишува “MODIFY HERE”.

Пример за интеракција

Тест случајот од примерот се игнорира при оценувањето и вашето решение не мора да работи за него.

Да претпоставиме дека $N = 4$ и дека Ема живее во куќата 1. Нека A е листата од броеви запишани на куќите. На почеток, $A = [0, 0, 0, 0]$, каде 0 означува дека не е запишан ниту еден број на соодветната куќа.

Во првото извршување на вашиот код:

$N = 4$ е дадено. Вашето решение одговара со $K = 3$.

A_2 се бара. Вашето решение одговара со 3. A сега е $[0, 0, 3, 0]$.

A_0 се бара. Вашето решение одговара со 1. A сега е $[1, 0, 3, 0]$.

A_3 се бара. Вашето решение одговара со 2. A сега е $[1, 0, 3, 2]$.

Конечно, оценувачот поставува $A_1 = 2$, така што $A = [1, 2, 3, 2]$ на крајот. Ова значи крај на првата фаза.

Во Фаза 2 од вашиот код, на вашето решение се предава листата `1 2 3 2`.

Тоа одговара со `1 3`.

Бидејќи едно од погодувањата е точниот индекс на куќата (1), Ана и Бертил се победници на играта.

излез од оценувач	ваш излез
1 4	
	3
2	
	3
0	
	1
3	
	2

излез од оценувач	ваш излез
2 4	
1 2 3 2	
	1 3