Скеч шоу

Фарук се подготвя за своя скеч от месеци, но все още не е намерил място за представлението. Той иска всички в Сараево да видят неговата изява, затова решава, че ще я изпълни в планината Требевич.

Фарук не иска мястото да бъде твърде високо или твърде ниско: иска надморската височина да е точно x. Планината се представя чрез матрица от височини A с размери $N \times N$. Всеки ред и всяка колона на A е ненамаляваща.

За да намери подходящо място, той може да пита екскурзовода си за височина на дадена точка (i, j). Тъй като е стеснителен, той иска да минимизира броя на въпросите, които задава на екскурзовода.

Задачата има две независими подзадачи, които са за 30 и 70 точки. Изпратете един файл, `sketch.cpp`, който имплементира трите функции `findVenue`, `init` и `query`, описани по-долу. Частично решение все пак трябва да съдържа всички три функции, за да се компилира програмата.

Подзадачи

Подзадача 1 (30 точки)

Фарук иска да знае дали се съдържа височина x в матрицата A. В началото той не знае никакви стойности на матрицата, но може да прави извиквания:

```
int ask(int i, int j);
```

Тази функция връща стойността $A[i][j]$. Индексите трябва да изпълняват $0 \leq i, j < N$.

Имплементирайте:

```
bool findVenue(int N, int x);
```

Върнете `true` само ако има клетка на матрицата със стойност x.

Ограничения

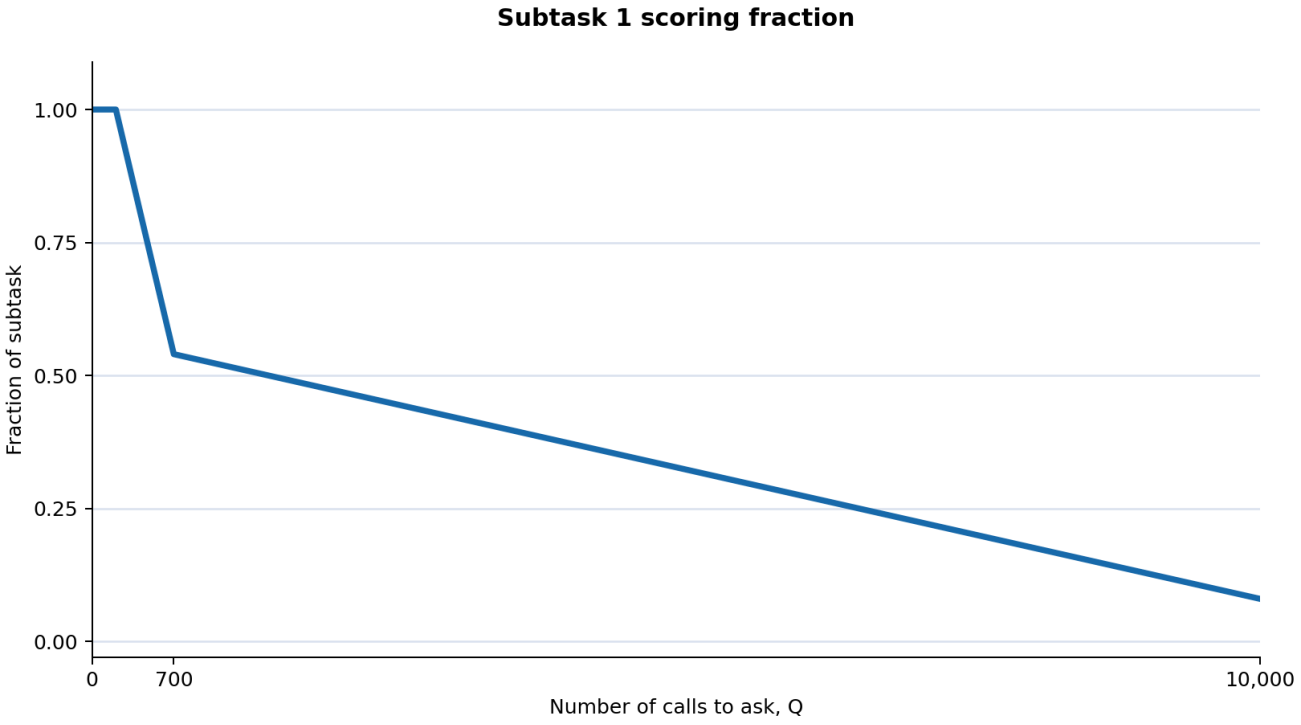
- $2 \le N \le 100$
- $1 \le x \le 10^9$
- $1 \le A[i][j] \le 10^9$
- $A[i][j] \le A[i][j+1]$ за всяко $0 \le i < N$, $0 \le j < N-1$.
- $A[i][j] \le A[i+1][j]$ за всяко $0 \le i < N-1$, $0 \le j < N$.
- Може да извиквате функцията `ask` най-много 10^4 пъти.

Оценяване

Нека Q е максималният брой извиквания на функцията `ask` измежду всички тестове. Ако `findVenue` върне неправилна стойност за някой тест или има невалидно извикване на `ask`, ще получите 0 точки за подзадачата.

В противен случай, оценката е $S_1(Q) \cdot 30$, където $S_1(Q)$ е дефинирано по следния начин:

- $S_1(Q) = 0$, ако $Q > 10^4$.
- $S_1(Q) = \frac{27 - \frac{23(Q-700)}{9300}}{50}$, ако $700 < Q \le 10^4$.
- $S_1(Q) = \frac{50 - \frac{23(Q-204)}{496}}{50}$, ако $204 < Q \le 700$.
- $S_1(Q) = 1$, ако $Q \le 204$.



Обърнете внимание, че грейдърът е адаптивен.

Подзадача 2 (70 точки)

В тази подзадача вие имплементирате ролята на екскурзовода. Екскурзоводът се наслаждава да говори на Фарук и иска този разговор да продължи възможно най-дълго. За да не го разочарова, екскурзоводът трябва да е сигурен, че матрицата винаги ще съдържа височина x; Фарук спира да задава въпроси само след като намери такава височина.

Ескурзоводът може да избира височините адаптивно вместо да е избрал матрицата предварително. Въпреки това всеки отговор трябва да е консистентен с поне една валидна ненамаляваща матрица, която съдържа x.

Детайли по имплементацията

За тази подзадача имплементирайте:

```
void init(int N, int x);
int query(int i, int j);
```

Грейдърът извиква функция `init(N, x)` в началото на всяка интеракция. Функцията `init` може да се извиква много пъти в рамките на един и същ тест. Всяко извикване започва нова интеракция. След това функцията `query(i, j)` трябва да връща отговора на екскурзовода за клетката (i, j).

За една интеракция отговорите на екскурзовода трябва да спазват следните правила:

- Едни и същи заявки за клетка трябва да получат един и същ отговор.
- Всеки отговор трябва да е цяло число в интервала $[1, 10^9]$.
- Всички отговори трябва да са консистентни с ненамаляващи редове и колони.
- Трябва да има валидна матрица консистентна с всички предишни отговори, която съдържа x.

Ограничения

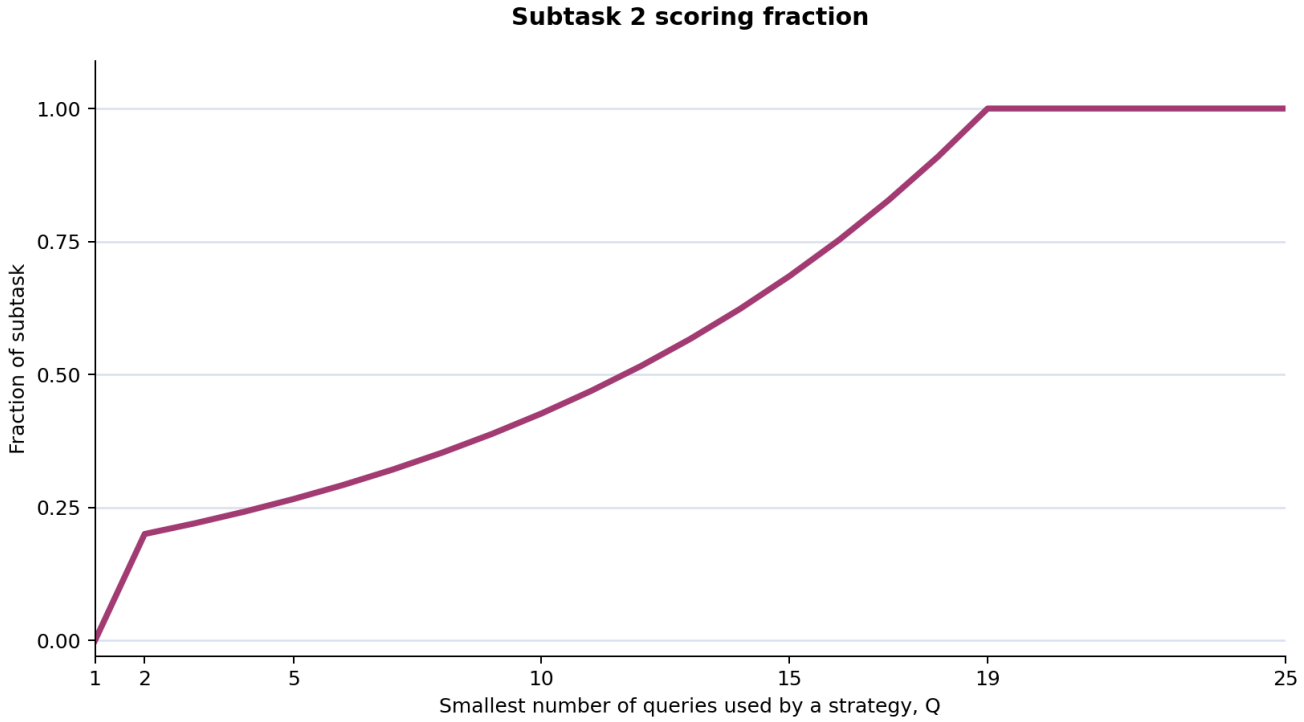
- $2 \leq N \leq 10$
- $2 \leq x \leq 10^9 - 1$

Оценяване

Грейдърът изпълнява няколко стратегии на Фарук. Обърнете внимание, че дадена стратегия може да е адаптивна. Всяко изпълняване на стратегия представлява една интеракция. Ако при някоя интеракция указаните правила се нарушат, получавате 0 точки за подзадачата. В противен случай, нека Q е минималният брой извиквания на функцията `query` измежду всички интеракции.

Точките, които получавате са $S_2(Q) \cdot 70$, където $S_2(Q)$ е дефинирано по следния начин:

- $S_2(Q) = 0$, ако $Q = 1$.
- $S_2(Q) = \min\left(1, \frac{1}{5}e^{\frac{\ln 5}{17}(Q-2)}\right)$, ако $Q \geq 2$.



Примери

Подзадача 1

Нека $N = 4$, $x = 7$ и скритата матрица е:

$$\begin{matrix} 1 & 3 & 5 & 8 \\ 2 & 4 & 7 & 9 \\ 3 & 6 & 10 & 12 \\ 5 & 8 & 11 & 15 \end{matrix}$$

Една възможна интеракция е:

Извикване	Върната стойност	Обяснение
`ask(2, 0)`	`3`	Стойността е по-малка от x.
`ask(0, 3)`	`8`	Стойността е по-голяма от x.
`ask(1, 2)`	`7`	Открита е търсената височина.

Затова функцията `findVenue(4, 7)` връща `true`. За същата матрица и $x = 13$ функцията трябва да върне `false`.

Подзадача 2

Нека отново $N = 4$ и $x = 7$. Екскурзоводът може да избере да използва стратегия, за която всички отговори са винаги консистентни със следната матрица:

$$\begin{matrix} 1 & 3 & 5 & 8 \\ 2 & 4 & 7 & 9 \\ 3 & 6 & 10 & 12 \\ 5 & 8 & 11 & 15 \end{matrix}$$

Например екскурзоводът може да отговори по следния начин:

Извикване на Фарук	Върната стойност от екскурзовода
`query(2, 0)`	3
`query(0, 3)`	8
`query(1, 2)`	7

Друга стратегия може да намери x веднага:

Извикване на Фарук	Върната стойност от екскурзовода
`query(1, 2)`	7

Тази стратегия намира x след една заявка. Резултатът за тази интеракция ще е 0, понеже $S_2(1) = 0$.

Събмит

Започнете с предоставения файл `att/sketch.cpp`. Съдържа базови версии на трите функции. Важно е да запазите следния ред в началото на програмата ви:

```
#include "grader.h"
```

Не дефинирайте функция `main`, не четете от стандартния вход и не печатайте на стандартния изход във вашия събмит.

Локален грейдър

Директорията `att` съдържа началния файл `sketch.cpp` и примерния грейдър. Копирайте или редактирайте `sketch.cpp` за файла, който ще съдържа вашия събмит, след което компилирайте от същата директория:

```
g++ -std=c++20 -o sketch sketch.cpp grader.cpp
```

Примерният грейдър има два режима на работа. Първо прочита `N x`, след което анализира следващото въведено:

- Ако това е число, започва режима на работа за подзадача 1, защото това е първото число в матрицата.
- Ако това е `?` или `!`, започва режима на работа за подзадача 2, защото това са команди за интеракцията.

Режим на работа за подзадача 1

Въведете `N x`, след което N^2 стойности на матрицата. Примерният грейдър извиква `findVenue(N, x)` веднъж.

```
4 7
1 3 5 8
2 4 7 9
3 6 10 12
5 8 11 15
```

Ако отговорът е верен, то се отпечатва:

```
1
Q
```

където Q е броят на извикванията на `ask`. В противен случай се отпечатва `0` и обяснение.

Режим на работа за подзадача 2

Първо въведете `N x`. След това въвеждайте команди, по една на ред:

- `? i j` извиква `query(i, j)` и отпечатва върната височина.
- `!` приключва интеракцията. Въведете това само след като една от отпечатаните височини е равна на x.

Например, започнете интеракцията по следния начин:

```
4 7
? 0 0
```

Може да продължавате да въвеждате команди `? i j`. Когато примерният грейдър върне x, въведете `!`.

Примерният грейдър проверява всеки отговор веднага при въвеждането. При успешно завършване се отпечатва:

```
1
Q
P
```

където Q е броят на заявките и $P = S_2(Q)$ е дробта, която се получава за интеракцията. Отпечатва се `0` и обяснение, ако отговор не е валиден или ако се въведе `!` преди намирането на x.