

Sketch show

Faruk has been preparing his sketch show for months, but he has not found a venue yet. He wants everybody in Sarajevo to see the play, so he decides to host it on Trebević mountain.

Faruk does not want the venue to be too high up or too low down: he wants an altitude of exactly x . The mountain is represented by an $N \times N$ matrix A of altitudes. Every row and every column of A is nondecreasing.

To find the venue he can ask his tour guide for the altitude at a point (i, j) . Since he is socially anxious he wants to minimise the number of questions he asks the tour guide.

This task has two independent subtasks worth 30 and 70 points. Submit one file, `sketch.cpp`, which implements the functions `findVenue`, `init` and `query` described below. A partial solution must still provide all three functions so that it compiles.

Subtasks

Subtask 1 (30 points)

Faruk wants to know whether an altitude x occurs in A . He initially knows no matrix values, but may call:

Implementation

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

This returns $A[i][j]$. The indices must satisfy $0 \leq i, j < N$.

Implement:

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

Return `true` if and only if some cell of the matrix equals x .

Constraints

- $2 \leq N \leq 100$

- $1 \leq x \leq 10^9$
- $1 \leq A[i][j] \leq 10^9$
- $A[i][j] \leq A[i][j+1]$ for every $0 \leq i < N, 0 \leq j < N-1$.
- $A[i][j] \leq A[i+1][j]$ for every $0 \leq i < N-1, 0 \leq j < N$.
- You may call `ask` at most 10^4 times.

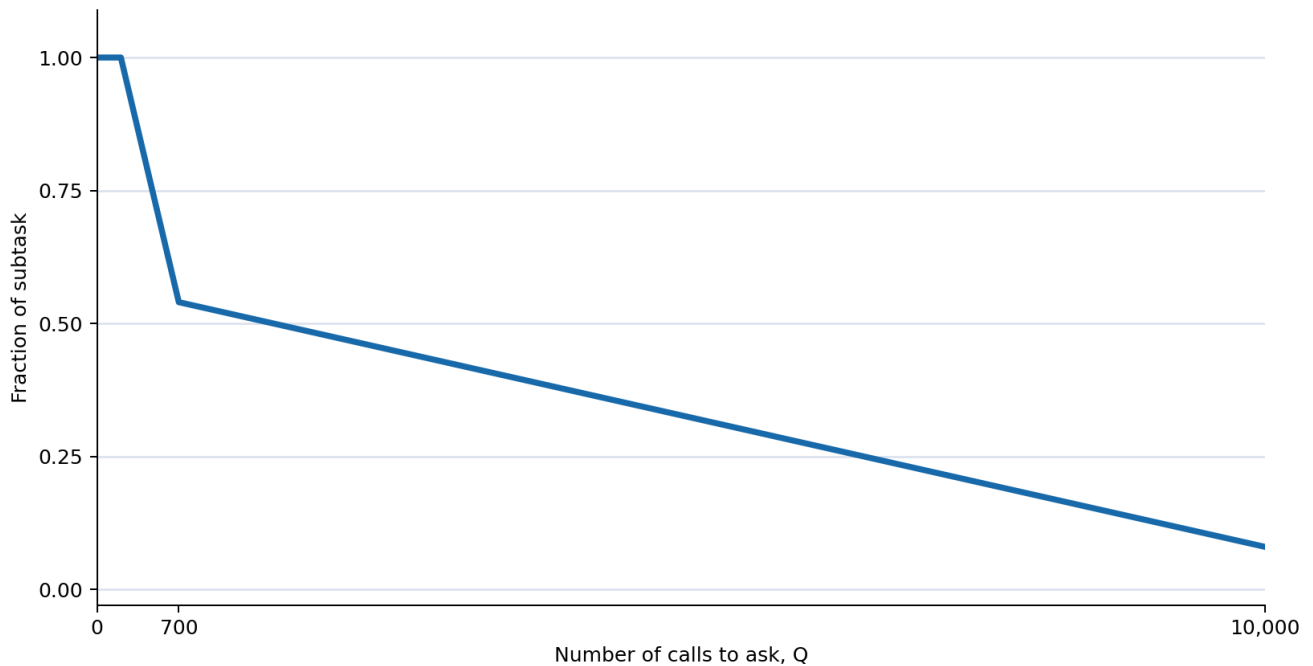
Scoring

Let Q be the maximum number of calls to `ask` over all test cases. If `findVenue` returns an incorrect value in any test case or an invalid call to `ask` is made, you receive 0 points for this subtask.

Otherwise, the score is calculated as $S_1(Q) \cdot 30$ where $S_1(Q)$ is defined as follows:

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

Subtask 1 scoring fraction



Note that the grader is adaptive.

Subtask 2 (70 points)

In this subtask, you implement the part of the tour guide instead. The guide enjoys talking to Faruk and wants to keep the conversation going for as long as possible. To avoid disappointing him, the guide must make sure that the matrix always contains a cell with altitude x ; Faruk stops asking questions only after he finds a suitable cell.

The guide may choose altitude replies adaptively rather than deciding the entire matrix in advance. However, every reply must remain consistent with at least one valid nondecreasing matrix that contains x .

Implementation

Implement:

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

The grader calls `init(N, x)` at the beginning of every interaction. It may call `init` multiple times during one test case. Each call begins a fresh interaction. Afterwards, `query(i, j)` must return the guide's answer for cell (i, j) .

For a single interaction the guide's answers must obey all of the following rules:

- Repeated queries for a cell receive the same answer.
- Every answer is an integer in $[1, 10^9]$.
- All answers are consistent with nondecreasing rows and columns.
- There is a valid matrix consistent with all previous answers, which contains x .

Constraints

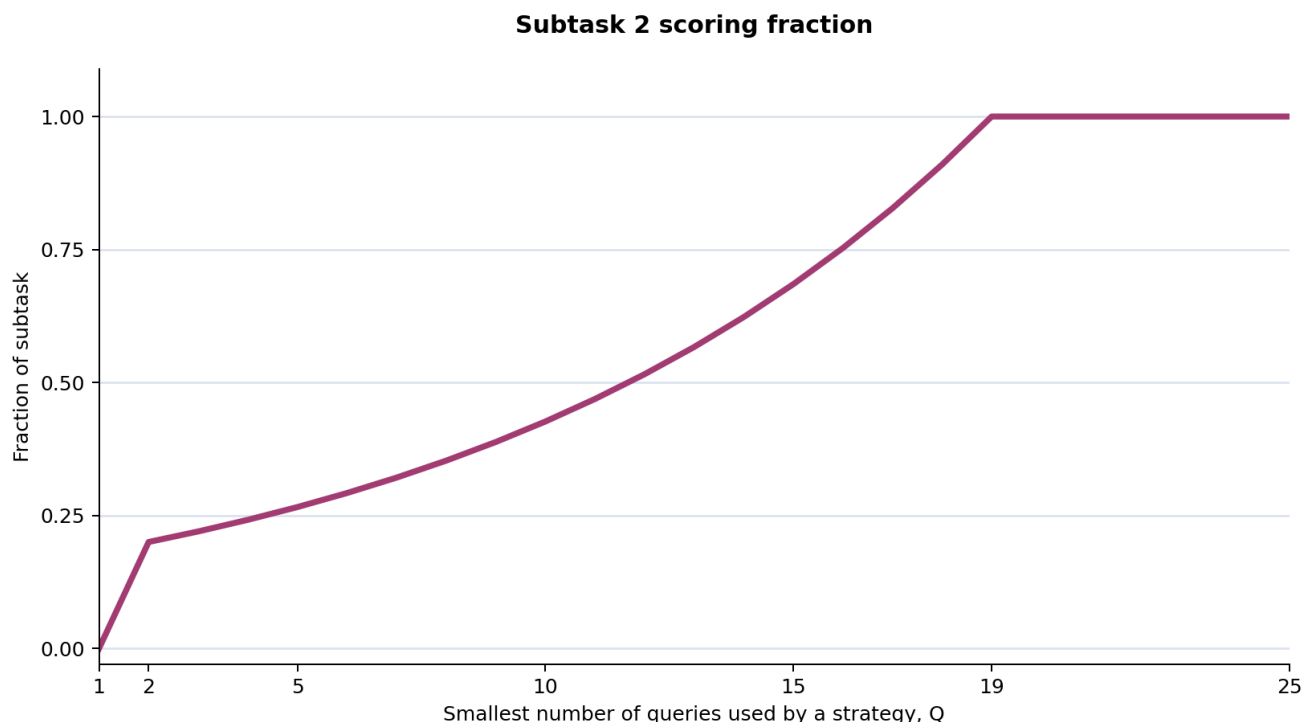
- $N = 10$
- $2 \leq x \leq 10^9 - 1$

Scoring

The grader runs several Faruk strategies. Note that a strategy can be adaptive. Each run of a strategy represents one interaction. If during one interaction the rules are broken, you receive 0 points for the subtask. Otherwise, let Q be the minimum number of calls to `query` over all interactions.

The number of points you get is $S_2(Q) \cdot 70$ where $S_2(Q)$ is defined as follows:

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



Examples

Subtask 1

Suppose $N = 4$, $x = 7$, and the hidden matrix is:

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

One possible interaction is:

Function call	Returned value	Explanation
<code>ask(2, 0)</code>	3	This value is smaller than x .
<code>ask(0, 3)</code>	8	This value is greater than x .
<code>ask(1, 2)</code>	7	The target altitude was found.

Therefore, `findVenue(4, 7)` returns `true`. For the same matrix and $x = 13$, it must return `false`.

Subtask 2

Suppose again that $N = 4$ and $x = 7$. A guide could decide to use a strategy where their answers are always consistent with the following matrix:

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

For example, the guide may answer as follows:

Faruk calls	Guide returns
<code>query(2, 0)</code>	3
<code>query(0, 3)</code>	8
<code>query(1, 2)</code>	7

Another strategy might find x immediately:

Faruk calls	Guide returns
<code>query(1, 2)</code>	7

This strategy finds x after one query. Its score for this interaction would be 0, since $S_2(1) = 0$.

Submission

Start with the supplied `att/sketch.cpp` file. It contains empty versions of all three functions. Fill them in and keep this line at the top of the file:

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

Do not define `main`, read standard input, or write standard output in your submission.

Sample Grader

The `att` directory contains the starter file `sketch.cpp` and the sample grader. Copy or edit `sketch.cpp` to contain your submission, then compile from that directory:

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

The sample grader has two modes. It reads `N x` first, then examines the next token:

- A number starts Subtask 1 mode, because it is the first matrix value.
- `?` or `!` starts Subtask 2 mode, because these are guide-interaction commands.

Subtask 1 mode

Provide N x , followed by all N^2 matrix values. The sample grader calls `findVenue(N, x)` once.

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

For a correct answer it prints:

```
1
Q
```

where Q is the number of calls to `ask`. Otherwise it prints 0 and an explanation.

Subtask 2 mode

First enter N x . Then enter commands, one per line:

- `? i j` calls `query(i, j)` and prints the returned altitude.
- `!` ends the interaction. Enter it only after one of the printed altitudes equals x .

For example, start an interaction like this:

```
4 7
? 0 0
```

Continue entering `? i j` commands as desired. When the sample grader prints x , enter `!`.

The sample grader checks every answer immediately. On successful completion it prints:

```
1
Q
P
```

where Q is the number of queries and $P = S_2(Q)$ is the fraction earned for that interaction. It prints 0 and an explanation if an answer is invalid or if `!` is entered before finding x .