Каменната къща

Арбен е наследил красив парцел земя сред хълмовете с изглед към Аргирокастро. Вдъхновен от прочутите каменни покриви на къщите в града, той планира да построи своя собствена традиционна къща.

Земята е представена като квадратна таблица $N \times N$, в която редовете и колоните са номерирани от 1 до N. Каменистият терен има една особеност: във всеки ред и във всяка колона на мрежата има точно по един голям, неподвижен камък.

За да положи основите на къщата си, Арбен трябва да избере **празна квадратна подтаблица** – област с размер $S \times S$, която не съдържа нито един камък.

Нека M е най-голямото цяло число, за което е гарантирано, че при всяко разположение на камъните, удовлетворяващо горните условия, съществува празна квадратна подтаблица с размер $M \times M$. Арбен иска да намери произволна празна квадратна подтаблица с размер поне $M \times M$.

За да огледа земята, Арбен моли своя приятел Беса за помощ. Беса е донесъл специално геодезическо устройство, наречено **Gurëmatës** ("брояч на камъни").

С една операция Беса може да настрои Gurëmatës да сканира правоъгълна подтаблица, зададена чрез горния ѝ ляв ъгъл (x_1, y_1) и долния ѝ десен ъгъл (x_2, y_2). Устройството връща точния брой на камъните в тази област. Gurëmatës се захранва от малка батерия и всяко сканиране консумира много енергия, затова Беса трябва да го използва възможно най-малко пъти.

Напишете програма, която използва Gurëmatës, за да намери празна квадратна подтаблица с размер $S \times S$, където $S \geq M$, и да докладва нейното местоположение.

Детайли по имплементацията

Трябва да имплементирате следната функция:

```
std::vector < int > solve(int N);
```

- N: размерът на таблицата.

- Функцията трябва да върне точно четири цели числа $\{x_1, y_1, x_2, y_2\}$: горния ляв и долния десен ъгъл на празната квадратна подтаблица, която е намерила. Трябва да е изпълнено условието $x_2 - x_1 = y_2 - y_1 = S - 1$, където $S \geq M$.
- Тази функция се извиква точно веднъж за всеки тестов случай.

Тази функция може да извиква следната функция:

```
int ask(int x1, int y1, int x2, int y2);
```

- x_1, y_1: редът и колоната на горния ляв ъгъл на сканираната подтаблица.
- x_2, y_2: редът и колоната на долния десен ъгъл на сканираната подтаблица.
- Трябва да е изпълнено $1 \leq x_1 \leq x_2 \leq N$ и $1 \leq y_1 \leq y_2 \leq N$. В противен случай програмата Ви се прекъсва незабавно и получава 0 точки за този тестов случай.
- Функцията връща броя на камъните в сканираната подтаблица.

Грейдърът не е адаптивен: позициите на камъните са фиксирани, преди да бъде извикана `solve`, и не зависят от Вашите извиквания на `ask`.

Ограничения

- $2 \leq N \leq 500$
- Можете да направите най-много $2.5 \cdot 10^5$ извиквания на `ask`.

Подзадачи

Подзадача	Точки	Допълнителни ограничения
1	15	$N \leq 40$
2	25	$N \leq 150$
3	60	Няма допълнителни ограничения.

Оценяване

Ако `solve` върне невалиден регион, направи невалидно извикване на `ask` или извика `ask` повече от $2.5 \cdot 10^5$ пъти, получавате 0 точки за този тестов случай.

В противен случай, нека Q е броят на извикванията на `ask`. За този тестов случай получавате точките на подзадачата, умножени по $\min(1, f(Q))$, където

$$f(Q) = \begin{cases} \max(0, -0.05 \cdot Q + 1.55), & \text{ако } Q \leq 15 \\ \max(0, -0.19 \cdot \log_{10}(Q) + 1.02), & \text{ако } Q > 15 \end{cases}$$

Точките Ви за дадена подзадача са минималният брой точки от всички тестове в тази подзадача. Крайният резултат е сумата от точките за подзадачите.

За да ви помогнем да оцените резултата си, таблицата по-долу показва приблизителния множител за някои конкретни стойности на Q:

Брой заявки (Q)	Множител ($\approx$)
$Q \leq 11$	1.00
12	0.95
15	0.80
50	0.70
100	0.64
1000	0.45
10000	0.26
100000	0.07
$Q \geq 233573$	0.00

Локален грейдър

Примерният грейдър чете входа в следния формат:

```
N
c_1 c_2 ... c_N
```

Тук c_i е колоната на камъка на ред i ($1 \leq c_i \leq N$).

Примерният грейдър извежда един ред, съдържащ множителя $\min(1, f(Q))$, получен от Вашите заявки, или 0, ако върнатата подтаблица или някое извикване на `ask` е невалидно.

Примери

Пример 1

Примерен вход на грейдъра:

```
4
1 2 3 4
```

Камъните са на $(1, 1)$, $(2, 2)$, $(3, 3)$ и $(4, 4)$. Вашата програма не знае това.

Грейдърът извиква `solve(4)`. Една възможна поредица от извиквания е:

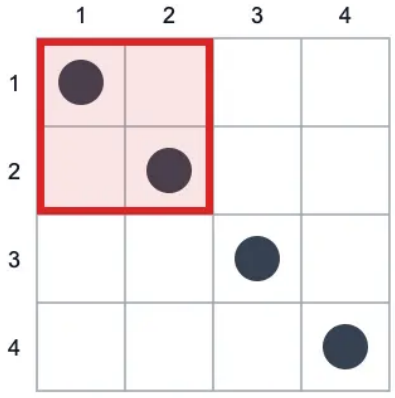
- `ask(1, 1, 2, 2)` връща `2`.
- `ask(2, 4, 4, 4)` връща `1`.
- `ask(3, 1, 3, 2)` връща `0`.

След това `solve` връща `{1, 3, 2, 4}`.

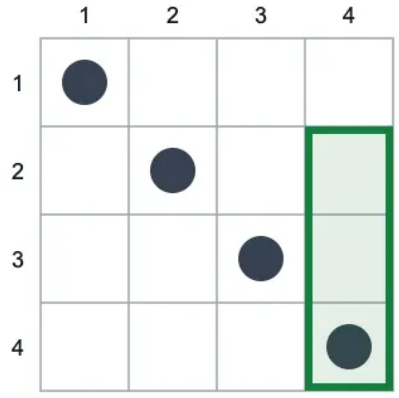
Примерен изход на грейдъра:

```
1
```

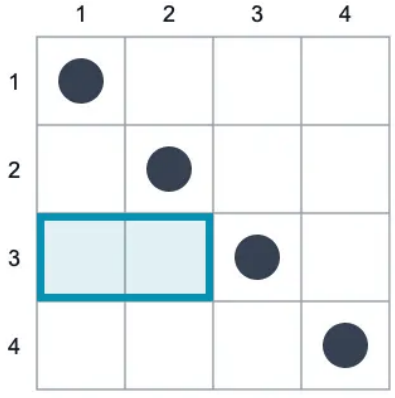
Първото извикване сканира горната лява подтаблица 2×2 и открива камъните в $(1, 1)$ и $(2, 2)$. Второто сканира редове от 2 до 4 в колона 4 и открива камъка в $(4, 4)$. Третото сканира първите две клетки на ред 3 и не открива камъни. Върнатата област обхваща редове от 1 до 2 и колони от 3 до 4, което представлява празна подтаблица с размер 2×2.



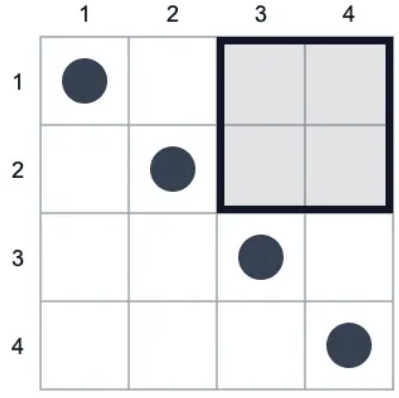
ask(1, 1, 2, 2) returns 2



ask(2, 4, 4, 4) returns 1



ask(3, 1, 3, 2) returns 0



return {1, 3, 2, 4}

За $N = 4$ програмата е използвала $Q = 3$ извиквания, така че $f(Q) = 1.4$ и множителят е 1. Приемат се и други отговори, например `{3, 1, 4, 2}`, `{1, 2, 1, 2}`, `{3, 2, 3, 2}`, `{3, 4, 3, 4}` или която и да е празна клетка, тъй като $M = 1$ за $N = 4$.