Die große Scheune

Der Jungbauer Markus[1] möchte eine möglichst große Scheune auf seiner quadratischen Farm errichten. Als Naturliebhaber hasst er es, unnötig Bäume auf seinem Land zu fällen. Daher möchte er einen Platz für seine Scheune finden, der bereits frei von Bäumen ist.

Für unseren Zweck ist sein Gebiet in $n \times n$ gleich große quadratische Grundstücksparzellen eingeteilt, und auf k davon stehen Bäume. Du sollst die Seitenlänge der größtmögliche quadratische Scheune ermitteln, die auf Markus' Farm errichtet werden kann, ohne dass Parzellen mit Bäumen verwendet werden müssen. Die Wände der Scheune müssen parallel zu den Parzellengrenzen sein.

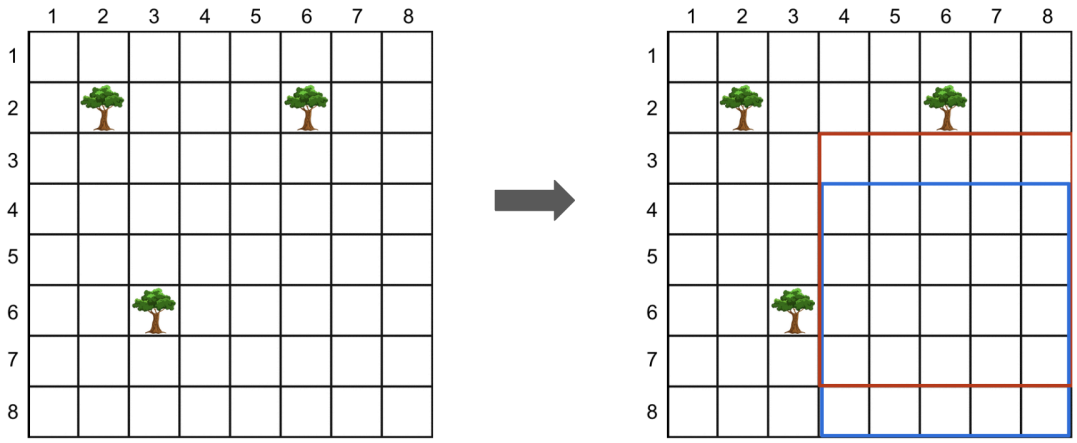

Abbildung 1: Diese Abbildung entspricht dem Beispielinput. Die größte Scheune hat Seitenlänge 5 und ihre links-obere Ecke kann entweder an Position (3, 4) oder (4, 4) sein.

Implementierungsdetails

Implementiere die Funktion `solve(int n, Point[] trees)`, wobei `n` die Seitenlänge der Grundstücksparzelle ist, und `trees` die Position der Bäume beschreibt (jeweils in `r` für die Reihe, und `c` für die Spalte). Gib in dieser Funktion die größtmögliche Seitenlänge einer Scheune zurück.

[1] AOI Trainer vor langer Zeit

Beispiel

Eingabe	Ausgabe	Anmerkungen
8 3 2 2 2 6 6 3	5	Es handelt sich um ein 8×8 Grundstück mit insgesamt 3 Baumparzellen, welche die Koordinaten $(2, 2)$, $(2, 6)$ und $(6, 3)$ haben. Dieser Input ist in Abbildung 1 illustriert.

Subtasks

Allgemein gilt:

- $1 \leq n \leq 10^9$
- $1 \leq k \leq 10^5$
- $1 \leq r_i, c_i \leq n$
- Auf keiner Grundstücksparzelle befindet sich mehr als ein Baum.

Subtask 1 (5 Punkte): $n \leq 25$

Subtask 2 (10 Punkte): $n \leq 100$

Subtask 3 (15 Punkte): $n \leq 500$

Subtask 4 (20 Punkte): $n \leq 3000$

Subtask 5 (15 Punkte): $k \leq 100$

Subtask 6 (15 Punkte): $k \leq 3000$

Subtask 7 (20 Punkte): Keine Einschränkungen

Limits

Zeitlimit: 1 s **Speicherlimit:** 256 MB

Eingabe

Das Einlesen der Eingabe wird von einem von uns geschriebenen Programm, dem Grader, übernommen. In der ZIP-Datei zu dieser Aufgabe findest du für jede Programmiersprache eine grader-Datei, das du zum Testen deiner Lösung mit deinem Programm kompilieren kannst.

Dieser Grader liest die Eingabe im folgenden Format ein: Die erste Zeile enthält zwei Zahlen n und k, die Seitenlänge und die Anzahl Bäume. Die folgenden k Zeilen enthalten jeweils zwei Integer r_i, c_i, welche die Zeile und Spalte des i-ten Baums beschreiben. Der Grader gibt das Ergebnis der `solve`-Funktion auf der Standardausgabe aus.