Problem Bugatti. Разрязване на торта

Input file:	`input.txt` or standard input
Output file:	`output.txt` or standard output
Time limit:	2 seconds
Memory limit:	512 megabytes

Тази задача може да се реши само на език за програмиране C++.

Организаторите на Закретата ученическа олимпиада по програмиране решили да поръчат огромна торта за тържествената вечеря. За целта, те поръчали торта с n свещички, номерирани от 0 до $n-1$. Известно е, че свещичките се намират в различни точки от равнината, никои тре свещички не лежат на една права, Никои четири свещички не образуват трапец(четириъгълник с две успоредни страни).

За да има парче торта за всеки, организаторите искат предварително да опишат начин за разрязване на тортата на максимален брой парчета. За съжаление, вие не знаете разположението на свещичките върху тортата, а единствения достъпен начин да се разбере, как изглежда тортата е кореспонденция със сладкърницата по електронната поща.

В едно съобщение можете да поискате да се претеглят не повече от четири триъгълни парчета, с върхове в свещички:

- Предоставяте два списъка с триъгълници `left` и `right`, общо съдържащи не повече от четири триъгълника. Триъгълниците може да се пресичат, да имат общи свещи и даже да съвпадат.
- В сладкарницата ще изрежаттриъгълни парчета, съответстващи на изпратения списък от няколко копия на тортата и ще сложат парченцата от `left` на лявата чаша на везната, а парченцата от `right` на дясната чаша на везната.
- След това ви изпращат резултата от претеглянето. Приема се, че теглото на парчето е равно на неговата площ. В резултат на това вие ще разберете, в кой от двата списъка от триъгълници общата площ е по-голяма, или че площите съвпадат.

След няколко претегляния, вие трябва да можете да намерите начин за разрязване на тортата. Всяко парче трябва да бъде изпъкнал многоъгълник с върхове в свещички. Върховете на дадено парче трябва да бъдат изброени по посока на часовниковата стрелка или обратно на часовниковата стрелка. Никои две парчета не трябва да се пресичат във вътрешна точка. Трябва да върнете списък с парчета, чиято обща площ е възможно най-голяма. В някои групи може също да искате да увеличите максимално броя на парчетата.

Solution Format

Това е нестандартна задача. Формата на тестването и е с грейдър и затова вие трябва да реализирате само функция `solve` с решението. Тази функция ще бъде извиквана от тестващата програма (грейдър) на журито и стойността, която ще върне функцията се ще приема като решение на задачата.

В частност, това означава, че в изпратеният от вас код**не трябва да има вход и изход**. Във вашия код **не трябва** да има функция `main`. При необходимост може да реализирате произволен брой помощни функции, структури, класове и глобални променливи, но целият код на вашето решение трябва да бъде в един файл.

Вие трябва да реализирате следната функция:

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

Функцията `solve` приема като параметър единствено цяло число n — броя на свещичките.

При реализацията на функцията `solve` може да използвате функцията `compare`, която реализира грейдъра:

```
int compare(const std::vector<Triangle>& left, const std::vector<Triangle>& right);
```

Тази функция приема като параметри два непразни списъка от триъгълници, с общ размер не повече от 4. Като резултат тя връща -1, ако общата площ на триъгълниците от `left` е по-малка от общата площ на триъгълниците от `right`, 0, ако площите са равни и 1, в третия вариант. Ако е направена некоректна заявка или бъде достигнат максималния брой заявки, програмата автоматично завършва своята работа. За да получите достъп до функцията `compare` в решението, на първия ред на вашия код трябва да включите заглавния файл на грейдъра:

```
#include "triangles.h"
```

В този файл е декларирана и използваната във функцията `compare` структура `Triangle`:

```
struct Triangle {
    int i, j, k;

    Triangle() = default;
    Triangle(int i_, int j_, int k_) : i(i_), j(j_), k(k_) {}
};
```

Дадената структура описва едно триъгълно парче, като член-данните `i`, `j` и `k` описват номерата на свещичките, образуващи върховете на триъгълното парче. Номерата на свещичките трябва да са различни за едно триъгълно парче, но могът да се повтарят в няколко триъгълника.

В кода на изпращаното решение не трябва да включвате декларация на структурата `Triangle`, защото тя ще се вземе автоматично от заглавния файл.

Всички параметри (индексите на свещичките в заявките, а също и връщаният от вас резултат) се задава в **0-индексация**.

Вашата функция `solve` с помощта на извикването на функцията `compare` трябва да намери произволно разделяне на тортата на изпъкнали парчета, в които кои да е две парчета не се пресичат във вътрешна точка, Общата площ на парчетата е максимална, и, возможно е, броят на парчетата е максимален.

Функцията `solve` трябва да върне списък от парчетата, на които се разделя тортата. Всяко парче се описва със стуктурата `std::vector<int>`, който се състои от свещички, образуващи изпъкнал многоъгълник. Свещичките трябва да са изброени **подред по границите на многоъгълника** (по часовниковата стрелка или обратно на часовниковата стрелка). Кои зда е две парчета не трябва да се пресичат във вътрешна точка. Трябва да върнете като резултат списък с парчетата, общата площ,на които е максимално възможна, а в някои групи допълнително се изисква да се максимизира броя парчета, при това съхранявайки условието за максимална възможна обща площ.

При едно стартиране на грейдъра **журито прави точно едно обръщение към функцията `solve`**. **Грейдърът не е адаптивен**, т.е. положенията на свещичките се фиксират предварително и не зависят от реализацията на функцията `solve`.

Testing

Предоставен ви е шаблон на решението `triangles.cpp`, а също и заглавния файл `triangles.h`, съдържащ декларации на функциите `solve` и `compare`. За удобство при тестването ви е предоставен грейдър във файла `grader.cpp`.В този файл е реализирано въвеждане на входни данни от стандартния вход, извикване на функцията `solve` и извеждане на стандартния изход на върнатата от функцията `solve` стойност.В тестващата система грейдърът може да е различен.

За да компилирате вашия код `triangles.cpp` на езика C++, използвайте командата

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

След изпълнение на тази команда ще бъде създаден изпълним файл на грейдъра `grader` или `grader.exe`, в зависимост от вашата операционна система, след стартирането на който може да

се въведе тест във формата, указан по-долу.

Ако компилирането чрез команда ви е трудно, за локално тестване може да копирате функцията `solve` в файла `grader.cpp` (трябва да я поставите преди функцията `main`) и да стартирате файла `grader.cpp`. При това, преди да изпратите вашето решение към системата за тестване, в него трябва да остане само реализацията на функцията `solve`, **като не забравяте да включите заглавния файл в началото на кода** (това става с помощта на реда `#include "triangles.h"`).Ако използвате някакви библиотеки на езика C++, тяхното включване също трябва да бъде добавено в началото на изпращания код.

В случай, че получите съобщение за грешка при компилация, убедете се, че **в изпратеният от вас код не се съдържа функция `main`, нито декларации на функцията `compare` или структурата `Triangle`**. Функцията `compare` и структурата `Triangle` може просто да използвате в изпратения от вас код.

Input

Грейдърът чете тест в следния формат:

На първия ред се намира цяло число n ($4 \le n \le 10\,000$) — броя на свещичките.

На следващите n реда са зададени по две цели числа x_i, y_i ($-10^9 \le x_i, y_i \le 10^9$) — координатите на i-тата свещичка.

Output

Грейдърът извежда резултатът, върнат от функцията `solve` — список от намерените парчета.

На първия ред се извежда броя на намерените многоъгълници (размера на вектора, който връща функцията `solve`).

На следващите редове за всеки многоъгълник (елемент на вектор от вектора, който връща функцията `solve`) в началото се извежда размера на многоъгълника, а на следващия ред номарата на свещичките, които са върхове на този многоъгълник (елементите на всеки от `std::vector<int>`, включен във върнатата от функцията `solve` стойност). Всеки многоъгълник трябва да бъде изпъкнал, неговите върхове трябва да са изброени последователно (по часовниковата стрелка или срещу часовниковата стрелка). Никои два многоъгълника не трабва да се пресичат във вътрешна точка.

Във файла `grader.cpp` има променлива `verbose`, която в началото е равна на 0. При увеличение на нейната стойност грейдърът ще извежда по-подробна информация за вашето решение и неговите заявки.

Examples

input	output
4 1 2 1 4 0 0 3 -1	3 3 0 1 2 3 0 2 3 3 0 1 3
5 -1 -1 4 4 4 -2 1 2 -2 2	1 4 0 4 1 2
6 2 2 0 -2 -1 3 -2 0 7 0 2 -3	4 3 2 4 3 3 3 4 5 3 1 3 5 3 0 2 4

Note

В първия пример възможна последователност на обръщения към функциите може да изглежда така:

Отначало се извиква функцията `solve(4)`.

От функцията `solve` се извиква функцията

`compare({Triangle(0, 1, 2)}, {Triangle(1, 2, 3)})}`

По време на изпълнението на функцията се прави сравнение на размерите на триъгълните парчета, образувани от триъгълник с върхове в свещичките с номера 0, 1 и 2 и триъгълник с върхове в свещичките с номера 1, 2, 3.

Тъй като първият триъгълник е по-малък от втория, функцията връща -1.

След това във функцията `solve` се извиква функцията

`compare({Triangle(1, 2, 3)}, {Triangle(0, 1, 2), Triangle(0, 1, 3)})}`

В отговор ще бъде върнато 1, тъй като площта на триъгълника с върхове в свещичките 1, 2 и 3 е по-голяма от общата площ на триъгълника с върхове в свещичките 0, 1 и 2 и на триъгълника с върхове в свещичките 0, 1 и 3.

След това във функцията `solve` се извиква функцията

`compare({Triangle(1, 2, 3)}, {Triangle(0, 1, 2), Triangle(2, 3, 0), Triangle(0, 1, 3)})}`

В отговор ще бъде върнато 0.

Накрая функцията `solve` връща`{{0, 1, 2}, {0, 2, 3}, {0, 1, 3}}` — описанието на разделянето на тортата на изпъкнали парчета, имащи максимално възможна обща площ и не пресичащи се във вътрешни точки.

В първия и третия пример, в намерения начин да се разреже тортата броят на парчетата е максималният възможен, затова тези отговори ще се приемат за коректни в групите с максимизация. Във втория тест броят на парчетата не е максималния възможен, затова този отговор ще се приеме за коректен в групите без максимизация, но ще получи 0 точки в групите с максимизация.

Обърнете внимание, че първият тест попада под допълнителните ограничения на групите 1 и 2, а третия тест попада под допълнителните ограничения на групите 5.

Scoring

Тестовете към тази задача се състоят от 11 групи. Правилата, по които се присъждат точките за групите са описани по-долу.Обърнете внимание, преминаването на тестовете от условието не е необходимо за някои от групите. Общия брой точки за всяка група е равен на максималния брой точки, получени за тази група тестове от всички събмити. Точките за решението за група тестове са равни на минималния брой точки, получени на всички тестове в групата.

- Ако вашето решение направи некоректно извикване на `compare` или върне отговор неудовлетворяващ описаните условия, то ще получи 0 точки за теста.
- Във всеки тест на вашето решение ще бъдат позволени не повече от $2 \cdot 10^6$ извиквания на функцията `compare`. В случай на превишаване на лимита, вашето решение ще получи 0 точки за този тест.
- В част от групите се изисква да се максимизира броя на парченцата в отговора. В тези групи, ако броят на парчетата не е максималния възможен, вашето решение ще получи 0 точки за този тест.

Ако нито едно от описаните условия в теста не е нарушено, отговорът на теста се приема за правилен. Нека пълният брой точки за група тестове е p. Част от групите се оценяват с използване на формули, а останалите се оценяват без използване на формули:

- Ако групата се оценява без използване на формули, точките за теста са p.
- Ако групата се оценява с използване на формули, нека q е броят на извиквания на функцията `compare`, което е направило вашето решение. Тогава точките за теста са $\left\lfloor p \cdot \frac{20n}{\max(q, 20n)} \right\rfloor$.

Група	Пълен брой точки	Оценка		Доп. ограничения	Необх. групи	Коментари
		Макс.	Формула	n		
0	0	не	не	–	–	Тестовете от условието.
1	13	не	не	$n = 4$	–	
2	4	да	не	$n = 4$	1	
3	13	не	не	–	–	$(10^9, -10^9)$, $(-10^9, 10^9)$, $(10^9, 10^9)$ има в множеството, останалите точки са случайни вътре в този триъгълник
4	8	да	не	–	3	$(10^9, -10^9)$, $(-10^9, 10^9)$, $(10^9, 10^9)$ има в множеството, останалите точки са случайни вътре в този триъгълник
5	11	да	да	–	–	точките са върхове на изпъкнал многоъгълник
6	9	не	не	$n \leq 100$	–	случайни точки
7	8	да	не	$n \leq 100$	6	случайни точки
8	9	не	да	–	–	случайни точки
9	8	да	да	–	–	случайни точки
10	9	не	да	–	–	
11	8	да	да	–	–	

В групите 6 – 9 всички точки са избрани случайно и независимо от квадрата $[-10^9, 10^9] \times [-10^9, 10^9]$.

В групите 3 – 4 точките, освен $(10^9, -10^9)$, $(-10^9, 10^9)$, $(10^9, 10^9)$, са избрани случайно и независимо от триъгълника с върхове $(10^9, -10^9)$, $(-10^9, 10^9)$, $(10^9, 10^9)$.

В тези групи от всички случайни тестове са оставени само тези, в които всички точки са различни, никои три не лежат на една права и никои четири не образуват трапец.

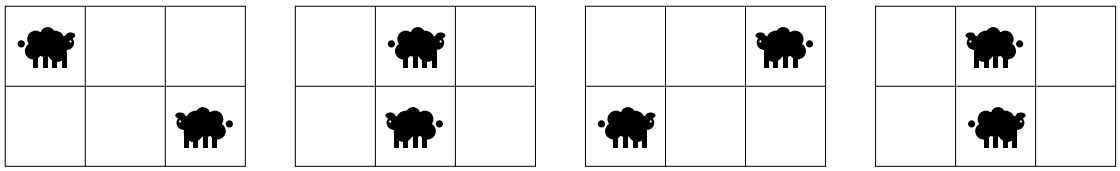
Problem Ferrari. Марш на овцете

Input file:	`input.txt` or standard input
Output file:	`output.txt` or standard output
Time limit:	1 second
Memory limit:	512 megabytes

Сашко се уморил от тежкия градски живот и решил да се пресели на село. За да се занимава с нещо, той решил да развъжда овце. За целта той си купил в селото правоъгълно парче земя, което може да се представи като мрежа от клетки с размери $n \times m$, в което редовете са номерирани отгоре надолу с числата от 1 до n, а колоните са номерирани отляво надясно с числата от 1 до m.

За своя участък Сашко си купил n овце, за всяка от които отделил цял ред. В началото i-тата овца била поставена в клетка с координати (i, a_i) (в ред с номер i и колона с номер a_i). Сашко установил, че овцете се преместват само хоризонтално по следните правила:

- Ако участъкът има само една колона, овцете не се преместват.
- Отначало i-тата овца се намира в клетка (i, a_i), а също така всяка овца има своя посока, на къде ще се движи — или наляво, или надясно. При това не съществува овца, която се намира в първата колона и върви наляво, или такава, която се намира в последната колона и върви надясно.
- Всяка секунда всяка овца се премества една колона наляво, ако посоката и е наляво или една колона надясно, ако посоката и е надясно.
- Ако след преместване овцата се окаже в колона 1 и се движи наляво, или ако овцата се окаже в последната колона и се движи надясно, тя порменя посоката си на движение на противоположната. По този начин овцете никога не излизат извън участъка.



Тук е нарисуван пример за движение на овцете, ако в началото първата овца се намира в колона 1 и се движи надясно, а втората се намира в колона 3 и се движи наляво. На първата картинка е нарисувано състоянието в началото, на втората - след една секунда, на третата - след 2 секунди след началото на движението, на четвъртата - след 3 секунди след началото на движението.

На Сашко не му харесва, че овцете се движат така хаотично и затова той иска да направи така, че всички овце да се движат еднакво. Това означава, че всички овце трябва да са в една и съща колона и да имат една и съща посока на движение. За да постигне това, Сашко може да ореже своя участък няколко (възможно, нула) пъти по следния начин:

- Той избира момент t (колко секунди са минали от началото на движението на овцете) и броя колони x, които ще останат в участъка след орязването.
- Всички овце в момент t трябва да се намират строго в участъка с размер $n \times x$. Това означава, че за произволно i от 1 до n, i-тата овца в момента t трябва да се намира в клетка (i, y_i), където $1 \le y_i \le x$. В противен случай такова орязване на участка е недопустимо.
- След тази операция, от момента t нататък, броят на колоните в участъка ще се намали на x.

- Всяка овца, която след тази операция се намира в последната колона и се движи надясно, ще промени посоката си на движение на противоположната.

Подробни примери за процеса на оразване на участъка са дадени в описанието към тестовете от условието.

На първо време Сашко иска да разбере, дали след няколко орязвания на участъка може да се постигне всички овце да се движат еднакво. Ако това е възможно, Сашко иска да знае, какъв е максималния брой колони, които могат да останат в участъка и как да стане това. Помогнете на Сашко да реши тази задача!

Input

На първия ред на стандартния вход е зададено едно число T ($1 \le T \le 3$) — параметър, който означава каква точно информация за отговора трябва да се изведе. Подробно описание е дадено във "Output".

На втория ред са записани две цели числа n и m ($2 \le n \le 200\,000$, $2 \le m \le 10^9$) — броя редове и броя колони на участъка.

На третия ред са дандени n числа $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le m$) — номерата на колоните, където в началото се намират овцете.

На четвъртия ред е записан низ s с дължина n, който се състои само от символи `L`, `R`, където i-тия символ е равен на `L`, ако i-тата овца в началото се движи наляво, и `R`, ако i-тата овца в началото се движи надясно. Гарантирано е, че никоя овца в първа колона не се движи наляво, и никоя овца в m-тата колона не се движи надясно.

Output

На първия ред изведете «`No`» (без кавичките), ако Сашко не може да постигне това овцете да се движат в една посока. В противен случай изведете «`Yes`» (без кавичките).

Ако $T = 2$ или $T = 3$, на втория ред изведете максималния брой колони, които Сашко може да остави на участъка си така, че всички овце да се движат еднакво. В случай, че $T = 1$, **това не е необходимо да се извежда**.

Ако $T = 3$, на третия ред изведете число q ($0 \le q \le 10^6$) — колко пъти Сашко ще трябва да орязва участъка си. На следващите q реда изведете описание на оразването на участъка.

За описание на всяко орязване на участъка изведете на един ред две цели числа t и x ($0 \le t \le 10^{18}$, $1 \le x < m$), където t е момента, в който Сашко трябва да ореже участъка си (от началото на движението на овцете), а x — броя на колоните от участъка, които ще останат в участъка след операцията.

Операциите трябва да се извеждат в ненамаляващ ред на t. Във всеки случай на орязване x трябва да е по-малък от този в предходното орязване.

Ако има няколко подходящи последователности на орязване, може да се изведе коя да е от тях.

Може да се докаже, че при дадени ограничения, ако съществува отговор, тогава съществува и отговор, който се вписва в тези ограничения.

В случай, че $T = 1$ или $T = 2$, **не е нужно да се извежда описанието на орязване на участъка**.

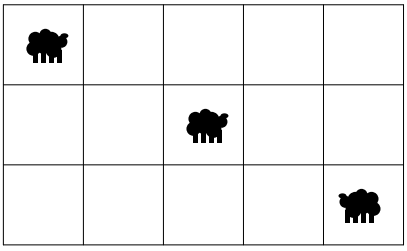
Examples

input	output
3 2 3 1 3 RL	Yes 2 1 3 2
3 2 3 1 2 RL	No
3 3 5 1 3 5 RRL	Yes 3 2 1 4 2 3
2 3 5 1 3 5 RRL	Yes 3
1 3 5 1 3 5 RRL	Yes
3 3 7 3 3 5 RRL	Yes 4 3 0 6 0 5 1 4

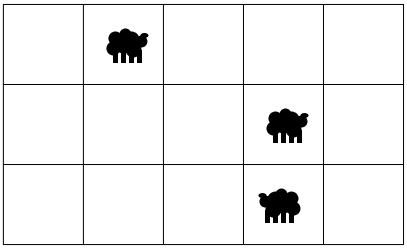
Note

Да разгледаме третия пример:

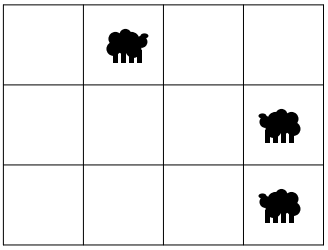
Състояние на овцете в нулевата секунда.



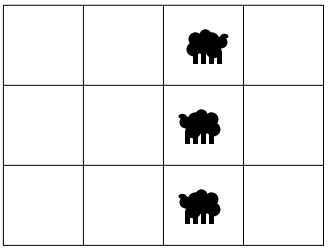
Състояние на овцете в първата секунда.



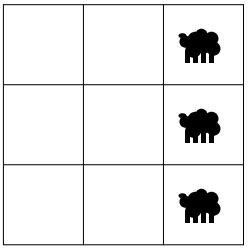
Орязва се участъка до 4 клетки.



Състояние на овцете в втората секунда.



Орязва се участъка до 3 клетки.



Scoring

Тестовете към тази задача се състоят от десет групи. Точките за всяка група се дават само ако са преминати всички тестове от групата и всички тестове от някои от предходните групи. Обърнете внимание, преминаването на тестовете от условието не е необходимо за някои от групите. **Offline-проверка** означава, че резултатите от тестването на вашите решения за дадена група ще бъдат достъпни след края на състезанието. Общия брой точки за всяка група е равен на максималния брой точки, получени за тази група тестове от всички събмити.

Група	Точки	Доп. ограничения			Необх. групи	Коментари
		T	n	m		
0	0	–	–	–	–	Тестовете от условието.
1	8	$T \leq 1$	–	–	–	
2	11	$T \leq 2$	$n \leq 3$	$m \leq 4$	–	
3	9	–	$n = 2$	–	–	$a_1 = 1, a_2 = m$
4	12	–	$n = 2$	$m \leq 200\,000$	–	
5	8	–	$n = 2$	–	3, 4	
6	14	$T \leq 2$	$n \leq 1000$	$m \leq 1000$	2	
7	10	$T \leq 2$	–	–	1, 2, 6	
8	9	–	–		–	Низа s съдържа само символи `R`.
9	12	–	$n \leq 1000$	$m \leq 1000$	0, 2, 6	
10	7	–	–	–	0 – 9	**Offline-проверка.**

Problem Lamborghini. Украсяване

Input file:	`input.txt` or standard input
Output file:	`output.txt` or standard output
Time limit:	1 second
Memory limit:	512 megabytes

В процеса на подготовка организаторите на Закритата ученическа олимпиада по програмиране решили да украсят аудиторията. За целта, на стената в една права са забити $2n$ колчета. Всички колчета се намират на различни позиции. Между тях са опънати n нишки, като i-тата от тях свързва колчетата с разстояние l_i и r_i от началото на стената ($l_i < r_i$). Известно е, че със всяко колче е свързана точно една нишка.

С нишките може да се изпълняват операции *пренатягане*. За една операция *пренатягане* може да се изберат две нишки (l_i, r_i) и (l_j, r_j), да се разкачат от колчетата, а след това да се прекарат две нови нишки, като се използва всяко от четирите освободени колчета точно веднъж. Т.е. от четирите освободени колчета се образуват две нови двойки, между които се прекарват две нови нишки.

Нишките, свързващи колчетата (l_i, r_i) и (l_j, r_j) се пресичат, ако отсечките $[l_i, r_i]$ и $[l_j, r_j]$ имат поне една обща точка. Приема се, че конфигурация от нишки има красота поне k, ако съществува множество от k нишки, такова че всеки две нишки в множеството се пресичат. Обърнете внимание, че ако дадена конфигурация има красота k, то тя има красота поне $k-1,\ k-2,\ \ldots,\ 0$.

Организаторите на олимпиадата имат q заявки за получаване на конфигурации с някаква красота в резултат от прилагане на няколко операции *пренатягане*. В i-тата заявка те биха искали да получат конфигурация с красота поне k_i. За всяка от заявките, организаторите искат да разберат, какъв минимален брой операции *пренатягане* трябва да се изпълнят, за да получат желания вид конфигурация. Заявките са независими една от друга. Т.е. операциите *пренатягане* не се съхраняват между заявките.

Input

От първия ред на стандартния вход се въвеждат две цели числа n и q ($1 \le q \le n \le 200\,000$) — броя нишки и броя заявки.

На следващите n реда се описват нишките. В i-тия от тях се задават две цели числа l_i и r_i ($1 \le l_i < r_i \le 10^9$) — номерата на колчетата, свързани с i-тата нишка. Гарантирано е, че всяко колче се среща точно веднъж.

На следващия ред са зададени q цели числа $k_1,\ k_2,\ \ldots,\ k_q$ ($1 \le k_i \le n$) — размерите на желаните красоти в заявките на организаторите. Гарантирано е, че всички k_i са различни.

Output

За всяка заявка изведете едно цяло, неотрицателно число — минималния брой операции *пренатягане*, необходими за получаване на конфигурация от нишки, която има зададената в заявката красота.

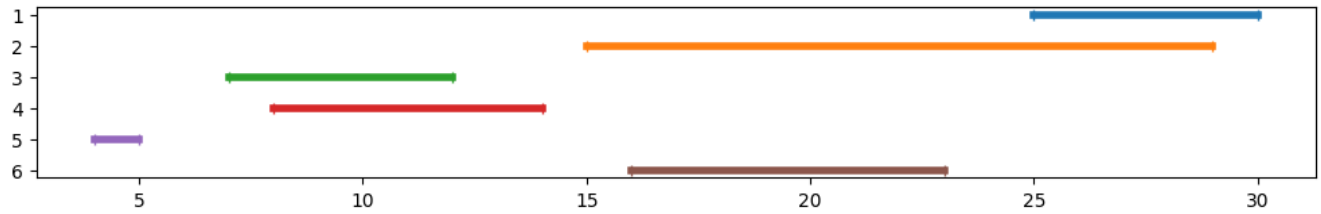
Гарантирано е, че за всяка заявка след няколко пренатягания може да се получи търсената красота.

Example

input	output
6 6 25 30 15 29 7 12 8 14 4 5 16 23 1 2 3 4 5 6	0 0 1 1 2 3

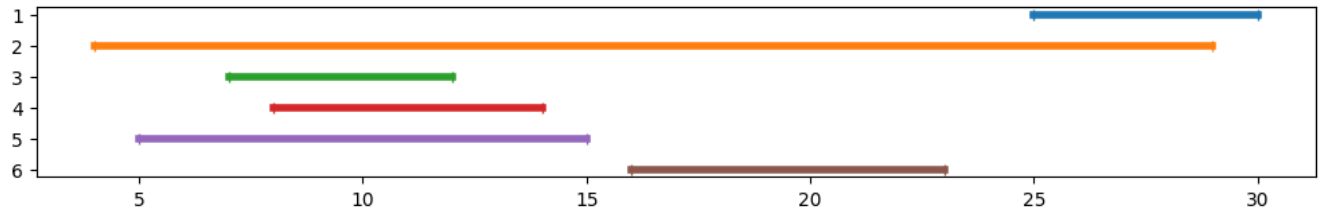
Note

В първия пример нишките първоначално имат следната конфигурация:

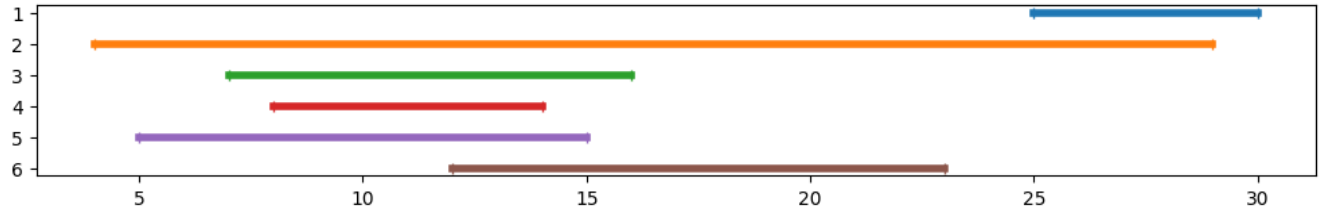


Тъй като нишките 3 и 4 вече се пресичат, то за получаване на красота поне 1 и красота поне 2 не е нужно да се изпълняват никакви операции.

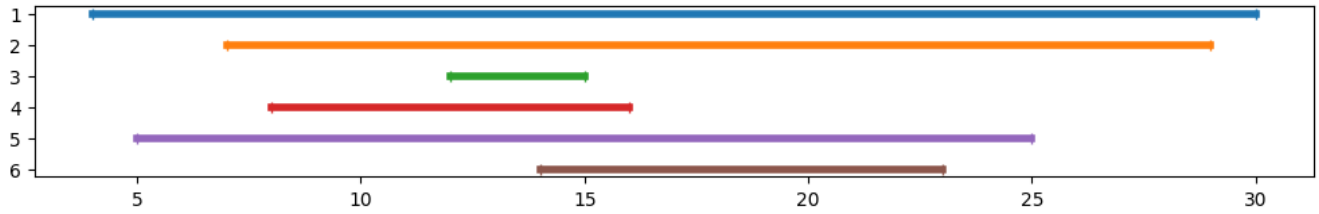
За получаване на красота 3 и 4 може да се изпълни операция *пренатягане* към нишките 2 и 5 и да се получат нишки $(4, 29)$ и $(5, 15)$



За получаване на красота 5 може допълнително да се приложи операция към нишките 3, 6 и да се получи $(7, 16)$ и $(12, 23)$.



За да се пресекат всичките 6 нишки, може да се приложат операции към нишките 1 и 5, 2 и 3, 4 и 6 и да се получи следната конфигурация:



Scoring

Тестовете към тази задача се състоят от шест групи. Точките за всяка група се дават само ако са преминати всички тестове от групата и всички тестове от някои от предходните групи. Обърнете внимание, преминаването на тестовете от условието не е необходимо за някои от групите.

Група	Точки	Доп. ограничения			Необх. групи	Коментари
		n	q	k_i		
0	0	–	–	–	–	Тестовете от условието.
1	14	$n \le 100$	–	–	0	–
2	16	$n \le 3000$	–	–	0, 1	–
3	13	–	–	–	–	Нишките две по две не се пресичат
4	25	–	$q = 1$	$k_i = n$	–	–
5	17	–	$q = 1$	$k_i \le 10$	–	–
6	15	–	–	–	0 – 5	–

Problem McLaren. Чудовища и мечове

Input file:	`input.txt` or standard input
Output file:	`output.txt` or standard output
Time limit:	1 second
Memory limit:	512 megabytes

Дадени са n чудовища, подредени в редица. За всяко чудовище са известни две величини: h_i — здравето на чудовището и r_i — наградата за победа над даденото чудовище, изразена в монети. Един рицар трябва да победи всички чудовища.

За битките с чудовищата рицарят разполага с m вида мечове. За всеки вид меч са известни следните характеристики: s_j — сила на меча и c_j — цена на меча в монети. За да купи меч с цена c_j рицарят трябва да има поне c_j монети. След купуването на меча, броят на монетите на рицаря се намалява с c_j. Отначало рицарят има x монети.

След покупката на един меч, той може да се използва не повече от k пъти при битките с чудовищата. Всеки вид меч може да бъде закупен неограничен брой пъти. Меч със сила s_j може да убие чудовище със здраве h_i, ако $s_j \geq h_i$. Във всеки момент от време, рицарят може да има само един меч, т.е. след купуване на нов меч старият меч вече не може да бъде използван (но в бъдеще може отново да се купи меч от този тип).

Чудовищата трябва да се побеждават във фиксиран ред: от първото до последното. Необходимо е да се изясни, може ли рицарят да направи това.

Input

Първия ред на стандартния вход съдържа четири цели числа n, m, k и x ($1 \leq n, m \leq 500\,000$, $1 \leq k \leq n$, $1 \leq x \leq 10^9$) — броя чудовища, броя на видовете мечове, максималния брой пъти, които може да се използва един меч и началния брой монети у рицаря.

На следващите n реда са дадени описанията на чудовищата. На i-тия ред са дадени две цели числа: h_i и r_i ($1 \leq h_i, r_i \leq 10^9$) — здравето на i-тото чудовище и наградата за победа над него.

На следващите m реда е дадено описанието на характеристиките на мечовете. На j-тия ред са дадени две цели числа s_j и c_j ($1 \leq s_j, c_j \leq 10^9$) — силата и цената на j-тия меч.

Output

Изведете «`Yes`» (без кавичките),ако рицарят може да победи всички чудовища, и «`No`» (без кавичките) в противен случай.

Examples

input	output
3 3 2 5 1 1 5 5 9 5 9 10 1 1 5 1	Yes
5 5 5 6 3 6 2 4 1 1 4 4 8 4 2 7 7 7 3 4 5 9 6 4	No

Note

В първия промер, рицарят отначало може да купи третия меч, с негова помощ може да победи първото и второто чудовище. След това той ще има $5 - 1 + 1 + 5 = 10$ монет. Благодарение на това, той може да купи първия меч и с него да победи последното чудовище.

Във втория пример рицарят не може да победи всички чудовища, защото няма вид меч, с помощта, на който може да победи петото чудовище.

Scoring

Тестовете към тази задача се състоят от 9 групи. Точките за всяка група се дават само ако са преминати всички тестове от групата и всички тестове от някои от предходните групи. Обърнете внимание, преминаването на тестовете от условието не е необходимо за някои от групите. **Offline-проверка** означава, че резултатите от тестването на вашите решения за дадена група ще бъдат достъпни след края на състезанието. Общия брой точки за всяка група е равен на максималния брой точки, получени за тази група тестове от всички събмити.

Група	Точки	Доп. ограничения			Необх. групи	Коментари
		n	m	k		
0	0	–	–	–	–	Тестовете от условието.
1	11	–	–	$k = 1$	–	–
2	9	$n \leq 100$	$m \leq 100$	$k = n$	–	–
3	14	$n \leq 100\,000$	$m \leq 3000$	$k = n$	2	–
4	16	–	–	$k = n$	2, 3	–
5	7	$n \leq 400$	$m \leq 400$	–	0, 2	–
6	8	$n \leq 3000$	$m \leq 3000$	–	0, 2, 5	–
7	10	$n \leq 150\,000$	$m \leq 150\,000$	–	0, 2, 3, 5, 6	–
8	12	$n \leq 300\,000$	$m \leq 300\,000$	–	0, 2, 3, 5 – 7	–
9	13	–	–	–	0 – 8	**Offline-проверка.**