Poprawne dwunawiasowanie **(dwunawiasowanie)**

Limit pamięci: 32 MB Limit czasu: 1.00 s

Poprawnym dwunawiasowaniem nazwiemy każdy ciąg złożony ze znaków `([)]`, który można otrzymać z rekurencyjnej zależności:

- Ciąg pusty jest poprawnym dwunawiasowaniem
- Jeśli `P` jest poprawnym dwunawiasowaniem, to `[P]`, oraz `(P)` są poprawnymi dwunawiasowaniami
- Jeśli `P` i `Q` są poprawnymi dwunawiasowaniami, to `PQ` jest poprawnym dwunawiasowaniem

Przykładowo, `[()[]]` jest poprawnym dwunawiasowaniem, ale `)(]`, lub `[(]` nie są.

Napisz program, który wczyta ciąg nawiasów oraz zapytania o poprawność dwunawiasowania wybranych spójnych podsłów ciągu, wyznaczy dla każdego zapytania czy dwunawiasowanie jest poprawne i wypisze wyniki na standardowe wyjście.

Wejście

W pierwszym wierszu wejścia znajdują się jedna liczba N, oznaczająca długość ciągu.
W drugim wierszu wejścia znajduje się ciąg nawiasów o długości N.
W trzecim wierszu wejścia znajdują się jedna liczba Q, oznaczająca ilość zapytań.
W kolejnych Q wierszach znajduje się opis kolejnych zapytań, po jednym w wierszu. Opis każdego zapytania składa się z dwóch liczb naturalnych L_i, R_i, oddzielonych pojedynczym odstępem. Określają one zapytanie o poprawność zadanego spójnego podciągu dwunawiasowania od L_i-tego znaku do R_i-tego znaku włącznie.

Wyjście

Twój program powinien wypisać na wyjście dokładnie Q wierszy. W i-tym wierszu powinna się znaleźć odpowiedź dla i-tego zapytania. Odpowiedź dla każdego zapytania to jedno słowo `TAK`, jeśli dwunawiasowanie jest poprawne lub `NIE` w przeciwnym przypadku.

Ograniczenia

$1 \leq N, Q \leq 1\ 000\ 000$, $1 \leq L_i \leq R_i \leq N$.
W testach wartych 30% wszystkich punktów zachodzi $1 \leq N, Q \leq 2\ 000$.
W testach wartych 20% wszystkich punktów ciąg składa się tylko ze znaków `()`.

Przykład

Wejście	**Wyjście**
5 [()][5 1 4 2 3 1 2 1 3 1 5	TAK TAK NIE NIE NIE

Wejście	**Wyjście**
4 [(]) 1 1 4	NIE

Jenga (jenga)

Limit pamięci: 256 MB Limit czasu: 3.00 s

To zadanie jest interaktywne.

W zadaniu będą obowiązywać następujące zasady gry w Jengę: Gra polega na naprzemiennym wyciągania klocków z wieży i budowaniu z nich kolejnych pięter. Każde piętro zbudowane jest z trzech podłużnych klocków o proporcji długości i szerokości 3:1 ułożonych w szeregu obok siebie w kierunku prostopadłym do kierunku klocków piętra poniżej. Gracz, po ruchu którego wieża się przewróci, przegrywa. Uznajemy, że wieża zawsze się przewraca, gdy na piętrze został tylko skrajny klocek, lub gdy zostanie podjęta próba wyjęcia środkowego klocka, gdy nie ma już innych klocków na piętrze. Wyciągnięte klocki są odkładane na najwyższe piętro, chyba że ma już ono trzy klocki, w tej sytuacji jest tworzone nowe piętro. Nie można wyciągać klocków nad którymi nie znajduje się żaden klocek. Rozgrywka nie musi się rozpoczynać z pełnej wieży, to znaczy, że na początku mogą występować piętra zawierające tylko dwa lub nawet jeden klocek (w przypadku jednego klocka może być to tylko środkowy).

Krzysztof i Igor ostatnio bez przerwy grają w Jengę. Wciągnęli się w nią tak bardzo, że opanowali tę grę do perfekcji. Jeżeli któryś z nich przegrywa, to tylko dlatego, że nie ma już ruchu, który nie przewróciłby wieży. Niestety Igor się rozchorował i teraz Krzysztof nie ma z kim grać, zaprosił więc Ciebie do wspólnej rozgrywki. Jako że Krzysztof wie, że nie masz takiego doświadczenia w tej grze jak on oraz Igor, postanowił, że da ci fory i pozwoli ci rozpocząć rozgrywkę z pozycji wygrywającej. Aby jeszcze mocniej zachęcić Ciebie do rozgrywki, Krzysztof pozwolił ci napisać program komputerowy, który będzie pomagał ci w rozgrywce. Możesz założyć, że już kiedyś grałeś w Jengę i też tak, jak Krzysztof i Igor potrafisz idealnie wyjmować klocki z wieży.

Twoim zadaniem jest więc napisanie programu, który poda ci ruchy pozwalające na wygranie z Krzysztofem.

Protokół interakcji

Na początku interakcji w pierwszym wierszu wejścia będzie znajdować liczba naturalna N opisująca liczbę pięter początkowej wieży. W następnych N wierszach pojawi się opis początkowego stanu wieży. W i–tym wierszu będzie znajdował się opis i–tego piętra S_i składający się ze trzech znaków # oraz `.`, oznaczających odpowiednio klocek oraz pustą przestrzeń. Dla przykładu $S_3 =$ `##.` oznacza, że trzecie piętro zawiera wszystkie klocki oprócz prawego.

Po wczytaniu wieży należy rozpocząć rozgrywkę. Ty wykonujesz ruch jako pierwszy. Aby wyciągnąć j–ty klocek od lewej z i–tego piętra należy użyć zapytania poprzez wypisanie dwóch liczb `i j`. Po każdym zapytaniu program sprawdzający wypisze na standardowe wejście dwie liczby całkowite i oraz j, które oznaczają, że Krzysztof wyciągnął j–ty klocek z i–tego piętra. W przypadku zakończonej rozgrywki lub wykonania ruchu niezgodnego z zasadami, program sprawdzający wypisze na wejście `-1 -1`. Należy wtedy natychmiast zakończyć działanie programu.

Należy pamiętać o opróżnianiu bufora wypisywania po każdym zapytaniu. Aby to uczynić należy wykonać `cout.flush();` lub `cout << endl` jeżeli używamy cin/cout w C++, `fflush(stdout)` dla printf/scanf w C++, `sys.stdout.flush()` w Pythonie oraz `System.out.flush()` w Javie.

Ograniczenia

$2 \leq N \leq 50\,000$.

Podzadania

Podzadanie	Warunki	Punkty
1	$N \leq 5$	35
2	$N \leq 5\,000$	25
3	brak dodatkowych ograniczeń	40

Przykładowa interakcja

Wejście	Wyjście
2	
###	
#.#	
	1 1
1 3	
	2 2
-1 -1	

Wyjaśnienie przykładu: Pomimo niestandardowego ułożenia klocków, po wyciągnięciu klocka z pierwszego piętra w pierwszym ruchu jest on odłożony na puste miejsce na drugim piętrze. Po wykonaniu drugiego ruchu stało się możliwe wyciąganie klocków z drugiego piętra.

Plakatowanie 2 **(plakatowanie2)**

Limit pamięci: 256 MB Limit czasu: 3.00 s

Pamiętacie zadanie “Plakatowanie” z OI? Krzysztof jest w trakcie przygotowywania trudniejszej wersji tego zadania pod sparing. W zadaniu mamy dane N budynków stojących w szeregu jeden przy drugim, i–ty budynek jest reprezentowany przez prostokąt o szerokości W_i i wysokości H_i. Razem tworzą one długą ścianę, którą należy pokrywać plakatami. Plakaty są prostokątami, których boki są pionowe i poziome. Plakatów nie można ciąć, zginać, ani obracać; mogą one natomiast przybierać dowolne wymiary. Plakatowaniem nazwiemy całkowite pokrycie ściany plakatami, tak aby żaden plakat nie nachodził na siebie, ani nie wychodził poza obrys ściany. Należy szybko odpowiadać na zapytania o plakatowanie składające się z minimalnej liczby plakatów na segmencie ściany składającej się tylko i wyłącznie z budynków od L_i–tego do R_i–tego.

Aby zadanie nie było za łatwe, Krzysztof postanowił **w niektórych testach** wymusić odpowiadanie na zapytania w podanej na wejściu kolejności, poprzez szyfrowanie zapytań. Zamiast liczb L_i, R_i, na wejściu będą podane liczby K, A_i, B_i. Aby odzyskać wartości L_i, R_i należy użyć następujących wzorów:

$$L_i = ((K \cdot Ans_{i-1}) \oplus A_i)$$
$$R_i = ((K \cdot Ans_{i-1}) \oplus B_i)$$

gdzie $\oplus$ oznacza operację XOR, a Ans_i odpowiedź na i–te zapytanie. Uznajemy, że $Ans_0 = 0$.

Pomóż Krzysztofowi z testowaniem poprzez napisanie rozwiązania do opisanego powyżej zadania.

Wejście

W pierwszym wierszu wejścia znajdują się trzy liczby naturalne N, Q, K opisujące odpowiednio liczbę budynków, liczbę zapytań oraz parametr służący do odszyfrowywania zapytań. W następnych N wierszach znajdują się opisy budynków, gdzie i–ty z nich zawiera liczby naturalne W_i, H_i oznaczające odpowiednio szerokość oraz wysokość i–tego budynku. W następnych Q wierszach znajdują się opisy zapytań, gdzie i–ty z nich zawiera liczby naturalne A_i, B_i, które po odszyfrowaniu oznaczają zapytanie na przedziale $[L_i, R_i]$.

Wyjście

Należy wypisać Q wierszy. W i–tym z nich ma się znaleźć odpowiedź na i–te zapytanie.

Ograniczenia

$1 \leq N, Q \leq 500\,000, 0 \leq K \leq 1, 1 \leq W_i, H_i \leq 10^9, 1 \leq L_i, R_i \leq N, 1 \leq A_i, B_i \leq 2 \cdot N$.

Podzadania

Podzadanie	Warunki	Punkty
1	$N, Q \leq 5\,000$	30
2	$N, Q \leq 50\,000, K = 0$	20
3	$N, Q \leq 50\,000$	10
4	$N, Q \leq 500\,000, K = 0$	20
5	brak dodatkowych ograniczeń	20

Przykład

Wejście	Wyjście

```
4 2 0
2 2
1 3
2 4
3 2
1 4
2 3
```

```
3
2
```