Crafting (A)

Limit pamięci: 1024 MB Limit czasu: 2.00 s

Jasio spędza popołudnie przy swoim crafting table. Ma do dyspozycji siatkę 3×3, w której w każdym z 9 pól może umieścić przedmiot — lub zostawić je puste. Interesują go tylko trzy typy zawartości: “|” - patyk, “#” - diament, “.” - puste pole.

Jasio prosi Cię o sprawdzenie, czy aktualny stan jego crafting table tworzy jedno z trzech narzędzi: miecz, kilof lub siekierę (zorientowaną w dowolną stronę).

Jasio ma nadzieję, że znasz receptury tworzące te narzędzia, jednak jeśli tak nie jest, wszystkie receptury tworzące któreś z narzędzi są w testach przykładowych.

Wejście

Wejście składa się z 3 wierszy po 3 znaki każdy — każdy znak to `#`, `|` lub `.`, oznaczający odpowiednio diament, patyk lub puste pole.

Wyjście

Wypisz jedno słowo spośród: `MIECZ` — jeśli układ odpowiada mieczowi, `KILOF` — jeśli układ odpowiada kilofowi, `SIEKIERA` — jeśli układ odpowiada siekierze (dowolnej z dwóch orientacji), `NIE` — jeśli układ nie pasuje do żadnej z powyższych receptur.

Uwaga: wszystkie słowa muszą być zapisane wielkimi literami.

Przykład

Wejście	Wyjście

`###`
`.|.`
`.|.`
	`KILOF`

Wejście	Wyjście

`.#.`
`.#.`
`.|.`
	`MIECZ`

Wejście	Wyjście

`##.`
`#|.`
`.|.`
	`SIEKIERA`

Wejście	Wyjście

`.##`
`.|#`
`.|.`
	`SIEKIERA`

Wejście	Wyjście

`.#.`
`.|.`
`.|.`
	`NIE`

Cykl (B)

Limit pamięci: 1024 MB Limit czasu: 3.00 s

Janek ostatnio zainteresował się teorią grafów. Szczególnie spodobały mu się cykle — a najbardziej te najkrótsze. Dziś analizuje graf nieskierowany, prosty, składający się z n wierzchołków ponumerowanych od 1 do n i m krawędzi.

Janek chciałby sprawdzić, czy w jego grafie istnieje trójkąt zawierający wierzchołek 1, czyli taki ciąg trzech różnych wierzchołków, z których każdy jest połączony z następnym, a ostatni dodatkowo połączony z pierwszym, i który zawiera wierzchołek numer 1.

Janek uważa, że sprawdzanie tego ręcznie byłoby zbyt czasochłonne, więc poprosił cię o pomoc. Twoim zadaniem jest stwierdzić, czy istnieje taki cykl.

Wejście

W pierwszej linii znajdują się dwie liczby całkowite n i m — liczba wierzchołków i liczba krawędzi w grafie. W kolejnych m liniach znajdują się po dwie liczby u i v oznaczające, że w grafie istnieje krawędź między wierzchołkami u i v.

Wyjście

Wypisz `TAK`, jeśli istnieje cykl długości 3 zawierający wierzchołek 1, w przeciwnym razie wypisz `NIE`.

Uwaga: słowa `TAK` i `NIE` muszą być zapisane wielkimi literami.

Ograniczenia

- $1 \leq n, m \leq 3 \cdot 10^5$
- $1 \leq u, v \leq n$
- Graf jest prostym grafem nieskierowanym (nie zawiera pętli ani wielokrotnych krawędzi).

Przykład

Wejście
```
4 5
1 2
3 2
1 4
1 3
4 2
```

Wyjście
```
TAK
```

Wyjaśnienie
W pierwszym teście przykładowym mamy trójkąt (1,2,3), istnieją krawędzie 1-2, 2-3, 3-1.

Wejście
```
6 7
1 2
2 3
3 4
4 1
4 5
5 6
6 1
```

Wyjście
```
NIE
```

Wyjaśnienie
W drugiem teście przykładowym mamy cykle, ale nie są one długości 3.

Drzewo (C)

Limit pamięci: 1024 MB Limit czasu: 2.00 s

Dane jest drzewo nieskierowane o n wierzchołkach ponumerowanych od 1 do n. Każda z $n-1$ krawędzi drzewa ma przypisaną dodatnią wagę całkowitą.

Rozważamy wszystkie pary różnych wierzchołków (x, y) $(x < y)$. Dla każdej takiej pary definiujemy ścieżkę między x i y jako unikalną ścieżkę w drzewie. Niech l oznacza liczbę krawędzi na tej ścieżce (długość ścieżki), a w największą wagę spośród wszystkich krawędzi na tej ścieżce.

Para (x, y) jest dobra, jeśli zachodzi nierówność:

$$w - l \geq k$$

Oblicz, ile jest dobrych par (x, y).

Wejście

Pierwsza linia wejścia zawiera dwie liczby całkowite: n liczba wierzchołków drzewa, k współczynnik definiujący, czy para jest dobra.
Kolejnych $n-1$ linii opisuje krawędzie drzewa.
Każda z nich zawiera trzy liczby całkowite:
u, v wierzchołki połączone krawędzią, w waga tej krawędzi.

Wyjście

Wypisz jedną liczbę całkowitą — liczbę dobrych par (x, y).

Ograniczenia

- $1 \leq n, k \leq 100\,000$
- $1 \leq u, v \leq n$, $u \neq v$
- $1 \leq w \leq 100\,000$

Dane wejściowe tworzą spójne drzewo.

Przykład

Wejście
```
5 3
1 2 5
2 3 1
3 4 3
4 5 3
```

Wyjście
```
2
```

Wyjaśnienie
Pary dla których zachodzi warunek to {1, 2} i {1, 3}.

Wejście
```
5 2
1 2 4
3 1 1
2 4 4
2 5 3
```

Wyjście
```
7
```

Dzielniki (D)

Limit pamięci: 1024 MB Limit czasu: 3.00 s

Dany jest zestaw t testów. W każdym teście otrzymujesz cztery liczby całkowite: a, b, l oraz r.
Twoim zadaniem jest sprawdzić, czy istnieje liczba całkowita x taka, że:

- $l \leq x \leq r$
- $\gcd(x, a) = \gcd(x, b) = \gcd(a, b)$

Jeśli taka liczba istnieje, wypisz `TAK`, w przeciwnym razie wypisz `NIE`.

Wejście

Pierwsza linia zawiera jedną liczbę całkowitą t — liczbę testów.
W kolejnych t liniach znajdują się po cztery liczby całkowite a, b, l, r.

Wyjście

Dla każdego testu wypisz w osobnej linii słowo `TAK`, jeśli istnieje liczba x spełniająca warunki, lub `NIE` w przeciwnym razie.

Uwaga: słowa `TAK` i `NIE` muszą być zapisane wielkimi literami.

Ograniczenia

$1 \leq t \leq 10\,000$
$1 \leq a, b \leq 10^{18}$
$1 \leq l \leq r \leq 10^{18}$

Przykład

Wejście	Wyjście
3 6 15 1 10 10 25 6 7 17 31 2 3	TAK NIE TAK

Pac-Man (E)

Limit pamięci: 512 MB Limit czasu: 2.00 s

Na planszy o wymiarach $n \times n$ znajdują się:

- . - wolne pole, na którym można postawić Pac-Mana,
- # - pole zablokowane,
- cyfra c (od 1 do k) - oznaczenie początkowej pozycji duszka nr c.

Każdy z k duszków porusza się ruchem czterokierunkowym (góra, dół, lewo, prawo), nie może wchodzić na pola zablokowane i ma przypisaną prędkość s_i, czyli pokonuje s_i pól w ciągu jednej sekundy.

Chcemy wybrać wolne pole (.) na planszy tak, aby maksymalnie opóźnić moment, w którym któryś z duszków dosięgnie Pac-Mana. Duszki wyruszają natychmiast ze swoich początkowych pozycji i poruszają się w kierunku Pac-Mana najkrótszą dostępną ścieżką. Pac-man postawiony na pewnej pozycji będzie na niej czekał, aż do momentu jak złapie go jakiś duszek.

Dla dowolnego pola p i duszka i czas dojścia obliczamy jako:

$$\left\lceil \frac{d(z_i, p)}{s_i} \right\rceil$$

gdzie $d(z_i, p)$ to długość najkrótszej ścieżki od startowego pola duszka i do pola p, a s_i to prędkość duszka.

Chcemy udzielić odpowiedzi na t przypadków testowych.

Wejście

Pierwsza linia wejścia: t — liczba przypadków testowych.

Dla każdego z t przypadków:

- n, k — liczby całkowite opisujące wymiar planszy i liczbę duszków.
- Następnie n wierszy: każdy zawiera n znaków: . # lub cyfrę od 1 do k.
- Ostatnia linia: k liczb całkowitych $s_1, s_2, \ldots, s_k$ — prędkości kolejnych duszków.

Gwarantowane:

- Na planszy jest co najmniej jedno wolne pole (.).
- Każda cyfra od 1 do k występuje dokładnie raz.

Wyjście

- Jeśli istnieje wolne pole nieosiągalne dla żadnego duszka, wypisz: `NIE`
- W przeciwnym razie wypisz pojedynczą liczbę całkowitą — maksymalny czas w sekundach, po którym Pac-Man zostanie złapany.

Ograniczenia

- $1 \leq n \leq 100$
- $1 \leq k \leq 9$
- $1 \leq s_i \leq 100$
- Suma wartości n we wszystkich przypadkach testowych nie przekroczy $5\,000$

Przykład

Wejście

```
2
3 1
1..
..#
.#.
1
3 2
1#2
.#.
...
1 3
```

Wyjście

```
NIE
2
```

Wyjaśnienie

W pierwszym przypadku, jeśli ustawimy Pac-mana na polu (3, 3) duszek nie będzie miał możliwości się do niego dostać. W drugim przypadku testowym możemy ustawić Pac-mana na polu (3, 1) dla którego wynik to 2.

Podciągi (F)

Limit pamięci: 1024 MB Limit czasu: 3.00 s

Janek od zawsze lubił liczby, a w szczególności permutacje. Ostatnio zainteresowały go ciekawe własności pewnych ciągów. Dostał do analizy permutację P długości n, czyli ciąg, w którym każda liczba od 1 do n występuje dokładnie raz.

Janek chciałby policzyć, ile istnieje niepustych podciągów (niekoniecznie spójnych) tej permutacji (czyli takich ciągów, które powstają przez usunięcie dowolnych elementów, ale bez zmiany kolejności), które spełniają następujący warunek:

Dla każdego kolejnego elementu podciągu $x_1, x_2, \ldots, x_k$ (gdzie $k \geq 1$), zachodzi: - ($x_{i+1} \bmod x_i = 1$) dla $1 \leq i < k$

Janek szybko zauważył, że takich podciągów może być bardzo dużo, dlatego chciałby poznać tylko ich liczbę modulo $10^9 + 7$.

Twoim zadaniem jest pomóc mu w obliczeniu odpowiedzi.

Wejście

W pierwszym wierszu znajduje się jedna liczba całkowita n — długość permutacji.
W drugim wierszu znajduje się n liczb całkowitych $P_1, P_2, \ldots, P_n$ — permutacja liczb od 1 do n.

Wyjście

Wypisz jedną liczbę — liczbę podciągów spełniających warunek ($x_{i+1} \bmod x_i = 1$), modulo $10^9 + 7$.

Ograniczenia

- $1 \leq n \leq 10^6$
- P jest permutacją liczb od 1 do n

Przykład

Wejście	Wyjście	Wyjaśnienie
5 3 2 1 4 5	11	Podciągi spełniające warunek to: {3, 1}, {2, 1}, {3, 4}, {3, 4, 5}, {2, 5}, {4, 5}, oraz podciągi jednoelementowe, które zawsze są poprawne (czyli {3}, {2}, {1}, {4}, {5}) W sumie: 11 podciągów.

Premia (G)

Limit pamięci: 512 MB Limit czasu: 1.00 s

Szef w jednej z dużych firm programistycznych wymyślił oryginalny sposób na wypłacanie premii pracownikom. Każdy z n pracowników może samodzielnie wybrać wysokość swojej premii jako całkowite l w zakresie od 1 do k dolarów. Jednak istnieje haczyk: jeśli wybrana przez pracownika premia l przekroczy $c\%$ średniej ze wszystkich wybranych premii, to pracownik nie otrzyma nic.

Bardziej formalnie premię l otrzymuje pracownik tylko wtedy, gdy:

$$l \leq \frac{c \cdot S}{100 \cdot n}$$

gdzie:

- l — wybrana przez pracownika premia,
- S — suma wszystkich wybranych premii,
- n — liczba pracowników,
- c — procentowy próg przyznania premii.

W przeciwnym razie premia przepada (jest równa 0).

Pracownicy szybko zorientowali się w podstępie szefa i proszą Cię o pomoc w wybraniu strategii, która pozwoli zmaksymalizować łączną sumę otrzymanych premii. Ponadto wiele firm poszło za tym przykładem i twój program powinien udzielić odpowiedzi dla q zapytań.

Wejście

Pierwsza linia wejścia zawiera jedną liczbę naturalną q.

Następne q linii zawierają opis kolejnych zapytań

- n — liczba pracowników
- k — maksymalna wartość premii, jaką może wybrać pracownik
- c — wartość procentowa progu

Wyjście

Dla każdego zapytania wypisz jedną liczbę całkowitą — największą możliwą sumę otrzymanych premii przy optymalnym wyborze strategii przez pracowników.

Ograniczenia

- $1 \leq q \leq 1\,000$
- $2 \leq n \leq 100\,000$
- $5 \leq k \leq 10^9$
- $5 \leq c \leq 95$

Suma n we wszystkich przypadkach testowych nie przekroczy $100\,000$.

Przykład

Wejście	Wyjście	Wyjaśnienie
2 3 10 50 2 3 25	4 0	W pierwszym przypadku jeden pracownik może wybrać premię 10 a dwóch pozostałych premię 2 i jest to optymalna strategia. W drugim przypadku niezależnie od wyboru pracowników otrzymana suma będzie równa 0.

Rajdowiec (H)

Limit pamięci: 512 MB Limit czasu: 1.00 s

Rajdowiec startuje z nowymi oponami o początkowej wytrzymałości k. Przejeżdża kolejne okrążenia w następujący sposób:

1. Dopóki aktualna wytrzymałość opon d jest dodatnia ($d > 0$):
   - Przejeżdża d okrążeń.
   - Po przejechaniu tych okrążeń wytrzymałość opon zmniejsza się o wartość m:

$$d \leftarrow d - m.$$

2. Gdy $d \leq 0$, rajdowiec przestaje jeździć.

Chcemy obliczyć, ile okrążeń w sumie przejedzie rajdowiec, zanim wytrzymałość opon spadnie do zera lub poniżej.

Wejście

W pierwszej linii znajduje się liczba całkowita q — liczba zapytań. W każdej z kolejnych q linii dwie liczby całkowite k_i oraz m_i.

Wyjście

Dla każdego zapytania wypisz w osobnej linii łączną liczbę okrążeń, które rajdowiec przejedzie, zanim wytrzymałość opon stanie się mniejsza lub równa zero.

Ograniczenia

- $1 \leq q \leq 1\,000$
- $1 \leq k_i, m_i \leq 10^9$

Przykład

Wejście	Wyjście	Wyjaśnienie
3 10 3 5 2 7 7	22 9 7	Dla $k = 10$, $m = 3$ kolejność wytrzymałości: 10 $\rightarrow$ 7 $\rightarrow$ 4 $\rightarrow$ 1 $\rightarrow$ -2, suma okrążeń $10 + 7 + 4 + 1 = 22$. Dla $k = 5$, $m = 2$: 5 $\rightarrow$ 3 $\rightarrow$ 1 $\rightarrow$ -1, suma $5 + 3 + 1 = 9$. Dla $k = 7$, $m = 7$: 7 $\rightarrow$ 0, suma 7.

Skoki (I)

Limit pamięci: 512 MB Limit czasu: 1.00 s

Poruszamy się po okręgu z polami ponumerowanymi od 0 do $n-1$. Zaczynamy w polu 0, a następnie wykonujemy dowolną sekwencję skoków o długości k_1 lub k_2 zgodnie z ruchem wskazówek zegara.

Chcemy sprawdzić, czy możliwe jest odwiedzenie wszystkich pól okręgu (czyli z pola 0 da się w kolejnych krokach wejść przynajmniej raz do każdego pola).

Mamy dowolną liczbę kroków oraz możemy odwiedzać poszczególne pola wielokrotnie.

Wejście

W pierwszej linii wejścia znajduje się liczba całkowita q — liczba zapytań. Każde z kolejnych q wierszy zawiera trzy liczby całkowite n, k_1 oraz k_2.

Wyjście

Dla każdego zapytania wypisz w osobnej linii słowo `TAK`, jeśli da się odwiedzić wszystkie pola, lub `NIE` w przeciwnym wypadku.

Uwaga: słowa `TAK` i `NIE` muszą być zapisane wielkimi literami.

Ograniczenia

- $1 \leq q \leq 1\,000$
- $1 \leq n \leq 10^9$
- $1 \leq k_1, k_2 \leq n-1$

Przykład

Wejście

```
3
6 2 3
7 2 4
6 2 4
```

Wyjście

```
TAK
TAK
NIE
```

Wyjaśnienie

Dla $n = 6$, $k_1 = 2$, $k_2 = 3$: przykładowa sekwencja skoków umożliwiająca odwiedzenie wszystkich pól to:

$$0 \xrightarrow{+2} 2 \xrightarrow{+2} 4 \xrightarrow{+3} 1 \xrightarrow{+2} 3 \xrightarrow{+2} 5$$

Dla $n = 7$, $k_1 = 2$, $k_2 = 4$: wystarczy wykonywać skok $+2$ sześć razy:

$$0 \xrightarrow{+2} 2 \xrightarrow{+2} 4 \xrightarrow{+2} 6 \xrightarrow{+2} 1 \xrightarrow{+2} 3 \xrightarrow{+2} 5$$

Dla $n = 6$, $k_1 = 2$, $k_2 = 4$: da się udowodnić, że nie ma możliwości odwiedzenia wszystkich pól.

Podział apartamentowca (J)

Limit pamięci: 256 MB Limit czasu: 1.50 s

Deweloper Januszex S.A. właśnie buduje wielki apartamentowiec. Żeby oszczędzić na budowie, będzie miał on kształt idealnego prostopadłościanu, a wszystkie jego piętra będą miały taki sam układ. Dla uproszczenia pomińmy wysokość pomieszczeń i skupmy się więc na projekcie pojedynczego piętra patrząc z góry. Piętro ma więc wymiary $A \times B$ metrów i trzeba je podzielić na mieszkania.

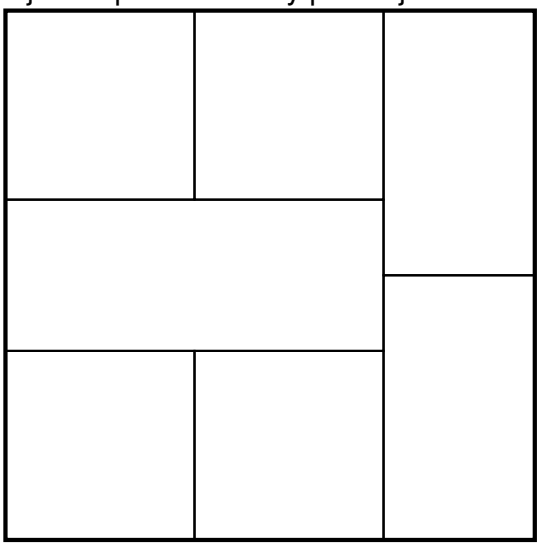
Ponieważ deweloper oszczędza na ludziach, nie ma ani dobrych inżynierów, ani dobrych księgowych – z tego względu boki wszystkich mieszkań muszą być równoległe do krawędzi całego budynku (kąty proste są prostsze dla inżynierów) oraz muszą mieć całkowite długości w metrach (księgowi łatwiej obliczają wtedy powierzchnię mieszkania używając w tym celu tabliczki mnożenia, a nie kalkulatora). A więc na mapie z góry trzeba narysować trochę ścian równoległych do boków prostokąta ograniczającego piętro apartamentowca, żeby każde mieszkanie było prostokątem o wymiarach $a_i \times b_i$, gdzie wartości a_i, b_i muszą być całkowite.

Żeby zarobić jak najwięcej, deweloper postanowił podzielić apartamentowiec na jak najwięcej mieszkań. Rozmiar mieszkania nie miałby żadnego znaczenia dla ceny, bo zdesperowany klient kupi wszystko co będzie dostępne (jako alternatywę ma spanie pod mostem). Niestety, jest jeszcze prawo budowlane, które uznaje za mieszkania tylko te o powierzchni co najmniej K metrów kwadratowych.

Deweloper chce wykorzystać każdy centymetr kwadratowy powierzchni, dlatego każda powierzchnia piętra musi być przypisana do jakiegoś mieszkania. Ponadto można przyjąć, że ściany mieszkań są pomijalnej grubości (są wykonane z kartonu, akustyką i wytrzymałością będą się przejmować mieszkańcy, a nie deweloper).

Jest jednak jeszcze jeden problem: do każdego mieszkania musi docierać naturalne światło z zewnątrz budynku. A więc każde mieszkanie musi mieć chociaż jedną ze ścian, która jest częścią zewnętrzną apartamentowca (w odpowiednim miejscu deweloper zrobi malutkie okno i wszystko będzie w zgodzie z przepisami). Nie przejmujemy się natomiast sposobem wejścia do mieszkania: najwyżej mieszkańcy sobie kupią odpowiednią drabinę i jakoś sobie poradzą.

Przykładowy sensowny podział piętra o wymiarach 14×14 na mieszkania o powierzchniach co najmniej 25 mkw jest zaprezentowany poniżej.



Napisz program, który wczyta wymiary piętra A, B oraz minimalną powierzchnię mieszkania K i wyznaczy największą liczbę mieszkań, które można upchnąć na tej powierzchni zgodnie z warunkami zadania.

Wejście

W pierwszym (jedynym) wierszu wejścia znajdują się trzy liczby naturalne A, B, K, pooddzielane pojedynczymi odstępami.

Wyjście

W pierwszym (jedynym) wierszu wyjścia powinna się znaleźć jedna liczba naturalna – największa liczba mieszkań, które można zmieścić na piętrze zgodnie z warunkami zadania.

Ograniczenia

$1 \leq A, B \leq 300, 1 \leq K \leq A \cdot B$.

Przykład

Wejście	Wyjście
`14 14 25`	7

Wyjaśnienie

Test jest zgodny z rysunkiem powyżej w treści. Można zrobić siedem mieszkań: na przykład cztery kwadratowe apartamenty o powierzchni 25 mkw, dwa apartamenty premium o powierzchni 28 mkw (4×7) oraz apartament luksusowy o powierzchni 40 mkw (4×10).