amppz

Prostokąt arytmetyczny ARY

Dostępna pamięć: 128 MB.

Na każdym polu kartki w kratkę, składającej się z $n \times m$ kwadratów jednostkowych, zapisano jedną liczbę całkowitą. W tym zadaniu interesują nas *prostokąty arytmetyczne* położone na tej kartce, czyli takie prostokąty złożone z kwadratów jednostkowych, że liczby w każdym wierszu i w każdej kolumnie tworzą ciągi arytmetyczne. Przypomnijmy, że ciąg arytmetyczny to ciąg liczbowy, w którym każde dwa kolejne wyrazy różnią się o tę samą liczbę.

Na danej kartce w kratkę poszukujemy największego prostokąta arytmetycznego, tj. obejmującego najwięcej kwadratów jednostkowych. Przykładowo, największy prostokąt arytmetyczny na poniższej kartce składa się z dziewięciu kwadratów jednostkowych:

5	3	5	7
2	4	4	4
3	5	3	1
6	3	2	4

Wejście

W pierwszym wierszu wejścia znajduje się jedna liczba całkowita t ($1 \leqslant t \leqslant 10\,000$) oznaczająca liczbę zestawów testowych opisanych w dalszej części wejścia. Opis każdego zestawu rozpoczyna się od wiersza z dwiema liczbami całkowitymi n i m ($1 \leqslant n, m \leqslant 3000$). W każdym z kolejnych n wierszy znajduje się m liczb całkowitych z zakresu od 0 do 10^9. Są to liczby wpisane w poszczególne kwadraty jednostkowe kartki w kratkę. Rozmiar każdego pliku wejściowego będzie nie większy niż 20 MB.

Wyjście

Należy wypisać t wierszy z odpowiedziami dla kolejnych zestawów testowych. Odpowiedzią dla jednego zestawu jest jedna liczba całkowita równa liczbie kwadratów jednostkowych zawartych w największym prostokącie arytmetycznym na kartce opisanej w danym zestawie.

Przykład

Dla danych wejściowych:

```
2
4 4
5 3 5 7
2 4 4 4
3 5 3 1
6 3 2 4
2 3
0 1 2
1 2 3
```

poprawnym wynikiem jest:

```
9
6
```

Bajtocki Bieg Uliczny — BAJ

Dostępna pamięć: 64 MB.

W centrum Bajtogrodu ma się jutro odbyć Bajtocki Bieg Uliczny. Ulice w Bajtogrodzie tworzą równomierną kratkę: wszystkie prowadzą z południa na północ bądź z zachodu na wschód. Uczestnikom biegu udostępniono jedynie pewne ich fragmenty.

Bajtazar ma się zająć rozstawieniem reklam sponsorów imprezy na niektórych skrzyżowaniach i w tym celu musi przyjrzeć się mapie trasy biegu. Mapa przedstawia fragmenty ulic, które udostępniono biegaczom. Zaznaczono na niej n skrzyżowań oraz m pionowych i poziomych odcinków ulic. Każdy odcinek zaczyna się i kończy na jakimś skrzyżowaniu i nie zawiera żadnych innych skrzyżowań. Odcinki ulic nie przecinają się poza skrzyżowaniami.

Skrzyżowania są ponumerowane od 1 do n. Bieg ma rozpocząć się na skrzyżowaniu numer 1 i zakończyć na skrzyżowaniu numer n. Biegacze mogą wybrać swoją trasę biegu, przy czym zobowiązani są biec tylko na południe lub na wschód i jedynie po odcinkach zaznaczonych na mapie. Odcinki ulic na mapie są dobrane tak, że biegnąc zgodnie z zasadami, z każdego miejsca da się dotrzeć do mety i każde miejsce jest osiągalne ze skrzyżowania startowego.

Bajtazar chciałby porozwieszać reklamy tak, żeby mieć pewność, że żaden biegacz nie zobaczy dwukrotnie reklamy tego samego sponsora. Wobec tego, dla niektórych par skrzyżowań Bajtazar musi sprawdzić, czy możliwe jest, by trasa jakiegoś uczestnika przebiegała przez obydwa skrzyżowania. Bieg startuje już jutro, dlatego pilnie potrzebny jest program, który pomoże mu w pracy.

Wejście

Pierwszy wiersz wejścia zawiera trzy liczby całkowite n, m i k ($2 \leqslant n \leqslant 100\,000$, $1 \leqslant m \leqslant 200\,000$, $1 \leqslant k \leqslant 300\,000$). Oznaczają one odpowiednio liczbę skrzyżowań na trasie biegu, liczbę odcinków na mapie oraz liczbę par skrzyżowań do sprawdzenia.

Kolejne n wierszy opisuje położenia skrzyżowań. W i-tym spośród nich znajdują się współrzędne i-tego skrzyżowania w postaci dwóch liczb całkowitych x_i, y_i ($-10^9 \leqslant x_i, y_i \leqslant 10^9$). Dodatkowo, $x_1 \leqslant x_n$ i $y_1 \geqslant y_n$. W danym miejscu może znajdować się co najwyżej jedno skrzyżowanie. Osie układu współrzędnych utożsamiamy w naturalny sposób z kierunkami świata: oś OX prowadzi na wschód, zaś oś OY — na północ.

Każdy z kolejnych m wierszy zawiera opis jednego odcinka na mapie składający się z pary liczb całkowitych a_i, b_i ($1 \leqslant a_i, b_i \leqslant n$, $a_i \neq b_i$) oznaczających numery skrzyżowań połączonych tym odcinkiem. Wszystkie te odcinki są pionowe lub poziome i nie mają punktów wspólnych poza wspólnymi końcami na skrzyżowaniach.

W następnych k wierszach znajdują się opisy par skrzyżowań do sprawdzenia. W i-tym z tych wierszy znajdują się dwie liczby całkowite p_i, q_i ($1 \leqslant p_i, q_i \leqslant n$, $p_i \neq q_i$).

Wyjście

Twój program powinien wypisać k wierszy. W i-tym spośród tych wierszy powinno znaleźć się słowo `TAK`, jeśli trasa pewnego uczestnika biegu może prowadzić przez skrzyżowania p_i i q_i (w dowolnej kolejności). W przeciwnym wypadku należy wypisać `NIE`.

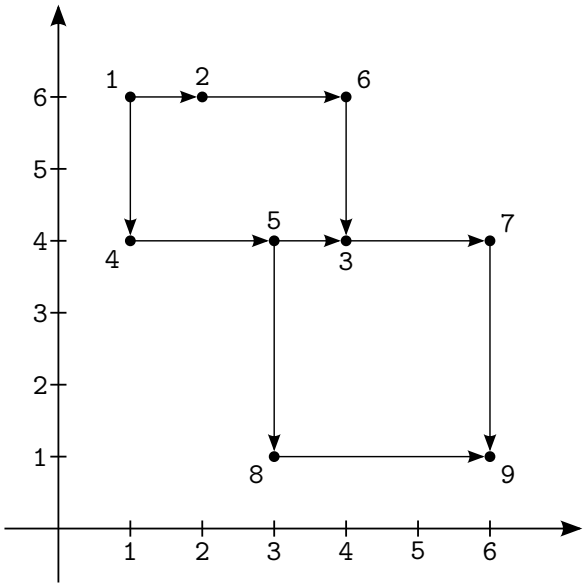
Przykład

Dla danych wejściowych:

```
9 10 4
1 6
2 6
4 4
1 4
3 4
4 6
6 4
3 1
6 1
1 2
4 1
2 6
3 6
5 4
5 3
5 8
3 7
7 9
9 8
4 8
2 5
8 7
7 6
```

poprawnym wynikiem jest:

```
TAK
NIE
NIE
TAK
```



amppz

Czy się zatrzyma? CZY

Dostępna pamięć: 64 MB.

Bajtazar przechadzał się koło Biblioteki Uniwersyteckiej w Warszawie i na jednej z fasad zobaczył fragment programu opatrzony pytaniem „Czy się zatrzyma?". Problem wyglądał intrygująco, dlatego Bajtazar postanowił zająć się nim po powrocie do domu. Niestety, gdy zapisywał kod na kartce, popełnił błąd i zanotował:

```
while n > 1 do
  if n mod 2 = 0 then
    n := n/2
  else
    n := 3 · n + 3
```

Bajtazar próbuje teraz ustalić, dla jakich wartości początkowych zmiennej n zapisany przez niego program zatrzyma się. Zakładamy przy tym, że zmienna n ma nieograniczony rozmiar, tj. może przyjmować dowolnie duże wartości.

Wejście

Pierwszy i jedyny wiersz wejścia zawiera jedną liczbę całkowitą n ($2 \leqslant n \leqslant 10^{14}$), dla której należy sprawdzić, czy podany program zatrzyma się.

Wyjście

W pierwszym i jedynym wierszu wyjścia Twój program powinien wypisać jedno słowo `TAK`, jeśli program zatrzyma się dla podanej wartości n, lub `NIE` w przeciwnym przypadku.

Przykład

Dla danych wejściowych:

```
4
```

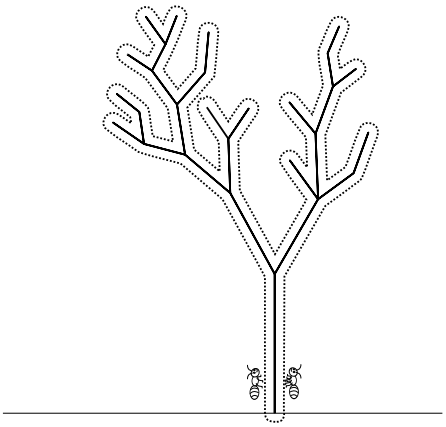
poprawnym wynikiem jest:

```
TAK
```

Drzewo i mrówki

DRZ

Dostępna pamięć: 6 MB.



Informatycy lubią drzewa. Mrówki także lubią drzewa. Niech więc będzie dane drzewo, po którym chodzą dwie mrówki — Lewa Mrówka i Prawa Mrówka — w sposób pokazany na powyższym rysunku (po ścieżce oznaczonej kropkowaną linią). Rozpoczynają one swoją wędrówkę na początku pnia, po przeciwnych jego stronach. Lewa Mrówka pokonuje każdą krawędź drzewa w czasie dwóch sekund, idąc od korzenia (w górę), a w czasie jednej sekundy, idąc w kierunku korzenia (w dół). Prawa Mrówka jest od niej dwa razy szybsza. W momencie, gdy mrówki spotykają się, obydwie zawracają. Jeśli któraś z nich zejdzie z drzewa na ziemię, natychmiast zaczyna wchodzić na drugą stronę pnia. Poza tym mrówki są tak malutkie, że nawet mikroskop nie wystarczyłby do ich zobaczenia (na rysunku celowo zostały powiększone). Twoim zadaniem jest napisanie programu, który obliczy, po jakim czasie mrówki zawrócą po raz drugi.

Wejście

W pierwszym wierszu wejścia znajduje się jedna liczba całkowita t ($1 \leqslant t \leqslant 1000$) oznaczająca liczbę zestawów testowych opisanych w dalszej części wejścia.

Opis każdego zestawu składa się z dwóch wierszy. W pierwszym z nich znajduje się parzysta liczba n ($2 \leqslant n \leqslant 100\,000\,000$) oznaczająca liczbę krawędzi drzewa. W drugim wierszu znajduje się opis drzewa. Jest to napis długości $\frac{n}{2}$ reprezentujący liczbę binarną $2n$-bitową zapisaną w systemie szesnastkowym (przy użyciu cyfr oraz małych liter od a do f). Liczba ta opisuje, jak wygląda obejście całego drzewa przez Lewą Mrówkę przy założeniu, że Prawa Mrówka stoi w miejscu. Kolejne bity tej liczby (od lewej) oznaczają, czy idąc po kolejnych krawędziach na swojej ścieżce, Lewa Mrówka oddala się od korzenia (bit 1), czy też zbliża się do korzenia (bit 0). Drzewo posiada pień, tzn. dokładnie jedna krawędź wychodzi z korzenia drzewa.

Rozmiar żadnego pliku wejściowego nie przekroczy 50 MB. Uwaga: to jest istotnie więcej niż rozmiar pamięci dostępnej dla Twojego programu.

Wyjście

Twój program powinien wypisać t wierszy z odpowiedziami dla kolejnych zestawów testowych. Odpowiedzią dla zestawu jest czas w sekundach, po którym mrówki zawrócą po raz drugi, wypisany jako nieskracalny ułamek p/q (bez spacji wokół /), gdzie p i q to liczby całkowite dodatnie. Jeśli odpowiedź jest liczbą całkowitą, to oczywiście $q = 1$.

Przykład

Dla danych wejściowych:

```
1
28
fb1da30d1b7230
```

poprawnym wynikiem jest:

282/5

Przykład odpowiada rysunkowi, a ciąg bitów wygląda następująco:

```
1111 1011 0001 1101 1010 0011 0000 1101 0001 1011 0111 0010 0011 0000
```

Eksterminacja świstaków EKS

Dostępna pamięć: 128 MB.

Gnębon Bajtopysk postanowił uprzykrzyć życie bajtockim świstakom. Te sympatyczne zwierzątka zamieszkują w norkach, w górnych partiach pasma Bajtogór Wysokich.

Gnębon odnalazł pewien grzbiet górski, wzdłuż którego w jednej linii jest rozmieszczonych n świstaczych norek (dla ułatwienia kolejne norki wzdłuż grzbietu, patrząc z zachodu na wschód, numerujemy od 1 do n). Diabelski pomysł Gnębona polega na gnębieniu świstaków muzyką z gatunku rock and roll. Aby go zrealizować, nasz bohater zakupił m magnetofonów, w każdym umieścił inny album Bajtelsów i rozmieścił je wszystkie w linii, wzdłuż której są wykopane norki. Wiadomo, że po rozkręceniu głośności w danym magnetofonie na cały regulator, wydobywająca się z niego muzyka nie daje spać świstakom znajdującym się w norkach odległych co najwyżej o l metrów od tego magnetofonu.

Zaniepokojone tą sytuacją świstaki poprosiły Cię o sprawdzenie, w których norkach na pewno nie uda im się tej zimy wyspać. Nie wiedziały, że to jeszcze nie koniec złośliwości Gnębona...

Otóż Gnębon postanowił wywołać jeszcze większe zamieszanie i co jakiś czas przestawiać niektóre magnetofony. Świstakom udało się wykraść tajny plan Gnębona i wiedzą dokładnie, że i-tego dnia rankiem weźmie on magnetofon położony p_i metrów od norki numer 1 i przestawi go w punkt odległy o r_i metrów od tejże norki. Pomóż świstakom sprawdzić, w ilu norkach nie uda im się zasnąć po każdej takiej zamianie.

Wejście

W pierwszym wierszu wejścia znajdują się cztery liczby całkowite n, m, d oraz l ($2 \leqslant n, m \leqslant 500\,000$, $1 \leqslant d \leqslant 500\,000$, $1 \leqslant l \leqslant 10^9$) oznaczające odpowiednio liczbę norek świstaków, liczbę magnetofonów Gnębona, liczbę dni „eksperymentu” Gnębona i zasięg rażenia magnetofonu.

Drugi wiersz wejścia zawiera $n-1$ liczb całkowitych $x_2, x_3, \ldots, x_n$ ($0 < x_2 < x_3 < \ldots < x_n \leqslant 10^9$) oznaczających odległości norek o numerach $2, 3, \ldots, n$ od norki numer 1.

Trzeci wiersz zawiera m liczb całkowitych $z_1, z_2, \ldots, z_m$ ($0 \leqslant z_1 < z_2 < \ldots < z_m \leqslant 10^9$) oznaczających odległości kolejnych magnetofonów od norki numer 1. Wszystkie magnetofony położone są na wschód od tej norki.

Dalej na wejściu następuje d wierszy; i-ty z nich zawiera dwie liczby całkowite p_i oraz r_i ($0 \leqslant p_i, r_i \leqslant 10^9$, $p_i \neq r_i$) oznaczające, że na początku i-tego dnia „zabawy” Gnębon zamierza przestawić magnetofon znajdujący się p_i metrów od norki numer 1 w punkt odległy o r_i metrów na wschód od tej norki. Możesz założyć, że przed wykonaniem takiej operacji na pozycji p_i znajduje się jakiś magnetofon oraz że na pozycji r_i nie ma jeszcze żadnego magnetofonu.

Wyjście

Twój program powinien wypisać $d+1$ wierszy. Wiersz numer i (dla $i = 1, 2, \ldots, d$) powinien zawierać jedną liczbę całkowitą, oznaczającą liczbę norek, w których żaden świstak na pewno się nie wyśpi w nocy *przed* wykonaniem i-tej zamiany. W ostatnim wierszu należy wypisać, w ilu norkach świstaki nie będą mogły spać po ostatniej zamianie.

amppz

Przykład

Dla danych wejściowych:

```
5 3 4 1
2 5 6 11
2 4 8
2 1
4 10
8 6
1 8
```

poprawnym wynikiem jest:

```
2
3
3
5
3
```

Frania FRA

Dostępna pamięć: 128 MB.

Kilkoro przyjaciół postanowiło zrobić pranie. Wszyscy oni są zupełnie porządni, wobec czego każdego dnia zużywają dokładnie jedną parę skarpetek i jedną koszulkę. Wrzucili więc wszystkie zużyte skarpetki i koszulki do swojej wysłużonej pralki i zaczęli zastanawiać się nad strategią ich wysuszenia. Żeby wyeliminować zamieszanie, postanowili, że:

- każda skarpetka będzie przypięta do sznurka jedną klamerką,
- każda koszulka będzie przypięta trzema klamerkami,
- wszystkie skarpetki jednej osoby będą przypięte klamerkami tego samego koloru,
- wszystkie koszulki jednej osoby będą przypięte klamerkami tego samego koloru,
- rzeczy należące do dwóch różnych osób nie mogą być przypięte klamerkami tego samego koloru,
- poza tym użyją najmniejszej możliwej liczby kolorów klamerek.

Tak ustaliwszy, wysypali wszystkie posiadane klamerki na podłogę i skrzętnie policzyli, ile mają klamerek każdego z kolorów. Niestety nie potrafili wykombinować, kto powinien użyć których. Napisz program, który im pomoże.

Wejście

W pierwszym wierszu wejścia znajdują się dwie liczby całkowite n oraz k ($2 \leqslant n, k \leqslant 1\,000\,000$) oznaczające liczbę przyjaciół oraz liczbę dostępnych kolorów klamerek. W drugim wierszu znajduje się n liczb $d_1, d_2, \ldots, d_n$ oznaczających, przez ile dni poszczególni przyjaciele gromadzili pranie ($1 \leqslant d_i \leqslant 1\,000\,000$). W trzecim wierszu znajduje się k liczb $l_1, l_2, \ldots, l_k$ oznaczających, ile jest klamerek poszczególnych kolorów ($1 \leqslant l_i \leqslant 4\,000\,000$).

Wyjście

Twój program powinien wypisać minimalną liczbę kolorów klamerek potrzebnych do rozwieszenia prania. Jeśli rozwieszenie prania w opisany sposób nie jest możliwe, Twój program powinien wypisać jedno słowo `NIE`.

Przykład

Dla danych wejściowych:

```
2 4
3 4
20 10 8 10
```

poprawnym wynikiem jest:

```
3
```

natomiast dla danych:

```
3 8
5 4 3
14 14 14 14 14 14 14 14
```

poprawnym wynikiem jest:

```
NIE
```

Wyjaśnienie do pierwszego przykładu: Pierwsza osoba potrzebuje 6 klamerek na skarpetki i 9 na koszulki. Druga osoba potrzebuje 8 klamerek na skarpetki i 12 na koszulki. Druga osoba powinna użyć klamerek pierwszego koloru zarówno do powieszenia skarpetek, jak i koszulek. Pierwsza osoba może wówczas użyć na przykład klamerek drugiego i czwartego koloru.

amppz

Generator bitów GEN

Dostępna pamięć: 64 MB.

Bajtazar lubi bawić się generatorem bitów losowych (w zasadzie pseudolosowych), który jest dostarczany przez oprogramowanie w jego komputerze. Zasada działania tego generatora jest bardzo prosta. Przy starcie komputera w magiczny sposób jest wybierana liczba całkowita z zakresu od 0 do $m-1$, którą nazwiemy *ziarnem* generatora i będziemy zapisywać w zmiennej z. Następnie w celu wygenerowania losowego bitu wywoływana jest poniższa funkcja, która oblicza nowe ziarno i na jego podstawie wyznacza zwracany bit:

$z := \lfloor (z \cdot a + c)/k \rfloor \bmod m$
if $z < \lfloor m/2 \rfloor$ then
  return 0
else
  return 1

Używane wartości a, c, k są pewnymi stałymi. Bajtazar wywołał powyższą funkcję n razy i uzyskał ciąg kolejnych bitów $b_1, b_2, \ldots, b_n$. Zastanawia się teraz, ile możliwych wartości mogło przyjąć początkowe ziarno generatora.

Wejście

W pierwszym wierszu wejścia znajduje się pięć liczb całkowitych a, c, k, m i n ($0 \leqslant a, c < m$, $1 \leqslant k < m$, $2 \leqslant m \leqslant 1\,000\,000$, $1 \leqslant n \leqslant 100\,000$). W drugim wierszu znajduje się n-literowy napis złożony z cyfr 0 i 1; i-ta cyfra napisu oznacza bit b_i.

Wyjście

Należy wypisać jedną liczbę całkowitą, która oznacza liczbę liczb z zakresu od 0 do $m-1$, które mogły być początkowym ziarnem generatora bitów.

Przykład

Dla danych wejściowych:

```
3 6 2 9 2
10
```

poprawnym wynikiem jest:

```
4
```

Wyjaśnienie do przykładu: Ziarnem generatora bitów mogło być 1, 2, 7 lub 8.

amppz

Herbata z mlekiem

HER

Dostępna pamięć: 32 MB.

Bajtazar podczas ostatniej wizyty na Wyspach Bitockich bardzo zasmakował w narodowym napoju Bitocjan, którym jest herbata z mlekiem. Sposób przyrządzania i picia tego napoju jest ściśle określony i przebiega według poniższego przepisu. Najpierw napełnia się filiżankę w połowie herbatą, w połowie mlekiem i dokładnie miesza. Następnie ustala się n-literowe *słowo ceremoniału* złożone z liter `H` i `M`. Teraz kolejno dla $i = 1, 2, \ldots, n$ wykonuje się, co następuje: jeśli i-tą literą słowa ceremoniału jest `H`, to należy wypić połowę napoju z filiżanki, a następnie dolać do pełna herbaty i zamieszać. Jeśli zaś i-tą literą tego słowa jest `M`, to wykonuje się to samo, tylko zamiast herbaty dolewa się mleka. Po wykonaniu tych czynności dla każdej litery słowa ceremoniału, pozostały w filiżance napój należy wylać.

Za każdym razem, gdy Bajtazar odprawi powyżej opisaną ceremonię, zastanawia się, czy wypił więcej herbaty, czy też mleka. Pomóż Bajtazarowi w rozwikłaniu tej zagadki.

Wejście

W pierwszym wierszu wejścia znajduje się liczba całkowita n ($1 \leqslant n \leqslant 100\,000$). W drugim wierszu znajduje się n-literowe słowo złożone z liter `H` i `M` — jest to słowo ceremoniału, którego użył Bajtazar.

Wyjście

Twój program powinien wypisać literę `H`, jeśli Bajtazar wypił więcej herbaty niż mleka; literę `M`, jeśli wypił więcej mleka niż herbaty; lub słowo `HM`, jeśli wypił tyle samo herbaty co mleka.

Przykład

Dla danych wejściowych:	poprawnym wynikiem jest:
5 HMHHM	H

Wyjaśnienie do przykładu: Bajtazar wypił w sumie $1\frac{37}{64}$ filiżanki herbaty i $\frac{59}{64}$ filiżanki mleka.

amppz

Iloraz inteligencji — ILO

Dostępna pamięć: 128 MB.

Na Uniwersytecie Bajtockim można studiować jedynie dwa kierunki: matematykę i informatykę. Aktualnie na uniwersytecie uczy się n studentów matematyki oraz m studentów informatyki. Kierunki te są bardzo trudne i nie istnieje student kształcący się w obu z nich jednocześnie.

Rektor Bajtazar zna iloraz inteligencji każdego ze studentów. Na tej podstawie chciałby wyznaczyć zespół studentów, który poradzi sobie z najtrudniejszymi problemami ludzkości. Postanowił więc, że wybierze zespół o możliwie dużej sumie ilorazów inteligencji wszystkich członków.

Iloraz inteligencji to jednak nie wszystko. Dlatego rektor chciałby, aby wszyscy członkowie zespołu się znali. Wiadomo, że wszyscy studenci matematyki się znają. Podobnie, każdy student informatyki zna każdego innego studenta informatyki.

Pomóż rektorowi i napisz program, który wyznaczy zespół studentów o możliwie dużej sumie ilorazów inteligencji, w którym wszyscy się znają.

Wejście

W pierwszym wierszu wejścia znajdują się trzy liczby całkowite n, m i k ($1 \leqslant n, m \leqslant 400$, $0 \leqslant k \leqslant n \cdot m$), które oznaczają odpowiednio liczbę studentów matematyki, liczbę studentów informatyki oraz liczbę par osób z różnych kierunków, które się znają.

Każdy z kolejnych k wierszy zawiera opis jednej pary znajomych: i-ty z tych wierszy zawiera dwie liczby całkowite a_i i b_i ($1 \leqslant a_i \leqslant n$, $1 \leqslant b_i \leqslant m$) oznaczające, odpowiednio, numer studenta matematyki i numer studenta informatyki z i-tej pary. Zarówno studentów matematyki, jak i studentów informatyki numerujemy liczbami całkowitymi, poczynając od 1.

W następnym wierszu znajduje się n liczb całkowitych z przedziału od 1 do 10^9, które oznaczają ilorazy inteligencji kolejnych studentów matematyki. Kolejny wiersz zawiera m liczb opisujących ilorazy inteligencji studentów informatyki, w analogicznym formacie.

Wyjście

W pierwszym wierszu wyjścia należy wypisać jedną liczbę całkowitą równą maksymalnej możliwej do uzyskania sumie ilorazów inteligencji w zespole, który spełnia oczekiwania Bajtazara.

Drugi wiersz powinien zawierać jedną liczbę całkowitą, będącą liczbą studentów matematyki, których powinien wybrać Bajtazar. W trzecim wierszu należy wypisać numery tych studentów (w dowolnej kolejności). Jeżeli w zespole nie ma żadnych studentów, należy wypisać pusty wiersz.

W kolejnych dwóch wierszach należy podać numery studentów informatyki wyznaczonych do zespołu, w analogicznym formacie.

W przypadku, gdy istnieje wiele rozwiązań, Twój program może wypisać dowolne z nich.

Przykład

Dla danych wejściowych:	poprawnym wynikiem jest:
3 2 3 1 1 2 1 2 2 1 3 1 1 2	6 1 2 2 1 2

Jaskinia

JAS

Dostępna pamięć: 256 MB.

Bajtazar odkrył jaskinię. Okazało się, że jaskinia ta składa się z n komnat połączonych korytarzami w taki sposób, że między dowolnymi dwiema komnatami można przejść na dokładnie jeden sposób.

Jaskinię trzeba teraz staranniej zbadać, dlatego Bajtazar poprosił swoich kolegów o pomoc. Wszyscy przybyli na miejsce i chcą podzielić się na grupy. Każdej grupie przypadnie do zbadania tyle samo komnat, a każda komnata zostanie przydzielona dokładnie jednej grupie. Dodatkowo, żeby ekipy nie wchodziły sobie w drogę, każda z nich powinna być w stanie poruszać się pomiędzy przydzielonymi sobie komnatami bez przechodzenia przez komnaty przydzielone innym grupom.

Na ile grup mogą podzielić się badacze jaskini?

Wejście

Pierwszy wiersz wejścia zawiera jedną liczbę całkowitą n ($2 \leqslant n \leqslant 3\,000\,000$), oznaczającą liczbę komnat w jaskini. Komnaty są ponumerowane od 1 do n.

Kolejne $n-1$ wierszy opisuje połączenia między komnatami. W i-tym spośród nich znajduje się liczba a_i ($1 \leqslant a_i \leqslant i$), która reprezentuje korytarz łączący komnaty o numerach $i+1$ oraz a_i.

Wyjście

Wypisz jeden wiersz zawierający wszystkie takie liczby k, że komnaty w jaskini można podzielić na k rozłącznych zbiorów równej wielkości, a pomiędzy dowolnymi dwiema komnatami w każdym zbiorze można przejść, korzystając jedynie z komnat z tego zbioru. Liczby należy wypisać w kolejności rosnącej, pooddzielane pojedynczymi odstępami.

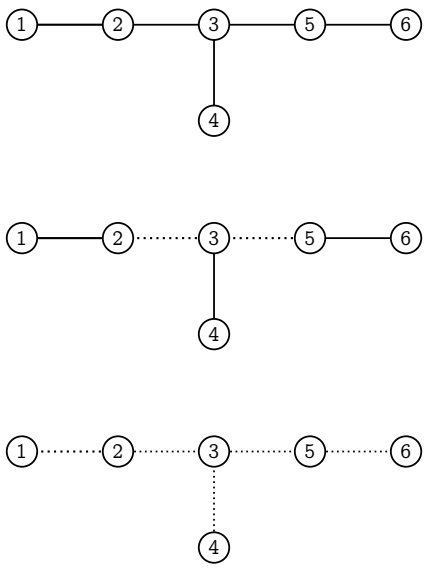
Przykład

Dla danych wejściowych:

```
6
1
2
3
3
5
```

poprawnym wynikiem jest:

```
1 3 6
```



amppz

Krzyżak KRZ

Dostępna pamięć: 32 MB.

Bajtocka odmiana pająka krzyżaka (gatunek *Araneida baitoida*) ma niesamowitą, nawet jak na bajtockie pająki, umiejętność. Potrafi mianowicie w ciągu ułamka sekundy rozpiąć dowolnie dużą pajęczynę, pod warunkiem, że wszystkie włókna składające się na taką sieć będą leżeć w jednej płaszczyźnie. To pozwala krzyżakowi stosować dość oryginalną technikę łowiecką. Nie musi czekać, aż mucha wpadnie w uprzednio zastawioną sieć — wystarczy, że pająk zna aktualną pozycję ofiary, i może natychmiast zbudować pajęczynę tak, by mucha przykleiła się do jednego z jej włókien.

Jeden z takich pająków wypatrzył właśnie w ogrodzie Bajtazara n much. Każda z nich lata nieruchomo w ustalonym punkcie przestrzeni. Pająk zastanawia się, czy będzie w stanie złapać wszystkie muchy w jedną sieć. Napisz program, który rozwieje wątpliwości pająka.

Wejście

W pierwszym wierszu wejścia znajduje się liczba całkowita n ($1 \leqslant n \leqslant 100\,000$). W kolejnych n wierszach znajduje się opis pozycji much w przestrzeni: i-ty z tych wierszy zawiera trzy liczby całkowite x_i, y_i, z_i ($-1\,000\,000 \leqslant x_i, y_i, z_i \leqslant 1\,000\,000$), będące współrzędnymi punktu (w trójwymiarowej przestrzeni euklidesowej), w którym znajduje się i-ta mucha. W dowolnym punkcie może znajdować się co najwyżej jedna mucha.

Wyjście

Twój program powinien wypisać słowo `TAK`, jeśli pająk, rozpinając jedną pajęczynę, jest w stanie złapać wszystkie muchy. W przeciwnym razie Twój program powinien wypisać słowo `NIE`.

Przykład

Dla danych wejściowych:

```
4
0 0 0
-1 0 -100
100 0 231
5 0 15
```

poprawnym wynikiem jest:

```
TAK
```

natomiast dla danych wejściowych:

```
4
0 1 0
-1 0 -100
100 0 231
5 0 15
```

poprawnym wynikiem jest:

```
NIE
```