Rok 2025 (A)

Limit pamięci: 32 MB Limit czasu: 0.50 s

Rok 2025 jest szczególny, bowiem $2025 = 1^3 + 2^3 + 3^3 + 4^3 + 5^3 + 6^3 + 7^3 + 8^3 + 9^3$. Jaś wie, że na kolejny tak szczególny rok będzie musiał poczekać 10^3 lat. To trochę długo. Dlatego Jaś zastanawia się, czy wcześniej nie ma innych lat, które są szczególne, choć być może w nieco innych sposób.

Wejście

W pierwszym (jedynym) wierszu wejścia znajduje się jedna liczba naturalna k.

Wyjście

W pierwszym (jedynym) wierszu wyjścia powinna się znaleźć najmniejsza liczba naturalna R, większa od 2025 i mniejsza od 3025, taka, że R jest sumą k-tych potęg początkowych liczb naturalnych albo słowo `NIE`, gdy taka liczba nie istnieje.

Ograniczenia

$1 \leq k \leq 20$.

Przykład

Wejście	Wyjście	Wyjaśnienie
`4`	`2275`	Dla $k = 4$ odpowiedzią jest 2275, ponieważ $1^4 + 2^4 + 3^4 + 4^4 + 5^4 + 6^4 = 2275$, a $1^4 + 2^4 + 3^4 + 4^4 + 5^4 < 2025$.
`5`	`NIE`	Dla $k = 5$ odpowiedzią jest `NIE`, ponieważ piąte potęgi kolejnych liczb naturalnych, równe kolejno $1, 32, 243, 1024, 3125, \ldots$, nie sumują się do liczby z przedziału $(2025, 3025)$.

Podział na zbiory trójkowiaste (B)

Limit pamięci: 32 MB Limit czasu: 2.00 s

Wielozbiór (czyli zbiór, w którym dopuszczamy powtarzające się elementy) liczb nazwiemy *trójkowiastym*, jeśli suma jego elementów jest podzielna przez 3. Przykładowo, zbiory $\{1, 2, 3\}$ oraz $\{12\}$ są trójkowiaste, a zbiór $\{2, 2, 2, 2\}$ – nie.

Napisz program, który dla danego zbioru N liczb naturalnych $\{A_1, A_2, \ldots, A_n\}$ sprawdzi, czy można go rozbić na K niepustych, rozłącznych podzbiorów, z których każdy jest trójkowiasty.

Wejście

W pierwszym wierszu znajduje się liczba naturalna T, oznaczająca liczbę testów. W kolejnych $2T$ wierszach znajdują się dane dla kolejnych testów.

Dane dla każdego testu zapisane są w dwóch wierszach:

- W pierwszym z nich znajdują się liczby naturalne N i K.
- W drugim z nich znajduje się N liczb naturalnych $A_1, A_2, \ldots, A_n$.

Wyjście

W i-tym wierszu należy wypisać wynik dla i-tego testu, którym jest jedno słowo:

- `TAK` – jeśli zbiór $\{A_1, A_2, \ldots, A_n\}$ można rozbić na K niepustych, rozłącznych podzbiorów trójkowiastych,
- `NIE` – w przeciwnym przypadku.

Ograniczenia

$1 \leq T \leq 15$, $1 \leq N, K \leq 1\,000\,000$, $1 \leq A_i \leq 10^{18}$.

Przykład

Wejście

```
2
3 3
1 2 3
10 4
1 2 7 23 1 7 8 61 32 8
```

Wyjście

```
NIE
TAK
```

Wyjaśnienie

Dla $N = 3$, $K = 3$ oraz zbioru $\{1, 2, 3\}$ odpowiedzią jest `NIE`, ponieważ z tych liczb można utworzyć tylko dwa niepuste zbiory trójkowiaste.

Dla $N = 10$, $K = 4$ oraz zbioru $\{1, 2, 7, 23, 1, 7, 8, 61, 32, 8\}$ odpowiedzią jest `TAK`. Przykładowe rozbicie na zbiory trójkowiaste:
$\{7, 32\}, \{61, 2\}, \{1, 7, 1\}, \{8, 23, 8\}$.

Ciąg coraz szybciej rosnący (C)

Limit pamięci: 32 MB Limit czasu: 0.50 s

Ciąg $A_1, A_2, \ldots, A_n$ (dla $N \geq 3$) nazywamy *coraz szybciej rosnącym*, jeśli dla każdego $i = 1, 2, \ldots, N-2$ zachodzi nierówność $A_{i+1} - A_i < A_{i+2} - A_{i+1}$.

Napisz program, który dla liczb naturalnych A, B, C takich, że $A < B < C$, sprawdza czy mogą one być elementami (niekoniecznie sąsiednimi) jakiegoś ciągu coraz szybciej rosnącego.

Wejście

W pierwszym wierszu znajduje się liczba naturalna T, oznaczająca liczbę testów. W kolejnych T wierszach znajdują się dane dla kolejnych testów.

Dane dla każdego testu zapisane są w jednym wierszu, zawierającym trzy liczby naturalne A, B, C, poodzielane pojedynczymi odstępami.

Wyjście

W i-tym wierszu należy wypisać wynik dla i-tego testu, którym jest jedno słowo: `TAK` – jeśli liczby A, B, C mogą być elementami jakiegoś ciągu coraz szybciej rosnącego oraz `NIE` w przeciwnym przypadku.

Ograniczenia

$1 \leq T \leq 100\,000$, $1 \leq A < B < C \leq 10^{18}$.

Przykład

Wejście	Wyjście	Wyjaśnienie
4 1 2 4 1 3 5 1 4 6 1 9 14	TAK NIE NIE TAK	W pierwszym przypadku $(1, 2, 4)$ jest ciągiem coraz szybciej rosnącym. W ostatnim przypadku liczby $1, 9, 14$ są elementami ciągu $(1, 2, 5, 9, 14)$, który jest coraz szybciej rosnący.

Podzielność przez 7 (D)

Limit pamięci: 32 MB Limit czasu: 0.50 s

Jaś fascynuje się liczbami. Nic więc dziwnego, że na imieniny w prezencie otrzymał od rodziców piękną, bardzo wielką liczbę S. Rodzice nie wiedzieli jednak, że Jaś najbardziej lubi liczby szczęśliwe, to jest takie, które są podzielne przez 7.

Jaś zaraz sprawdzi, czy S jest liczbą szczęśliwą. A jeśli nie, to czy po małej zmianie może taką się stać. Przez małą zmianę Jaś rozumie zmianę dokładnie jednej cyfry o 1, czyli zwiększenie cyfry o 1 lub zmniejszenie jej o 1. Przy tym zwiększenie cyfry 9 o jeden oznacza zastąpienie jej przez 0, a zmniejszenie cyfry 0 o jeden oznacza zastąpienie jej przez 9 (inne cyfry nie ulegają zmianie przy takiej operacji).

Napisz program, obliczający na ile sposobów Jaś może dokonać małej zmiany w liczbie S, tak by stała się ona liczbą szczęśliwą.

Wejście

W pierwszym wierszu znajduje się liczba naturalna T, oznaczająca liczbę testów. W każdym z kolejnych T wierszy znajduje się jedna liczba naturalna S.

Wyjście

W i-tym wierszu należy wypisać jedną liczbę naturalną – wynik dla i-tego testu (liczbę sposobów dokonania małej zmiany, żeby liczba stała się szczęśliwa).

Ograniczenia

$1 \leq T \leq 1000$.

Każda liczba S ma nie więcej niż $10\,000$ cyfr dziesiętnych.

Uwaga

Jaś może zmienić najbardziej znaczącą cyfrę liczby S na zero, np. zamieniając 9 w liczbie `90` otrzyma `00`, co jest zapisem liczby zero (a więc liczby szczęśliwej). Jaś nie może zaś dokładać żadnych cyfr na początku liczby (udając, że zamienia zero wiodące na jedynkę lub dziewiątkę).

Przykład

Wejście	Wyjście
10 1 2 3 4 5 6 7 8 9 10	1 0 0 0 0 1 0 1 1 1

Wejście	Wyjście
3 2344 1902 459751	2 0 3

Podsłowo z dwóch liter (E)

Limit pamięci: 32 MB Limit czasu: 0.50 s

Jaś miał zapisane na kartce bardzo długie słowo S, którego każda literka była jedną z literek `a`, `b` lub `c`. Niestety porozcinał je i w ten sposób otrzymał karteczki z krótszymi słowami. Teraz chce odtworzyć słowo S, ale nie jest w stanie, ponieważ prawie go nie pamięta. Pamięta jedynie to, że S zawierało bardzo długie spójne podsłowo, w które nie zawierało co najmniej jednej z literek `a`, `b` lub `c`. To za mało informacji by odtworzyć S, ale Jaś postanowił, że przynajmniej utworzy z tych krótszych słów słowo, zawierające maksymalnie długie podsłowo (nazwijmy je X) nie zawierające literki `a` lub nie zawierające literki `b` lub nie zawierające literki `c`. Pomóż Jasiowi i napisz program, który obliczy jaką największą długość może mieć podsłowo X.

Wejście

W pierwszym wierszu znajduje się liczba naturalna N, oznaczająca liczbę podsłów powstałych ze słowa S, po rozcięciu go przez Jasia. W drugim wierszu znajduje się N słów $S_1, S_2, \ldots, S_N$ – każde z nich zawiera jedynie litery `a`, `b`, `c`.

Wyjście

W jedynym wierszu należy wypisać jedną liczbę naturalną, która jest równa maksymalnej długości podsłowa X, opisanego w treści zadania.

Ograniczenia

$1 \leq N \leq 1000$, suma długości słów S_i nie przekracza $10\,000$.

Przykład

Wejście	Wyjście	Wyjaśnienie
2 `ccbaa aaaccbcb`	8	Jaś może z tych słów utworzyć słowo `ccbaaaaaccbcb` lub `aaaccbcbccbaa`. W pierwszym z nich najdłuższym podsłowem X jest `aaaaacc` o długości 7 (nie zawiera literki `b`), w drugim – najdłuższym podsłowem X jest `ccbcbccb` o długości 8 (nie zawiera `a`).

Wejście	Wyjście
3 `bb bbbbbb bbb`	11

Pieniążki (F)

Limit pamięci: 32 MB Limit czasu: 0.50 s

Jaś ma bardzo dużo monet. Dokładniej są to monety o nominałach A oraz B bajtalarów. Możemy dla uproszczenia założyć, że pieniążków jest tak dużo, że Jaś ma ich nieskończenie wiele. Mogłoby się wydawać, że nieskończenie bogaty Jaś powinien być zadowolony... Niestety Jaś zawsze widzi dziurę w całym. Zauważył, że są takie kwoty, których nie da się wydać za pomocą jego pieniążków. Napisz program, który obliczy sumę tych kwot.

Wejście

W pierwszym (jedynym) wierszu wejścia znajdują się dwie liczby naturalne A oraz B, oddzielone pojedynczym odstępem.

Wyjście

W pierwszym (jedynym) wierszu wyjścia powinna się znaleźć suma dodatnich całkowitych kwot, które nie są możliwe do uzyskania za pomocą pewnej liczby monet o nominałach A i B. Jeżeli ta suma jest nieskończona, zamiast tego należy wypisać `infinity`.

Ograniczenia

$1 \leq A, B \leq 100\,000$.

Przykład

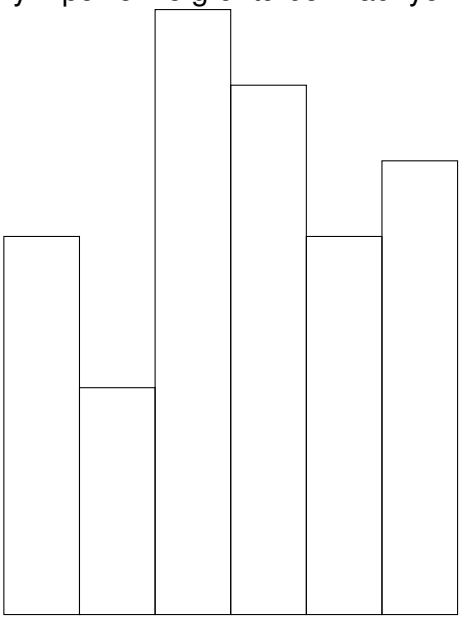
Wejście	Wyjście	Wyjaśnienie
`3 5`	`14`	Jaś nie jest w stanie wydać kwot 1, 2, 4 oraz 7. Ich suma to $1 + 2 + 4 + 7 = 14$.

Wejście	Wyjście	Wyjaśnienie
`4 6`	`infinity`	Jaś nie jest w stanie wydać żadnej nieparzystej kwoty.

Powódź (G)

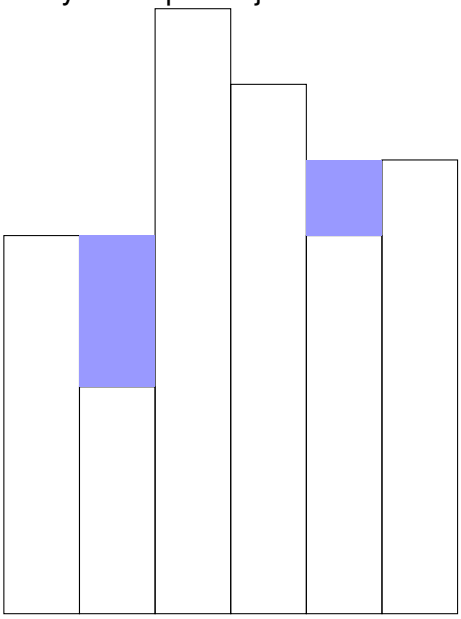
Limit pamięci: 32 MB Limit czasu: 0.50 s

W dwuwymiarowej Bajtocji przy głównej ulicy znajduje się N wieżowców o ustalonych (być może różnych) wysokościach. Każdy wieżowiec ma szerokość jednego metra, wszystkie wieżowce stoją obok siebie, na tym samym poziomie gruntu bez żadnych przerw między nimi. Sytuacja wygląda więc jak na poniższym obrazku:



Wyobraźmy sobie, że dwuwymiarową Bajtocję nawiedza powódź. Nieskończona, dwuwymiarowa chmura deszczu leje się z góry na budynki. Jakie będzie pole deszczu, który utrzyma się między budynkami? Woda przelewa się w lewo lub prawo, gdy osiągnie poziom budynku. Woda nie utrzymuje się na budynku, jeżeli może z niego spaść z lewej lub z prawej strony.

Na rysunku poniżej niebieskim kolorem zaznaczono utrzymującą się wodę (o powierzchni 3 jednostek).



Napisz program, który wczyta wysokości budynków w dwuwymiarowej Bajtocji, wyznaczy pole wody, która utrzyma się między budynkami po nieskończonym deszczu i wypisze wynik na standardowe wyjście.

Wejście

W pierwszym wierszu wejścia znajduje się jedna liczba naturalna N, określająca liczbę budynków. W drugim (ostatnim) wierszu wejścia znajduje się ciąg N liczb naturalnych A_i, pooddzielanych pojedynczymi odstępami. Są to wysokości kolejnych budynków stojących od lewej do prawej przy ulicy.

Wyjście

W pierwszym (jedynym) wierszu wyjścia powinna się znaleźć jedna liczba całkowita – pole wody, która utrzyma się między budynkami.

Ograniczenia

$1 \leq N \leq 500\,000$, $1 \leq A_i \leq 10^9$.

Przykład

Wejście	Wyjście	Wyjaśnienie
6 5 3 8 7 5 6	3	Ten test przykładowy odpowiada rysunkowi powyżej w treści.