Adam Karczmarz
Treść zadania, Opracowanie

Piotr Smulewicz, Bartosz Tarnawski
Program

Dostępna pamięć: 128 MB. *OI, etap II, dzień pierwszy, 11.02.2015*

Pustynia

Droga z Bajtadu do Bajtary wiedzie przez piaski Wielkiej Pustyni Bajtockiej. Jest to męcząca wędrówka, zwłaszcza że na całej trasie znajduje się tylko s studni. Widząc, że gospodarka Bajtocji zależy w dużej mierze od dostępności szlaków komunikacyjnych, władca Bajtocji postanowił wykopać nowe studnie na tej trasie. Odległość z Bajtadu do Bajtary wynosi $n + 1$ bajtomil i w każdym punkcie w odległości całkowitej liczby bajtomil od Bajtadu znajduje się lub może znajdować się studnia. Im głębiej jest położona woda w danym miejscu, tym trudniejsze i bardziej kosztowne jest wykopanie w tym miejscu nowej studni.

Władca zlecił zatem zbadanie sytuacji nadwornemu geologowi Bajtazarowi. Bajtazar dysponuje m pomiarami wykonanymi za pomocą sieci satelitarnej. Niestety, informacje dostarczone przez satelity nie dają wprost informacji na temat głębokości wody. Każdy pomiar wykonany jest na spójnym fragmencie trasy i wskazuje jedynie, że w pewnych punktach na tym fragmencie woda znajduje się głębiej niż w pozostałych. Dodatkowo wiadomo, że woda w każdym punkcie leży na całkowitej głębokości od 1 do 10^9 bajtometrów.

Pomóż Bajtazarowi i wyznacz, jak może wyglądać rzeczywista głębokość wody w każdym punkcie trasy. Może się okazać, że dane satelitarne są sprzeczne.

Wejście

Pierwszy wiersz standardowego wejścia zawiera trzy liczby całkowite n, s i m ($1 \leqslant s \leqslant n \leqslant 100\,000$, $1 \leqslant m \leqslant 200\,000$) pooddzielane pojedynczymi odstępami, opisujące odległość między miastami, liczbę studni na trasie oraz liczbę pomiarów satelitarnych.

Kolejne s wierszy opisuje studnie: i-ty z nich zawiera dwie liczby całkowite p_i i d_i ($1 \leqslant p_i \leqslant n$, $1 \leqslant d_i \leqslant 1\,000\,000\,000$), oznaczające, że i-ta studnia znajduje się w odległości p_i bajtomil od Bajtadu i ma głębokość d_i bajtometrów (tzn. w punkcie, w którym znajduje się studnia, woda jest na głębokości d_i bajtometrów). Studnie podane są w kolejności rosnących wartości p_i.

Kolejne m wierszy opisuje wykonane pomiary satelitarne: i-ty z nich zawiera trzy liczby całkowite l_i, r_i i k_i ($1 \leqslant l_i < r_i \leqslant n$, $1 \leqslant k_i \leqslant r_i - l_i$), po których następuje ciąg k_i liczb całkowitych $x_1, x_2, \ldots, x_{k_i}$ ($l_i \leqslant x_1 < x_2 < \ldots < x_{k_i} \leqslant r_i$). Oznacza to pomiar na odcinku od l_i do r_i (włącznie), w wyniku którego ustalono, że woda w punktach $x_1, \ldots, x_{k_i}$ znajduje się **ściśle głębiej** *niż woda w pozostałych punktach z tego odcinka. Suma wszystkich wartości k_i nie przekracza 200 000.*

W testach wartych łącznie 60% punktów zachodzą dodatkowe warunki $n, m \leqslant 1000$. W testach wartych łącznie 30% punktów zachodzi dodatkowy warunek, że suma wszystkich wartości k_i nie przekracza 1000.

Wyjście

Jeśli nie istnieje układ głębokości zgodny z wykonanymi pomiarami, pierwszy wiersz standardowego wyjścia powinien zawierać jedno słowo `NIE`*. W przeciwnym wypadku w pierwszym wierszu*

wyjścia powinno znaleźć się słowo `TAK`*, natomiast drugi wiersz powinien zawierać ciąg n liczb całkowitych z przedziału od 1 do 1 000 000 000 oznaczający głębokości wody w kolejnych punktach na trasie (idąc od Bajtadu). Jeśli istnieje więcej niż jedno poprawne rozwiązanie, Twój program powinien wypisać dowolne z nich.*

Przykład

Dla danych wejściowych:	*jednym z poprawnych wyników jest:*

`5 2 2`
`2 7`
`5 3`
`1 4 2 2 3`
`4 5 1 4`
	`TAK`
	`6 7 1000000000 6 3`

Dla danych wejściowych:	*poprawnym wynikiem jest:*

`3 2 1`
`2 3`
`3 5`
`1 3 1 2`
	`NIE`

Również dla danych wejściowych:	*poprawnym wynikiem jest:*

`2 1 1`
`1 1000000000`
`1 2 1 2`
	`NIE`

Testy „ocen":

`1ocen`: *n = 100 000, pomiary wskazują, że woda w punkcie i jest głębiej niż we wszystkich wcześniejszych punktach trasy (dla* $i = 2, \ldots, n$*);*

`2ocen`: *n = 100 000, z jednego pomiaru wynika, że woda w punktach o numerach parzystych jest głębiej niż w punktach o numerach nieparzystych.*

Rozwiązanie

Naszym zadaniem jest konstrukcja pewnego ciągu liczb całkowitych $a_1, \ldots, a_n$ (czasami w skrócie będziemy go nazywać a). Liczbę a_i definiujemy jako głębokość studni położonej w odległości i bajtomil od Bajtadu. Szukany ciąg musi spełniać następujące warunki:

(1) $1 \leqslant a_i \leqslant 10^9$ dla $i = 1, \ldots, n$,

(2) dla $i \in \mathcal{F}_0$, gdzie $\mathcal{F}_0 \subseteq \{1, \ldots, n\}$, a_i jest ustalone i dane na wejściu,

(3) dla każdego spośród m danych na wejściu ograniczeń postaci $(l, r, x_1, \ldots, x_k)$, gdzie $l \leqslant x_1 < \ldots < x_k \leqslant r$, każda z liczb $a_{x_1}, \ldots, a_{x_k}$ jest ostro większa niż dowolne a_j, takie że $j \in \{l, \ldots, r\} \setminus \{x_1, \ldots, x_k\}$.

Może się także okazać, że podane warunki są sprzeczne – nasz program ma wtedy wypisać na wyjście słowo `NIE`.

Reprezentacja ciągu i ograniczeń

Naszą strategią będzie iteracyjne upraszczanie ograniczeń nałożonych na nasz ciąg, aż dotrzemy do ograniczeń, które jednocześnie są trywialne do sprawdzenia i pozwalają łatwo znaleźć przykładowe rozwiązanie. Mówiąc ściślej, będziemy utrzymywać:

- ograniczenia górne $c_1, \ldots, c_n$, oznaczające, że szukany ciąg powinien spełniać $1 \leqslant a_i \leqslant c_i$ dla $i = 1, \ldots, n$. Początkowo ustawiamy $c_i := 10^9$.
- Zbiór $\mathcal{F}$ zawierający indeksy $i \in \{1, \ldots, n\}$, dla których a_i jest już określone. Początkowo mamy $\mathcal{F} = \mathcal{F}_0$.
- Ważony graf skierowany $G = (V, E)$ o wierzchołkach $V = \{v_1, \ldots, v_n\}$, reprezentujący nierówności pomiędzy elementami ciągu a. Krawędź od v_i do v_j o wadze d w takim grafie oznacza, że $a_i \geqslant a_j + d$.

Początkowy graf G konstruujemy na podstawie danych ograniczeń typu (3). Dla każdego ograniczenia postaci $(l, r, x_1, \ldots, x_k)$ i każdej pary (a_i, a_j) takiej, że $i \in \{x_1, \ldots, x_k\}$ i $j \in [l, r] \setminus \{x_1, \ldots, x_k\}$, tworzymy krawędź od v_i do v_j o wadze 1.

Upraszczanie ograniczeń

Dopóki będzie to możliwe, będziemy stosować jedną z dwóch reguł upraszczających ograniczenia. Intuicyjnie, chcemy zastępować ograniczenia wynikające z grafu G takimi, które wyrażone są za pomocą ograniczeń górnych c_i i zbioru $\mathcal{F}$. Każda z tych reguł ma następujące własności.

(1) Jeżeli istnieje ciąg spełniający ograniczenia przed zastosowaniem reguły, to istnieje też ciąg spełniający ograniczenia uproszczone.

(2) Każdy ciąg spełniający ograniczenia po zastosowaniu reguły spełnia również ograniczenia przed zastosowaniem reguły.

Reguła 1. Jeśli istnieje takie i, że $i \notin \mathcal{F}$ i w G nie ma krawędzi wchodzącej do v_i, to dodajemy i do $\mathcal{F}$ i ustawiamy $a_i := c_i$.

Musimy pokazać, że jeśli istnieje ciąg a spełniający ograniczenia przed zastosowaniem reguły, to istnieje też ciąg spełniający uproszczone ograniczenia. Istotnie, jako że a spełnia ograniczenia przed zastosowaniem reguły, to $1 \leqslant a_i \leqslant c_i$. Ponieważ do v_i nie wchodzi w G żadna krawędź, nie mamy ograniczeń typu $a_j \geqslant a_i + d$. Być może istnieją ograniczenia postaci $a_i \geqslant a_j + d$, ale każde z nich jest tym bardziej spełnione, jeśli wartość a_i zwiększymy do wartości c_i. To pokazuje własność (1) dla tej reguły.

Zauważmy, że ograniczenia przed zastosowaniem reguły są podzbiorem ograniczeń po jej zastosowaniu. Stąd łatwo wynika własność (2).

Reguła 2. Jeśli istnieje takie i, że $i \in \mathcal{F}$ i istnieje krawędź $(v_i, v_j) \in E$ o wadze d, to przypisujemy $c_j := \min(c_j, a_i - d)$. Ponadto, usuwamy z E krawędź (v_i, v_j).

Dla dowolnego ciągu a, ograniczenia $a_j \leqslant c_j$ i $a_i \geqslant a_j + d$ są równoważne ograniczeniu $a_j \leqslant \min(c_j, a_i - d)$. To dowodzi jednocześnie obu własności (1) i (2).

Zastanówmy się teraz, w jakiej sytuacji nie jesteśmy w stanie zastosować żadnej z reguł upraszczających. Dzieje się tak dokładnie wtedy, gdy do każdego v_i, $i \notin \mathcal{F}$, wchodzi co najmniej jedna krawędź (w przeciwnym razie można by zastosować regułę 1), a dla każdego $j \in \mathcal{F}$, z v_j nie wychodzi w G żadna krawędź (w przeciwnym razie można by zastosować regułę 2). Każda krawędź w grafie G musi mieć zatem początek w v_j takim, że $j \notin \mathcal{F}$.

Może się zdarzyć, że $\mathcal{F} = \{1, \ldots, n\}$. Wtedy w grafie G nie ma ani jednej krawędzi. Własności (1) i (2) stosowanych reguł gwarantują, że jeżeli $1 \leqslant a_i \leqslant c_i$ dla każdego i, to $a_1, \ldots, a_n$ spełnia oryginalne ograniczenia. W przeciwnym wypadku, na mocy własności (1), mamy pewność, że rozwiązanie nie istnieje.

Jeśli natomiast $\mathcal{F} \neq \{1, \ldots, n\}$, to po usunięciu z grafu wierzchołków $\{v_i : i \in \mathcal{F}\}$ dostajemy niepusty graf skierowany, w którym do każdego wierzchołka wchodzi przynajmniej jedna krawędź. Łatwo pokazać, że w takim grafie musi istnieć cykl, na przykład $w_1 \to \ldots \to w_p \to w_1$, gdzie $p > 1$. Ponieważ każda krawędź w G ma wagę 1, cykl taki odpowiada p nierównościom: $w_i \geqslant w_{i+1} + 1$ dla każdego $i = 1, 2, \ldots, p-1$ i $w_p \geqslant w_1 + 1$. Dodając stronami wszystkie te nierówności, otrzymujemy $w_1 + \ldots + w_p \geqslant w_1 + \ldots + w_p + p$, nierówność w oczywisty sposób nieprawdziwą. Zatem na mocy własności (1), w tym przypadku ograniczenia są sprzeczne.

Implementacja

Aby efektywnie zaimplementować stosowanie reguł, oprócz ograniczeń c_i, zbioru $\mathcal{F}$ i grafu G, utrzymujemy kolejki Q_j, dla $j = 1, 2$, zawierające indeksy i, dla których można zastosować regułę j. Po zastosowaniu reguły 1, usuwamy indeks i z kolejki Q_1 i wstawiamy go do Q_2, o ile z v_i wychodzi aktualnie co najmniej jedna krawędź. Po zastosowaniu reguły 2, indeks i usuwamy z kolejki Q_2 tylko wtedy, gdy krawędź (v_i, v_j) była jedyną krawędzią wychodzącą z v_i. Dodatkowo jeśli w wyniku usunięcia tej krawędzi v_j staje się wierzchołkiem, do którego nie wchodzi żadna krawędź, wstawiamy j do Q_1, jeśli $j \notin \mathcal{F}$.

Każdy indeks jest wstawiany co najwyżej raz do Q_1 i co najwyżej raz do Q_2. Dodatkowo, reguła 2 jest stosowana dla każdego indeksu i w kolejce Q_2 co najwyżej tyle razy, ile początkowo krawędzi wychodzi z v_i. Stąd, na kolejce Q_1 wykonujemy $O(n) = O(|V|)$ operacji, a na kolejce Q_2 wykonujemy $O(|E|)$ operacji. Nasz algorytm działa zatem w czasie $O(|V| + |E|)$.

Aby ostatecznie obliczyć złożoność tego rozwiązania, musimy oszacować liczbę krawędzi w grafie G przed pierwszym uproszczeniem ograniczeń. Każde wejściowe ograniczenie postaci $(l, r, x_1, \ldots, x_k)$ tworzy $(r-l+1-k) \cdot k \leqslant nk$ krawędzi. Oznaczmy przez S sumę liczb k we wszystkich ograniczeniach. Wówczas złożoność czasowa tego algorytmu szacuje się przez $O(n \cdot S)$. Zauważmy, że w grafie mogą znajdować się krawędzie wielokrotne. Takie rozwiązanie było oceniane na zawodach na ok. 30% punktów. Przykładowa implementacja znajduje się w pliku `puss5.cpp`.

Zmniejszanie rozmiaru grafu

Rozważmy g-te ograniczenie postaci $(l, r, x_1, \ldots, x_k)$. Zauważmy, że dla każdego ciągu a spełniającego to ograniczenie, istnieje liczba a^*_g taka, że $a_{x_i} \geqslant a^*_g + 1$ dla $i = 1, \ldots, k$ i jednocześnie $a^*_g \geqslant a_j$ dla każdego $j \in [l, r] \setminus \{x_1, \ldots, x_k\}$. Nic nie stoi zatem na przeszkodzie, żeby rozszerzyć poszukiwany ciąg $a_1, \ldots, a_n$ o dodatkowe m pomocniczych liczb $a^*_1, \ldots, a^*_m$, które nasz algorytm będzie próbował wyznaczyć, a następnie zignoruje je przy wypisywaniu wyniku.

Przypomnijmy, że algorytm upraszczający ograniczenia działał w czasie proporcjonalnym do rozmiaru grafu G. Dodatkowe elementy ciągu pozwolą nam na zmniejszenie rozmiaru początkowego grafu G, ale nie wpłyną na sam algorytm upraszczający.

Nasz nowy graf G ma większy zbiór wierzchołków $V = \{v_1, \ldots, v_n, v^*_1, \ldots, v^*_m\}$. Podobnie jak poprzednio, ważone i skierowane krawędzie pomiędzy wierzchołkami odpowiadają nierównościom pomiędzy elementami $a_1, \ldots, a_n, a^*_1, \ldots, a^*_m$. Aby zakodować g-te ograniczenie postaci $(l, r, x_1, \ldots, x_k)$, do grafu dodajemy krawędzie (v_{x_i}, v^*_g) dla $i = 1, \ldots, k$, każdą o wadze 1. Dodatkowo, dla każdego $j \in [l, r] \setminus \{x_1, \ldots, x_k\}$, dodajemy do grafu krawędź (v^*_g, v_j) o wadze 0.

Inaczej niż poprzednio, tym razem w grafie G mamy dwie możliwe wagi na krawędziach – 0 i 1. Aby zastosować algorytm upraszczający ograniczenia, należy się upewnić, że dowolny cykl w takim grafie wciąż prowadzi do sprzeczności. Tak istotnie jest: w dowolnym cyklu w G na zmianę występują wierzchołki odpowiadające liczbom a_i i te odpowiadające liczbom a^*_j. Stąd, co druga krawędź na cyklu ma wagę 1, a więc każdy cykl ma dodatnią wagę. Na mocy analogicznego jak uprzednio argumentu otrzymujemy sprzeczność.

Aby oszacować złożoność tego rozwiązania, musimy obliczyć rozmiar grafu G. Mamy $n + m$ wierzchołków. Dla każdego z m ograniczeń typu $(l, r, x_1, \ldots, x_k)$ tworzymy $k + (r - l + 1 - k) = r - l + 1 \leqslant n$ krawędzi. Stąd, sumaryczna liczba krawędzi szacuje się przez nm i złożonością czasową całego rozwiązania jest $O(nm)$. Takie rozwiązanie otrzymywało na zawodach około 60% punktów. Program `puss4.cpp` realizuje ten pomysł.

Rozwiązanie wzorcowe

W rozwiązaniu wzorcowym postępujemy podobnie jak poprzednio: zmniejszamy rozmiar grafu G, wprowadzając pomocnicze wyrazy ciągu, które, odpowiednio zdefiniowane, pozwolą nam na zmniejszenie liczby krawędzi potrzebnych do zakodowania wszystkich ograniczeń.

W poprzednim rozwiązaniu, rozważając g-te ograniczenie $(l, r, x_1, \ldots, x_k)$, do G dodawaliśmy $r - l + 1 - k$ ograniczeń postaci (v^*_g, v_j). Liczba $r - l + 1$ mogła być jednak duża, rzędu $\Theta(n)$. Spróbujmy uporać się z tym problemem. Zauważmy, że zbiór $[l, r] \setminus \{x_1, \ldots, x_k\}$ jest tak naprawdę sumą $k + 1$ (być może pustych) przedziałów: $[l, x_1 - 1], [x_1 + 1, x_2 - 1], \ldots, [x_{k-1} + 1, x_k - 1], [x_k + 1, r]$. Gdybyśmy dla każdego przedziału $[x, y]$ (gdzie $1 \leqslant x \leqslant y \leqslant n$) mieli dodatkowy element ciągu $a_{[x,y]}$ taki, że $a_{[x,y]} \geqslant a_z$ dla każdego $z = x, \ldots, y$, to dla g-tego ograniczenia wystarczyłoby nam tylko k dodatkowych krawędzi w G. Niestety, wprowadzenie elementów $a_{[x,y]}$ dla każdego możliwego przedziału $[x, y]$ wiązałoby się z dodatkowymi krawędziami w grafie:

moglibyśmy na przykład zakodować za pomocą krawędzi ograniczenia $a_{[x,y]} \geqslant a_z$ dla każdych $1 \leqslant x \leqslant z \leqslant y \leqslant n$, otrzymując w ten sposób $O(n^3)$ dodatkowych krawędzi. Moglibyśmy również postąpić odrobinę sprytniej i brać pod uwagę tylko ograniczenia postaci $a_{[x,y]} \geqslant a_{[x+1,y]}$ i $a_{[x,y]} \geqslant a_{[x,y-1]}$ dla każdych $x < y$, generując w ten sposób tylko $O(n^2)$ nowych krawędzi. Ponieważ n może być dość duże, takie rozwiązanie nie jest jednak wystarczające do uzyskania kompletu punktów.

Nie rezygnujemy jednak z pomysłu użycia elementów ciągu ograniczających z góry pewne spójne grupy elementów oryginalnego ciągu. Skorzystamy z pomysłu powtarzającego się wielokrotnie w zadaniach olimpijskich: użyjemy tak zwanych *przedziałów bazowych*. Przypomnijmy pokrótce, czym są przedziały bazowe. Dla liczby naturalnej B, zbiór P_B przedziałów bazowych definiujemy jako najmniejszy zbiór taki, że:

- przedział $[1, 2^B]$ należy do P_B,
- jeśli przedział $[a, b]$ należy do P_B i $a < b$, to przedziały $[a, s]$ i $[s + 1, b]$, gdzie $s = \lfloor \frac{a+b}{2} \rfloor$, także należą do P_B.

Przykładowo, jeśli $B = 2$, to do zbioru przedziałów bazowych P_B należą przedziały $[1, 4], [1, 2], [3, 4], [1, 1], [2, 2], [3, 3]$ i $[4, 4]$.

Do zbioru P_B należy dokładnie $2^{B+1} - 1$ przedziałów bazowych. Każdy przedział $[x, y]$, gdzie $x \leqslant y$ i $x, y \in \{1, \ldots, 2^B\}$, można rozbić na $O(B)$ parami rozłącznych przedziałów bazowych, które pokrywają wszystkie elementy całkowite zawarte w $[x, y]$. Co więcej, rozbicie to można obliczyć w czasie $O(B)$. Przedziały bazowe łączy się z konstrukcją drzewa przedziałowego, o którym można przeczytać np. w opracowaniu zadania *Logistyka* w tej książeczce i w zawartych tam odnośnikach.

Powróćmy teraz do naszego rozwiązania. Niech B będzie najmniejszą liczbą naturalną taką, że $n \leqslant 2^B$. Mamy $n > 2^{B-1}$, a więc zbiór przedziałów bazowych P_B zawiera $O(n)$ przedziałów. Wiemy też, że każdy przedział $[x, y]$, gdzie $1 \leqslant x \leqslant y \leqslant n$, można rozbić na $O(\log n)$ przedziałów z P_B. Dodajemy do konstruowanego ciągu $a_1, \ldots, a_n, a_1^*, \ldots, a_m^*$ jeszcze co najwyżej $2^B - 1 = O(n)$ elementów: dla każdego przedziału bazowego $[p, q] \subseteq [1, n]$, gdzie $p < q$, mamy element $a_{[p,q]}$ taki, że $a_{[p,q]} \geqslant a_z$ dla każdego $z \in [p, q]$. Będziemy też używali zapisu $a_{[p,p]}$ do oznaczenia elementu a_p. Dla każdego dodanego elementu mamy także w grafie G odpowiadający mu wierzchołek $v_{[p,q]}$. Aby zakodować za pomocą grafu ograniczenia postaci $a_{[p,q]} \geqslant a_z$ dla każdego $z \in [p, q]$, dodajemy do G krawędzie $(v_{[p,q]}, v_{[p,s]})$ i $(v_{[p,q]}, v_{[s+1,q]})$, gdzie $s = \lfloor \frac{p+q}{2} \rfloor$ – obie o wadze 0. Definicja przedziałów bazowych gwarantuje, że $[p, s]$ i $[s + 1, q]$ także należą do P_B i zawierają się w $[1, n]$. Przez indukcję po długości przedziału bazowego można łatwo udowodnić, że dla $z \in [p, q]$ istnieje w grafie G skierowana ścieżka od $v_{[p,q]}$ do v_z o wadze 0. Stąd, ograniczenia zakodowane w grafie implikują, że $a_{[p,q]} \geqslant a_z$. W ten sposób dodajemy do grafu co najwyżej $2|P_B| = O(n)$ wierzchołków i $O(n)$ krawędzi.

Pozostaje ustalić, jak zakodować g-te ograniczenie $(l, r, x_1, \ldots, x_k)$ za pomocą krawędzi grafu. Podobnie jak poprzednio, mamy w G krawędzie (v_{x_i}, v_g^*) o wadze 1. Każdy niepusty przedział I spośród $[l, x_1 - 1], [x_1 + 1, x_2 - 1], \ldots, [x_{k-1} + 1, x_k - 1], [x_k + 1, r]$ rozbijamy na $O(\log n)$ przedziałów bazowych $I'_1, I'_2, \ldots, I'_h$, a następnie dla każdego $j = 1, \ldots, h$ dodajemy do grafu krawędź $(v_g^*, v_{I'_j})$ o wadze 0. Ponieważ $I = I'_1 \cup \ldots \cup I'_h$, krawędzie te będą implikować nierówność $a_g^* \geqslant a_j$ dla każdego $j \in [l, r] \setminus \{x_1, \ldots, x_k\}$. Ostatecznie, w związku z tym ograniczeniem do G dodajemy $O(k \log n)$ krawędzi.

Graf G ma zatem $O(n+m)$ wierzchołków i $O(S \log n)$ krawędzi. Stosując ponownie algorytm upraszczający ograniczenia, otrzymujemy rozwiązanie wzorcowe o złożoności czasowej $O(n + S \log n)$. Implementacja tego rozwiązania znajduje się w pliku `pus3.cpp`.

Testy

Przygotowano 10 grup testów. Większość testów z odpowiedzią `TAK` była generowana w dwóch niezależnych fazach:

1. Wybór przykładowego ciągu spełniającego warunki i wybór zbioru $\mathcal{F}_0$. Ciągi były generowane za pomocą kilku procedur o różnych stopniach losowości.
2. Wybór ograniczeń spełnionych dla ustalonego ciągu. Dwie przykładowe metody wyboru ograniczeń to metoda losowa i wybór dla każdego i maksymalnego przedziału, dla którego a_i jest maksymalnym elementem tego przedziału.

W testach z odpowiedzią `NIE`, generowany był graf G z cyklem, który następnie był zamieniany na wejściowe ograniczenia, bądź wymuszane było, aby niektóre elementy a_i dla $i \notin \mathcal{F}_0$ musiały pochodzić spoza przedziału $[1, 10^9]$.