Krasnoludki

PREOI 2025 Kod zadania: **kra**
Dzień 3 – 27 stycznia 2025

Sekcje Podzadań zawierają kolejne kroki potrzebne do rozwiązania zadania. Zachęcam do czytania w kolejności.

Podzadanie 1

W tym podzadaniu wystarczy symulować ścieżkę każdego krasnoludka z osobna. Jak się zaraz okaże takie rozwiązanie daje nam rozwiązanie $O(n^2mq)$.

Podzadanie 2

Żeby rozwiązać to zadanie szybciej musimy zastanowić się jak będzie wyglądała ścieżka krasnoludka. Zaczyna on od przesunięcia się o 1 w prawo i zostawienia jednego znaku < po jego lewej. Jeśli kolejny znak jest > to będziemy «, w przeciwnym wypadku zawrócimy i otrzymamy ». Każdy element przez który przejdziemy co najmniej raz zostanie "zjedzony"przez nasz przedział, który będzie składał się z samych > jeśli jesteśmy po jego lewej stronie, oraz samych < jeśli jesteśmy po prawej stronie. Jeśli krasnoludek wyjdzie w lewo, to nasz przedział będzie prefiksem złożonym z >. Jeśli w prawo, to otrzymamy sufiks z <.

Za zmianę kierunku odpowiedzialne są tylko elementy < po prawej stronie startowego oraz > po lewej. Niech a oznacza liczbę < po prawej stronie, a b liczbę > po lewej. Jeśli $a \leq b$ to skończymy po prawej stronie, a sufiks, którego dotkniemy będzie zaczynał się od a-tego (licząc w lewo od punktu startowego) znaku >. W przeciwnym wypadku skończymy po lewej stronie zmieniając prefiks kończący się na $b+1$-szym znaku < (licząc w prawo od punktu startowego).

Jesteśmy więc w stanie określać stan po jednym krasnoludku w czasie $O(n)$, co daje nam złożoność $O(nmq)$, która wystarczy, żeby przejść to podzadanie.

Podzadanie 3

Korzystając z poprzednich obserwacji jesteśmy w stanie przyspieszyć symulowanie pojedyńczego krasnoludka utrzymując drzewo przedziałowe, które pozwala nam:

1. Pytać o ilość < lub > na przedziale.
2. Ustawiać znak na przedziale.
3. Znajdować k-ty znak < lub > licząc od pewnego elementu.

Najprostszą strukturą umożliwiającą pierwsze dwie operacje będzie drzewo przedział-przedział. Trzecią operację możemy obsługiwać wyszukując się binarnie po wyniku pierwszej w $O(log^2 n)$ lub $O(logn)$ dla pojedyńczego krasnoludka. Daje nam to złożoność $O(mqlog^2 n)$ lub $O(mqlogn)$ w zależności od implementacji.

Wiele krasnoludków naraz

Żeby być w stanie rozwiązać zadanie w pełni, będziemy musieli w jakiś sposób symulować wiele krasnoludków na raz. Możemy zatem zastanowić się, jak będzie wyglądać ciąg, po kilku krasnoludkach. Ponieważ krasnoludki ustawiają > na prefiksie lub < na sufiksie, jeśli zaczniemy kolejnym z tak ustawionego prefiksu, to albo skończymy na sufiksie skracając przy tym długość takiego prefiksu, albo długośc prefiksu się przedłuży.

Nasz stan będzie składał się zawsze z prefiksu złozonego z >, sufiksu złożonego z <, oraz pewnego (być może pustego) przedziału nie ruszonych elementów. W szczególności ten przedział będzie się ciągle zmniejszał, do momentu kiedy nie zniknie w całości. W momencie kiedy przedział zniknie, nasz ciąg już zawsze będzie składał się z t $(0 \leq t \leq n)$ znaków

$>$ na początku oraz $n - t$ znaków $<$ na końcu. Dla wygody od teraz taki stan po "spotkaniu"prefisku i sufiksu będziemy określać jako t (długośc prefiksu $>$).

Dla krasnoludka, który zaczyna ze stanu t oraz pozycji a:

- Jeśli $a \leq t$ to stan po przejściu tego krasnoludka zmieni się na $(t + a) mod (n + 1)$.
- Jeśli $a > t$ to stan po przejściu tego krasnoludka zmieni się na $(t + a + 1) mod (n + 1)$.

Możemy przekonać się o słuszności tych stwierdzeń rozpatrzając kilka przypadków na kartce (a małe, t małe, a duże t duże itp).

Daje nam to bardzo prosty algorytm na liczenie stanu po "spotkaniu"działający w czasie $O(nmq)$.

Podzadanie 4

Dla małego n możemy przyspieszyć ten algorytm traktując krasnoludka, jako operację, która dla każdego z $n + 1$ początkowych stanów będzie mówiła na jaki stan się on zmieni. Złożenie dwóch operacji też będzie taką samą operacją, więc możemy utrzymywać je na drzewie przedziałowym, co da nam złożoność $O(nlogm)$ na jedno pytanie. Całość będzie działała w $O((m + q)nlogm)$.

Podzadanie 5

W rozwiązaniu poprzedniego podzadania kosztowne jest tak naprawdę tylko budowanie drzewa, przy odpowiadaniu na pytanie możemy iterować się po przedziałach bazowych od lewej do prawej i patrzeć tylko na jaką wartość przechodzi aktualny element, więc możemy rozwiązać je w czasie $O(mnlogm)$ na budowanie oraz $O(qlogm)$ na odpowiadanie na pytania. Zastanówmy się teraz, czy w przypadku dużego n możemy w jakiś sposób trzymać skompresowaną wersję takiego drzewa.

W takim wektorze przejść możemy zauważyć pewien niezmiennik: Zaczynając od najmniejszej wartości i idąc (cyklicznie) w prawo nasz ciąg jest niemalejący. Zaczynamy z wektora identyczności i za każdym razem dwa elementy z tą samą wartością dalej będą miały tą samą wartość. Niektóre elementy zwiekszamy dodatkowo o 1 względem wszystkich, więc element z wartością a będzie co najwyżej równy elementowi, który przed operacją ma wartość $a + 1$.

Jeśli podzielimy wektor wierzchołka drzewa przedziałowego na przedziały, na których dodajemy tą samą wartość, to wierzchołek o rozmiarze poddrzewa x będzie miał co najwyżej $2x$ takich przedziałów.

Budowanie takiego drzewa zajmie nam $O(mlogm)$, jednak teraz "przechodzenie"przez wierzchołek bazowy nie działa w $O(1)$, tylko w $O(logm)$. Mamy zatem rozwiązanie działające w $O(mlogm + qlog^2m)$.