Informatyczny kącik olimpijski (128): *Trzej znajomi*

W tym odcinku omówimy rozwiązanie zadania „Trzej znajomi", które pojawiło się na drugim etapie XIII Młodzieżowej Olimpiady Informatycznej.

Trzej znajomi: Adam, Błażej i Cezary postanowili napisać ciąg n liczb naturalnych. Każdy z nich miał inny pomysł na ciąg. Adam chciał napisać ciąg $a = (a_1, a_2, \ldots, a_n)$, Błażej ciąg $b = (b_1, b_2, \ldots, b_n)$, zaś Cezary ciąg $c = (c_1, c_2, \ldots, c_n)$. Znajomi długo nie mogli dojść do porozumienia, dlatego postanowili zbudować nowy ciąg $d = (d_1, d_2, \ldots, d_n)$. Ustalili, że na każdej pozycji wybiorą którąś ze swoich opcji, czyli $d_i \in \{a_i, b_i, c_i\}$. Koledzy zgadzają się, że różnica pomiędzy największym a najmniejszym elementem ciągu d powinna być jak najmniejsza. Jaką najmniejszą różnicę może mieć ciąg d?

Rozwiązanie $O(3^n \cdot n)$
Zacznijmy od rozwiązania, które generuje wszystkie możliwe ciągi d. Takich ciągów jest 3^n (wartość każdego z n elementów można wybrać na 3 sposoby). Spośród wygenerowanych ciągów należy wybrać ten, który ma najmniejszą różnicę pomiędzy największym a najmniejszym elementem, oraz wypisać tę różnicę. Dla ustalonego ciągu d znalezienie najmniejszego i największego elementu można wykonać w czasie $O(n)$ (przeglądamy kolejne elementy, pamiętając najmniejszy oraz największy). Rozwiązanie działa w czasie $O(3^n \cdot n)$.

Obserwacja:
Zauważmy, że dla ustalonych całkowitych x, y $(x \leqslant y)$ możemy w czasie $O(n)$ sprawdzić, czy istnieje ciąg d, którego wartości elementów należą do przedziału $[x; y]$. Warunkiem koniecznym i wystarczającym jest, aby dla każdego indeksu i $(1 \leqslant i \leqslant n)$ przynajmniej jeden z trzech elementów a_i, b_i, c_i należał do $[x; y]$.

Rozwiązanie $O((m_2 - m_1)^2 \cdot n)$
Niech m_1 i m_2 oznaczają odpowiednio wartość najmniejszego i największego elementu w ciągach a, b i c. Korzystając z powyższej obserwacji, wystarczy dla wszystkich par liczb całkowitych x i y $(m_1 \leqslant x \leqslant y \leqslant m_2)$ sprawdzić, czy istnieje ciąg d, którego wartości należą do przedziału $[x; y]$. Spośród poprawnych przedziałów należy wybrać ten o najmniejszej różnicy $y - x$. Wszystkich przedziałów do sprawdzenia jest $O((m_2 - m_1)^2)$. Zatem rozwiązanie zajmuje czas $O((m_2 - m_1)^2 \cdot n)$.

Rozwiązanie $O(n^3)$
Niech S będzie zbiorem wartości, które występują przynajmniej raz w ciągu a, b lub c. Zauważmy, że wystarczy rozważać tylko takie przedziały $[x; y]$, dla których $x \leqslant y$ oraz $x, y \in S$. Moc zbioru S jest rzędu $O(n)$ (co najwyżej $3n$ różnych wartości może pojawić się w a, b, c, ponieważ tyle jest w sumie elementów). Stąd do sprawdzenia mamy $O(n^2)$ przedziałów, zatem całkowita złożoność czasowa wynosi $O(n^3)$.

Rozwiązanie $O(n^2 \cdot \log(n))$
Okazuje się, że nie musimy sprawdzać wszystkich wyżej wymienionych przedziałów. Wystarczy, że dla każdego $x \in S$ wyznaczymy takie najmniejsze y, że istnieje ciąg d, którego wartości elementów należą do $[x; y]$. Ustalmy $x \in S$, dla którego szukamy najmniejszego y. Następnie wartość y wyszukajmy binarnie w przedziale $[x; \max(S)]$. Rozwiązanie działa w czasie $O(n^2 \cdot \log(n))$.

Rozwiązanie $O(n^2)$
Podobnie jak wyżej, dla każdego $x \in S$ wyznaczymy najmniejsze poprawne y. W tym celu skonstruujemy ciąg d, którego największy element będzie możliwie najmniejszy. Dla każdego d_i mamy trzech kandydatów a_i, b_i, c_i. Spośród nich wybieramy najmniejszego, który jest większy lub równy x. Jeśli taki element nie istnieje, to znaczy, że nie istnieje ciąg d z minimalnym elementem x. W przeciwnym przypadku $y = \max(d_1, d_2, \ldots, d_n)$. Dla ustalonego x wyznaczenie minimalnego y odbywa się w czasie $O(n)$, zatem całe rozwiązanie działa w czasie $O(n^2)$.

Rozwiązanie $O(n \cdot \log(n))$
Każdy z elementów ciągów a, b i c opiszmy parą liczb: wartość i indeks. W ten sposób otrzymamy $3n$ par: $(a_1, 1), (a_2, 2), \ldots, (a_n, n)$, $(b_1, 1), (b_2, 2), \ldots, (b_n, n)$, $(c_1, 1), (c_2, 2), \ldots, (c_n, n)$. Następnie posortujmy te pary rosnąco względem pierwszej współrzędnej, aby otrzymać ciąg $(p_1, q_1), (p_2, q_2), \ldots, (p_{3n}, q_{3n})$. Zauważmy, że z ciągu $p_i, \ldots, p_j$ (dla $1 \leqslant i \leqslant j \leqslant 3n$) można wybrać n elementów, które utworzą ciąg d, jeśli $\{1, 2, \ldots, n\} \subseteq \{q_i, q_{i+1}, \ldots, q_j\}$. Innymi słowy, dla każdej pozycji (od 1 do n) musimy ustalić przynajmniej jeden element. Zatem chcemy znaleźć takie indeksy i, j $(1 \leqslant i \leqslant j \leqslant 3n)$, że z $p_i, \ldots, p_j$ można zbudować ciąg d oraz $p_j - p_i$ jest minimalne. Wystarczy, że dla każdego i od 1 do $3n$ znajdziemy najmniejsze takie $j \geqslant i$, że z elementów $p_i, \ldots, p_j$ można zbudować ciąg d. Do rozwiązania tego problemu wykorzystamy metodę gąsienicy. Na początku $i = j = 1$. Dopóki fragment $p_i, \ldots, p_j$ nie pozwala stworzyć poprawnego ciągu, to zwiększamy j o jeden. W przeciwnym przypadku przechodzimy do kolejnego i. W każdym kroku zliczamy liczbę wystąpień każdego indeksu na rozpatrywanym fragmencie oraz pamiętamy liczbę różnych indeksów, które występują przynajmniej raz. Jeśli ta wartość jest równa n, to wtedy rozpatrywany fragment tworzy ciąg d.

Sortowanie działa w czasie $O(n \cdot \log(n))$, zaś faza szukania najkrótszych fragmentów dla każdego początku zajmuje czas $O(n)$. Zatem całe rozwiązanie działa w czasie $O(n \cdot \log(n))$.

Bartosz ŁUKASIEWICZ