

Informatyczny kącik olimpijski (115): Gotówka, Startup

Tym razem omówimy dwa zadania, które na pierwszy rzut oka wyglądają podobnie, jednak w istocie są różne. Pierwsze z nich pochodzi z II etapu IX Olimpiady Informatycznej Gimnazjalistów, zaś drugie z Bałkańskiej Olimpiady Informatycznej Juniorów w 2015 roku.

Zadanie Gotówka: Danych jest n transakcji opisanych za pomocą ciągu liczb całkowitych $a = a_1, a_2, \dots, a_n$ (dodatnie liczby to wpływy, zaś ujemne to wydatki). Poprawnym ciągiem transakcji nazywamy taki, podczas którego stan konta nigdy nie spadnie poniżej zera. Ile jest permutacji ciągu a , które są poprawnymi ciągami transakcji?

Rozwiązanie $O(n! \cdot n)$

Pierwszy, najbardziej intuicyjny, pomysł polega na wygenerowaniu wszystkich $n!$ permutacji ciągu a oraz zliczeniu tych, które są poprawne. Sprawdzenie, czy dana permutacja transakcji jest poprawna, możemy wykonać w czasie liniowym od jej długości. Wystarczy, że sprawdzimy, czy suma każdego prefiksu jest nieujemna. Niestety, takie podejście jest czasochłonne i działa w czasie $O(n! \cdot n)$.

Rozwiązanie $O(n \cdot 2^n)$

W tym rozwiązaniu wykorzystamy technikę programowania dynamicznego. Niech $DP[s]$ oznacza liczbę poprawnych ciągów transakcji, które są permutacją ciągu s . Obliczmy wartość DP dla każdego podciągu $s = s_1, s_2, \dots, s_m$ ciągu a . Jeśli s jest pusty lub suma elementów podciągu s jest ujemna, wtedy $DP[s] = 0$. W przeciwnym przypadku istnieje przynajmniej jedna poprawna permutacja ciągu s . Aby obliczyć $DP[s]$, rozważmy m możliwości wybrania transakcji, która zostanie wykonana jako ostatnia. Liczba poprawnych ciągów transakcji, przy założeniu, że ostatnia transakcja to s_i , wynosi $DP[s_1, \dots, s_{i-1}, s_{i+1}, \dots, s_m]$ (liczba poprawnych uporządkowań $m - 1$ pozostałych transakcji). Zatem:

$$DP[s] = \sum_{i=1}^m DP[s_1, \dots, s_{i-1}, s_{i+1}, \dots, s_m].$$

Liczba poprawnych permutacji dla całego ciągu to $DP[a]$.

Ciąg a ma 2^n podciągów. Obliczenie $DP[s]$ dla i -elementowego podciągu s kosztuje i operacji, zaś i -elementowych podciągów jest $\binom{n}{i}$. Zatem obliczenie wszystkich wartości DP wymaga wykonania

$$\sum_{i=1}^n \binom{n}{i} \cdot i = n \cdot 2^{n-1}$$

operacji. Rozwiązanie działa w czasie $O(n \cdot 2^n)$.

Zadanie Startup: Dany jest ciąg n transakcji opisanych za pomocą ciągu liczb całkowitych $a = a_1, a_2, \dots, a_n$ (dodatnie liczby to wpływy, zaś ujemne to wydatki). Poprawnym ciągiem transakcji nazywamy taki, podczas którego stan konta nigdy nie spadnie poniżej zera. Ile jest rotacji ciągu a , które są poprawnymi ciągami transakcji? Rotacją ciągu a nazywamy n -elementowy ciąg, powstały poprzez konkatencję sufiksu i prefiksu ciągu a . Rotacjami ciągu $(1, 2, 3, 4)$ są: $(1, 2, 3, 4)$, $(2, 3, 4, 1)$, $(3, 4, 1, 2)$ i $(4, 3, 2, 1)$.

Rozwiązanie $O(n^2)$

Zauważmy, że wszystkich rotacji jest n (każdy element wyznacza początek rotacji). Wystarczy zatem każdą z nich rozpatrzeć niezależnie. Sprawdzenie, czy dana rotacja jest poprawnym ciągiem transakcji, możemy wykonać w czasie liniowym od jej długości. Wystarczy sprawdzić, czy suma każdego prefiksu jest nieujemna. W ten sposób otrzymujemy rozwiązanie, które działa w czasie $O(n^2)$.

Rozwiązanie $O(n \cdot \log(n))$

Zastanówmy się teraz, czy możemy przyspieszyć fazę sprawdzania, czy dana rotacja jest poprawnym ciągiem transakcji – oczywiście możemy. Zaczniemy od sklejenia dwóch kopii ciągu a w jeden ciąg b :

$$b = a_1, a_2, \dots, a_n, a_1, a_2, \dots, a_n.$$

Wówczas każda rotacja jest podslowem ciągu b .

Niech $p = p_0, p_1, p_2, \dots, p_{2n}$ będzie ciągiem sum prefiksowych dla ciągu b (tzn. $p_k = \sum_{i=1}^k b_i$).

Załóżmy, że chcemy sprawdzić poprawność rotacji $a_x, a_{x+1}, \dots, a_n, a_1, \dots, a_{x-1}$. Odpowiadającym podslowem słowa b jest $b_x, b_{x+1}, \dots, b_{x+n-1}$. Podslowo jest poprawnym ciągiem transakcji, jeśli każdy prefiks ma nieujemną sumę:

$$\forall_{i=1}^n \left(\sum_{j=1}^i b_{x+j-1} \geq 0 \right).$$

Powyższy warunek możemy równoważnie zapisać jako

$$\forall_{i=1}^n (p_{x+i-1} - p_{x-1} \geq 0)$$

oraz jako

$$\forall_{i=1}^n (p_{x-1} \leq p_{x+i-1}).$$

Zauważmy, że powyższy warunek jest prawdziwy wtedy i tylko wtedy, gdy

$$p_{x-1} \leq \min(p_x, p_{x+1}, \dots, p_{x+n-1}).$$

W celu wyznaczenia minimum na przedziale możemy skorzystać ze struktury drzewa przedziałowego rozpiętego na ciągu p . Drzewo przedziałowe pozwala w czasie $O(\log(n))$ znajdować minimum na przedziale – czyli sprawdzać, czy dana rotacja jest poprawna. Wszystkich rotacji jest n , zatem rozwiązanie działa w czasie $O(n \cdot \log(n))$.

Bartosz ŁUKASIEWICZ