Informatyczny kącik olimpijski (121): *Egzamin*

Tym razem omówimy zadanie *Egzamin*, które pojawiło się w 2010 roku na *Junior Balkan Olympiad in Informatics* w Szumen (Bułgaria).

Egzamin: *Dany jest ciąg $a = (a_1, a_2, \ldots, a_n)$, złożony z n liczb naturalnych, oraz liczba s. Ile jest podciągów ciągu a, których suma elementów wynosi przynajmniej s? Dla przykładu, $a = (2, 5, 3, 5)$ ma 6 podciągów, które mają sumę przynajmniej $s = 10$. Są to podciągi: $(2, \underline{5}, 3, \underline{5})$, $(\underline{2}, \underline{5}, \underline{3}, 5)$, $(\underline{2}, \underline{5}, 3, \underline{5})$, $(\underline{2}, 5, \underline{3}, \underline{5})$, $(2, \underline{5}, \underline{3}, \underline{5})$ oraz $(\underline{2}, \underline{5}, \underline{3}, \underline{5})$.*

Rozwiązanie $O(n \cdot 2^n)$
Pomysł, który jako pierwszy nasuwa się na myśl, polega na obliczeniu sumy każdego podciągu, a następnie zliczeniu tych sum, których wartość jest nie mniejsza niż s.

Zastanówmy się teraz, jaka jest złożoność opisanego rozwiązania. Ciąg a ma $\binom{n}{1}$ podciągów długości 1, $\binom{n}{2}$ podciągów długości $2, \ldots,$ wreszcie $\binom{n}{n}$ podciągów długości n. Zatem, suma długości wszystkich podciągów wynosi: $1 \cdot \binom{n}{1} + 2 \cdot \binom{n}{2} + \ldots + n \cdot \binom{n}{n} = n \cdot 2^{n-1}$. Powyższą sumę można zinterpretować nieco inaczej: każdy z n elementów ciągu a należy do 2^{n-1} podciągów (dla ustalonego elementu $n-1$ pozostałych elementów tworzy 2^{n-1} podciągów). Zatem liczba składników we wszystkich sumach wynosi $n \cdot 2^{n-1}$, co daje nam złożoność czasową $O(n \cdot 2^n)$.

Rozwiązanie $O(2^n)$
Spróbujmy przyspieszyć powyższe rozwiązanie. Będziemy obliczali sumy podciągów w kolejności niemalejących długości (najpierw podciągi długości jeden, potem długości dwa, trzy, itd.). Załóżmy, że obliczamy sumę podciągu $a_{i_1}, a_{i_2}, \ldots, a_{i_m}$ (podciąg ma długość m, indeksy kolejnych elementów tworzą ciąg $i_1, i_2, \ldots, i_m$). Suma elementów tego ciągu to suma podciągu $a_{i_1}, a_{i_2}, \ldots, a_{i_{m-1}}$ (którą obliczyliśmy wcześniej, ponieważ przeglądamy podciągi od najkrótszych do najdłuższych) powiększona o a_{i_m}. W tym podejściu obliczenie sumy każdego z $2^n - 1$ podciągów odbywa się w czasie stałym, co daje nam złożoność czasową $O(2^n)$.

Rozwiązanie $O(n \cdot 2^{\frac{n}{2}})$
W tym rozwiązaniu skorzystamy z techniki *Meet in the middle*.

Podzielmy ciąg a na dwa ciągi równej długości (jeśli długość ciągu jest nieparzysta, wtedy niech pierwszy ciąg zawiera o jeden element więcej): lewy $L = (a_1, a_2, \ldots, a_{\lceil \frac{n}{2} \rceil})$ oraz prawy $P = (a_{\lceil \frac{n}{2} \rceil + 1}, a_{\lceil \frac{n}{2} \rceil + 2}, \ldots, a_n)$. Następnie, dla każdego z nich, obliczmy sumy podciągów. Możemy to zrobić w czasie $O(2^{\frac{n}{2}})$, korzystając z algorytmu opisanego w poprzedniej sekcji. Niech $S^L = (S^L_1, S^L_2, \ldots, S^L_l)$ oznacza sumy podciągów L, i niech $S^P = (S^P_1, S^P_2, \ldots, S^P_p)$ oznacza sumy podciągów P.

Zauważmy teraz, że dowolny podciąg q ciągu a spełnia jeden z trzech poniższych warunków:

Warunek (I): wszystkie elementy q należą do L.
Aby obliczyć liczbę podciągów o sumie przynajmniej s, których wszystkie elementy należą do L, wystarczy zliczyć te elementy ciągu S^L, które mają wartość przynajmniej s. Formalnie jest to moc zbioru: $\{i \in \{1, 2, \ldots, l\} : S^L_i \geqslant s\}$. Sprawdzenie tego warunku odbywa się w czasie liniowym od liczby podciągów L, czyli $O(2^{\frac{n}{2}})$.

Warunek (II): wszystkie elementy q należą do P.
Analogicznie jak warunek (I).

Warunek (III): część elementów q należy do L, część należy do P.
W tym przypadku chcemy obliczyć liczbę takich par (i, j) dla $i \in \{1, 2, \ldots, l\}$ oraz $j \in \{1, 2, \ldots, p\}$, że $S^L_i + S^P_j \geqslant s$. Ten problem rozbijemy na l podproblemów. Dla każdego $i \in \{1, 2, \ldots, l\}$ obliczymy, ile jest takich $j \in \{1, 2, \ldots, p\}$, że $S^L_i + S^P_j \geqslant s$.

W tym celu posortujmy ciąg S^P i otrzymany ciąg nazwijmy S'^P. Załóżmy, że obliczamy wynik dla ustalonego $i \in \{1, 2, \ldots, l\}$. Za pomocą wyszukiwania binarnego szukamy takiego najmniejszego $k \in \{1, 2, \ldots, p\}$, że $S^L_i + S'^P_k \geqslant s$. Jeśli takie k istnieje, wtedy dla każdego $j \in \{k, k+1, \ldots, p\}$ zachodzi $S^L_i + S'^P_j \geqslant s$, ponieważ S'^P jest niemalejący. Stąd, S^L_i należy do $p - k + 1$ par, które mają sumę przynajmniej s. Jeśli zaś takie k nie istnieje, to S^L_i nie tworzy żadnej pary o sumie przynajmniej s.

Faza sortowania zajmuje czas $O(n \cdot 2^{\frac{n}{2}})$. Wyznaczenie liczby poprawnych par, dla ustalonej wartości, za pomocą wyszukiwania binarnego zajmuje czas $O(n)$. Wszystkich wartości do sprawdzenia jest l (l jest rzędu $O(2^{\frac{n}{2}})$), co daje całkowitą złożoność $O(n \cdot 2^{\frac{n}{2}})$.

Przykład
Prześledźmy opisany algorytm na przykładzie ciągu $a = (2, 5, 3, 4, 2, 4)$, w którym szukamy liczby podciągów o sumie przynajmniej 10. Najpierw dzielimy ciąg a na dwa ciągi i dla każdego z nich obliczamy sumy wszystkich jego podciągów:

$L = (2, 5, 3)$	$P = (4, 2, 4)$
$(\underline{2}, 5, 3) \to 2$	$(\underline{4}, 2, 4) \to 4$
$(2, \underline{5}, 3) \to 5$	$(4, \underline{2}, 4) \to 2$
$(2, 5, \underline{3}) \to 3$	$(4, 2, \underline{4}) \to 4$
$(\underline{2}, \underline{5}, 3) \to 7$	$(\underline{4}, \underline{2}, 4) \to 6$
$(\underline{2}, 5, \underline{3}) \to 5$	$(\underline{4}, 2, \underline{4}) \to 8$
$(2, \underline{5}, \underline{3}) \to 8$	$(4, \underline{2}, \underline{4}) \to 6$
$(\underline{2}, \underline{5}, \underline{3}) \to 10$	$(\underline{4}, \underline{2}, \underline{4}) \to 10$
$S^L = (2, 5, 3, 7, 5, 8, 10)$	$S^P = (4, 2, 4, 6, 8, 6, 10)$

Z warunku (I) mamy 1 podciąg, z warunku (II) również 1. Przechodzimy do warunku (III). Najpierw sortujemy S^P, otrzymując $S'^P = (2, 4, 4, 6, 6, 8, 10)$. Teraz wyznaczamy wyniki dla kolejnych elementów S^L: $(2, 4, 2, 6, 4, 7, 7)$. Ostatecznym wynikiem jest: $(1) + (1) + (2 + 4 + 2 + 6 + 4 + 7 + 7) = 1 + 1 + 32 = 34$.

Bartosz ŁUKASIEWICZ