

Spóźnienia przy okrągłym stole

Pełną treść zadania (z przykładami) można znaleźć na stronie oij.edu.pl/oij20/etap2/zadania/spo/spozad.pdf.

W tym odcinku pochylimy się nad zadaniem *Spóźnienia* z zawodów II stopnia XX Olimpiady Informatycznej Juniorów.

Rozważmy naradę z udziałem N rycerzy siedzących przy okrągłym stole. Każdy rycerz wygłosi jedno przemówienie w kolejności zajmowanych miejsc. Pierwszy zacznie przemawiać pewien rycerz s , po nim głos zabierze rycerz $s + 1$ i tak dalej (przy czym po rycerzu N będzie przemawiał rycerz 1), aż każdy przemówi. Każdy rycerz i jest opisany przez dwa parametry: czas trwania jego przemówienia A_i oraz moment przybycia na spotkanie B_i (zakładamy, że narada zaczyna się w chwili 0). Jeśli nadejdzie kolej rycerza, którego nie ma jeszcze przy stole, wszyscy muszą czekać na jego pojawienie się. Naszym zadaniem jest obliczenie czasu trwania takiego spotkania dla każdej możliwej wartości s .

Na przekór cykлом

Pierwszym krokiem rozwiązania jest eliminacja problematycznego zawijania indeksów na okręgu. Problem ten możemy rozwiązać poprzez połączenie ciągu wejściowego z jego kopią. Możemy więc stworzyć nowe rozszerzone ciągi o długości $2N$, gdzie zachodzą następujące warunki dla $k \in [N + 1, 2N]$:

$$A_k = A_{k-N} \text{ oraz } B_k = B_{k-N}.$$

Dzięki temu każde pełne okrażenie stołu rozpoczynające się od rycerza s można traktować jako spójny podciąg o stałej długości N w tych rozszerzonych ciągach.

Największy spóźniony

Przeanalizujmy sytuację, w której obrady rozpoczyna konkretny rycerz s . Moment zakończenia całego spotkania zależy przede wszystkim od rycerza, na którego przemówienie będziemy musieli czekać ostatni raz. Dla dowolnego rycerza j wyznaczmy więc czas, w którym narada dobiegłaby końca, gdyby to właśnie on był tym wąskim gardłem systemu. Czas ten stanowi po prostu sumę momentu jego przybycia B_j oraz łącznego czasu przemówień jego i wszystkich pozostałych rycerzy przemawiających po nim. Możemy to zapisać następująco:

$$C_j = B_j + \sum_{k=j}^{s+N-1} A_k.$$

Prawdziwy całkowity czas narady W_s oczywiście jest wyznaczany przez tę osobę, która wymusza najpóźniejszy termin jej zakończenia. Szukamy zatem wartości maksymalnej spośród wszystkich uczestników:

$$W_s = \max_{s \leq j < s+N} C_j.$$

Sumy prefiksowe

Aby zoptymalizować sumowanie czasów wystąpień A_k , możemy wprowadzić ciąg sum prefiksowych P , którego elementy definiuje się następująco:

$$P_m = \sum_{k=1}^m A_k.$$

Dzięki temu sumę czasów na dowolnym przedziale od indeksu j do końcowego indeksu $s + N - 1$ możemy obliczyć w czasie stałym, wyznaczając różnicę $P_{s+N-1} - P_{j-1}$. Podstawiając tę zależność do wcześniejszego równania, otrzymujemy nową postać czasu narady determinowaną przez uczestnika j :

$$C_j = B_j + P_{s+N-1} - P_{j-1}.$$

Ostateczny wzór na całkowity czas trwania narady W_s przy starcie od rycerza s przyjmuje wtedy następującą formę:

$$W_s = \max_{s \leq j < s+N} (B_j - P_{j-1}) + P_{s+N-1}.$$

Optymalizacja w oknie

Składnik P_{s+N-1} pozostaje stały dla wybranego rycerza startowego s . Głównym wyzwaniem jest zatem sprawne odnajdowanie wartości maksymalnej wyrażenia $B_j - P_{j-1}$ w oknie o stałej szerokości N .

Jeśli zdefiniujemy nową zmienną $V_j = B_j - P_{j-1}$, zadanie sprowadza się do klasycznego problemu znajdowania ekstremum (w tym przypadku maksimum) w oknie.

Do tego celu możemy skorzystać ze struktury kolejki monotonicznej, która przechowuje w dwustronnej kolejce (*deque*) indeksy elementów w taki sposób, aby na jej początku zawsze znajdował się kandydat na aktualne maksimum. Podczas przesuwania okna w prawo i dodawania nowej wartości V_j usuwamy z końca kolejki wszystkie indeksy, których wartości są mniejsze od nowo przybyłej (mniejsze wartości nigdy nie staną się maksimum, skoro nowa i większa wartość będzie obecna w oknie dłużej). Następnie sprawdzamy, czy element na początku kolejki nie wyszedł już poza zakres okna ze względu na swój indeks i w razie potrzeby go usuwamy. Dzięki tej strategii aktualne maksimum znajduje się zawsze bezpośrednio na początku kolejki. Każdy element jest dodawany i usuwany z kolejki co najwyżej raz.

Ostatecznie zatem jesteśmy w stanie obliczyć wszystkie wartości W_s dla $1 \leq s \leq N$ w złożoności czasowej (i pamięciowej) $\mathcal{O}(N)$.

Bartosz KOSTKA