4C – Naszyjnik

Autor zadania: Marcin Smulewicz

Autor omówienia: Paweł Parys

Treść

Dany jest ciąg liczb $a_1, \ldots, a_n$. Dla ustalonego przesunięcia cyklicznego tego ciągu *zachwytem* nazwiemy takie miejsce, na którym stoi liczba ściśle większa niż wszystkie występujące przed nią w tym przesunięciu cyklicznym (innymi słowy: jest to maksimum na prefiksie). Interesuje nas takie przesunięcie cykliczne, w którym jest jak najwięcej zachwytów. Należy wypisać liczbę tych zachwytów.

Rozwiązanie

W pierwszym kroku przesuwam ciąg cyklicznie tak, aby największa liczba (dokładniej: dowolna z największych) znalazła się na samym końcu. Oczywiście nie zmienia to zbioru dostępnych przesunięć cyklicznych, a zatem także wyniku. Daje nam jednak ciekawą własność: dla każdego miejsca x, liczba zachwytów na sufiksie ciągu zaczynającym się w x jest taka sama jak liczba zachwytów na przesunięciu cyklicznym ciągu zaczynającym się w x. Istotnie, gdy zobaczymy już ostatni element ciągu, który jest największy, to nie ma znaczenia, czy potem jeszcze damy początek naszego ciągu, czy już nic nie damy – tak czy inaczej nic większego już nie zobaczymy.

Po powyższej operacji wystarczy zatem wyznaczyć liczbę zachwytów na każdym z sufiksów naszego ciągu. Będziemy w tym celu przesuwać się od prawej do lewej i trzymać wszystkie zachwyty z aktualnego sufiksu na stosie, począwszy od największego. Zobaczmy co dzieje się, gdy do naszego sufiksu dokładamy z lewej kolejny element a. Otóż, wszystkie zachwyty nie większe niż a z poprzedniego sufiksu przestają być zachwytami. Zdejmujemy je więc ze stosu (są one na jego szczycie), a następnie wkładamy na stos liczbę a (pierwsza liczba zawsze jest zachwytem). Rozmiar naszego stosu to liczba zachwytów w rozważanym sufiksie; chcemy więc wypisać maksimum osiągane przez ten rozmiar podczas przechodzenia przez cały ciąg.

W implementacji należy zwrócić uwagę na to, że stos może stać się pusty (i wtedy porównywanie z wartością na szczycie stosu nie ma sensu). Można to łatwo obsłużyć dodatkowym warunkiem lub wstawiając na sam dół stosu jakąś dodatkową wielką wartość, która nigdy nie zostanie zdjęta („strażnika").

Mimo że przetwarzając niektóre pozycje musimy zdejmować ze stosu wiele elementów, to łączna liczba zdejmowanych elementów nie przekroczy łącznej liczby wstawianych elementów, a ta jest równa długości wejściowego ciągu. Powyższe rozwiązanie działa zatem w czasie $O(n)$.