Zad 3.

niedziela, 7 maja 2023 19:55

3. (2pkt). W n-elementowej tablicy A pamiętany jest rosnący ciąg liczb naturalnych. Nie znamy wartości jej elementów, ale możemy się o nie pytać. Pytanie o wartość $A[i]$ kosztuje nas c_i.

Ułóż algorytm, który dla danych liczb naturalnych $c_1, c_2, \ldots, c_n$ oraz liczby k obliczy najmniejszym kosztem (liczonym jako suma kosztów zadanych pytań), ile liczb w tablicy A ma wartość większą niż k.

zadanie polega zasadniczo na znalezieniu upper-bounda k

żeby dowiedzieć się jak to zrobić najmniejszym kosztem konstruujemy pomocniczą tablice

dp[l][r] - koszt znalezienia odpowiedzi na przedziale [l,r] + indeks z tego przedziału, o który najlepiej jest się zapytać

szukana wartość potencjalnie może się znajdować w każdym z pól tablicy A, dlatego chcemy wiedzieć o co mamy zapytać po wcześniejszym ograniczeniu przedziału do [l,r]

policzenie dp sprowadza się do poniższej zależności

najgorszy przypadek

$$dp[l][r] = \begin{cases} c_l, & l = r \\ \min\limits_{m \in (l,r)} \left(c_m + \max(dp[l][m-1], dp[m+1][r]) \right) \end{cases}$$

po wypełnieniu tablicy algorytm postępuje zgodnie ze schematem binsearcha, wybierając „środki" przedziałów według dp[l][r]

analiza złożoności: $O(n^3)$ czas, $O(n^2)$ pamięć

wypełniamy $\approx n^2$ komórek, w każdej rozważamy $\approx n$ punktów m