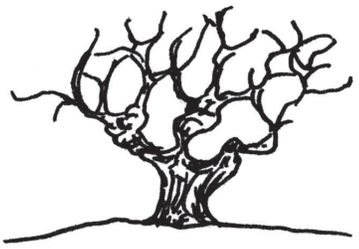
Informatyczny kącik olimpijski (13) – drzewo potęgowe



Zgodnie z obietnicą z zeszłego miesiąca w tym odcinku zajmiemy się znanym problemem i jego „alternatywnym" rozwiązaniem. Naszym zadaniem jest znalezienie w danym ciągu liczb naturalnych $A_1, \ldots, A_n$ najdłuższego podciągu rosnącego, czyli podciągu postaci $A_{k_1}, \ldots, A_{k_m}$ gdzie $k_1 < \ldots < k_m$ oraz $A_{k_1} < \ldots < A_{k_m}$.

Pokażemy, jak metodą programowania dynamicznego znaleźć długość takiego podciągu. W tym celu przeglądamy kolejne wyrazy ciągu A. Jeśli przez x oznaczymy bieżący wyraz, to żeby zaktualizować nasz stan wiedzy o rosnących podciągach w dotychczasowym prefiksie, chcielibyśmy zapytać „jaki jest najdłuższy podciąg rosnący kończący się w x?". Innymi słowy „jaki był do tej pory najdłuższy podciąg rosnący, kończący się wartością mniejszą od x?".

Do odpowiedzi na takie pytania przydałaby się nam struktura danych, która radzi sobie z następującymi zapytaniami dotyczącymi pewnej tablicy $V[1 \ldots N]$:

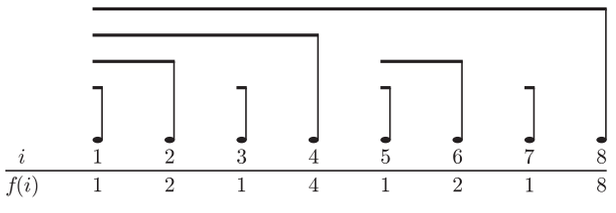
- Podstaw $V[i] := l$
- Podaj maksimum z wartości $V[1], V[2], \ldots, V[i]$.

W naszym problemie $V[i]$ będzie długością najdłuższego dotychczas podciągu rosnącego kończącego się liczbą i. Tablica V musi mieć tyle pól, ile wynosi największy wyraz ciągu. Alternatywnie, można zacząć od przeindeksowania ciągu tak, aby jego wyrazy były liczbami od 1 do n – zajmie to czas potrzebny na sortowanie, czyli $O(n \log n)$.

Struktura, którą zaraz opiszemy, nazywana jest *drzewem potęgowym*. Wykorzystuje się w niej tablicę $W[1 \ldots N]$. Dodatkowo oznaczmy przez $f(k)$ największą potęgę dwójki dzielącą k. W pomocniczej tablicy W będziemy przechowywać wartości:

$$W[i] = \max_{i-f(i)<j\leqslant i} V[j].$$

Na przykład dla nieparzystych i mamy $W[i] = V[i]$, dla i parzystych, ale niepodzielnych przez 4 – $W[i] = \max(V[i-1], V[i])$, itd. (patrz rysunek)



Drzewo potęgowe dla $n = 8$: u góry przedziały w V, za które „odpowiadają" poszczególne pola tablicy W. Obliczając maksimum z $V[1], \ldots, V[6]$, odczytujemy pola o indeksach 6 i $6 - f(6) = 4$. Zmieniając $V[3]$, będziemy musieli uaktualnić pola o indeksach 3, $3 + f(3) = 4$ i $4 + f(4) = 8$.

Obliczenie $\max(V[1], V[2], \ldots, V[i])$ wygląda tak:

$wynik := -\infty$
Dopóki $i > 0$:
  $wynik := \max(wynik, W[i])$
  $i := i - f(i)$

Poprawności tej procedury raczej nie trzeba dowodzić. A co z jej czasem wykonania? Zauważmy, że $f(i - f(i)) > f(i)$ dla dowolnego dodatniego i.

Faktycznie, gdy rozpiszemy i binarnie, widzimy, że operacja $i := i - f(i)$ wybiera najmniej znaczący bit ustawiony na 1 i zamienia go w 0. Tym samym pętla obraca się $O(\log i)$ razy (co szacuje się przez $O(\log N)$).

A jak wygląda aktualizacja tej struktury? Załóżmy, że zmieniamy wartość $V[i]$. Przez $j_1, j_2, \ldots$ oznaczmy wszystkie indeksy spełniające $j - f(j) < i \leqslant j$ (są to wszystkie pola w W, które musimy zaktualizować). Wyznaczymy je kolejno. Załóżmy, że $f(j_1) < f(j_2) < \ldots$ (wszystkie $f(j_k)$ są parami różne, dlaczego?). Kolejne spostrzeżenie – w przedziale od $j - f(j) + 1$ do j nigdy nie ma takiego j', że $f(j') > f(j)$. Z tego z kolei wynika, że również $j_1 < j_2 < \ldots$. Jeśli więc oznaczymy przez $g(x)$ najmniejsze takie y, że $y - f(y) < x < y$, to okazuje się, że $(j_1, j_2, \ldots) = (j_1, g(j_1), g(g(j_1)), \ldots)$ Co więcej, $i \leqslant j_1$, oraz na pewno i należy do zbioru pól, które musimy zaktualizować. W takim razie j_1 musi być równe i.

Pozostaje nam jeszcze wyznaczyć $g(x)$. Ustaliliśmy już, że na pewno $f(g(x)) > f(x)$. Najmniejszym kandydatem na $g(x)$ jest więc $x + f(x)$. Ale skoro $f(x + f(x)) > f(x)$, to $x + f(x) - f(x + f(x)) < x < x + f(x)$ – a więc $g(x) = x + f(x)$.

Ostatecznie aktualizacja wygląda bardzo prosto i także zajmuje czas $O(\log N)$:

Dopóki $i \leqslant N$:
  $W[i] := \max(W[i], l)$
  $i := i + f(i)$

Uwaga przydatna podczas implementacji tego algorytmu: $f(i) = \frac{i \oplus (i-1)+1}{2}$ (gdzie $\oplus$ to bitowy „xor").

Warto w tym momencie dodać, że drzewo potęgowe można stosować na wiele innych sposobów – tablica W może przechowywać inne funkcje od elementów $V[1], \ldots, V[i - f(i) + 1]$, na przykład sumę

$$\sum_{i-f(i)<j\leqslant i} V[i].$$

Mając taką strukturę, umiemy aktualizować V oraz szybko odpowiadać na pytania o sumy liczb z komórek od a do b w V. Co więcej, takie rozwiązanie można uogólnić na więcej wymiarów.

Filip WOLSKI