Kamil Matuszewski

21 kwietnia 2016

Zadanie 9

Znajdź najdłuższy podciąg rosnący ciągu $a_1 \dots a_k$ (longest increasing subsequence - LIS) (zadanie z bobrami - bobry to indeksy, ich wybranki to wartości. Zauważmy, że wtedy elementy są różne).

Chciałbym stworzyć tablicę $t_i(j)$ która będzie nam mówić, że j jest najmniejszym ostatnim elementem jakiegoś podciągu rosnącego długości i prefiksu $a_1 \dots a_i$. Jeśli taki element nie istnieje, niech $t_i(j) = \infty$. W ten sposób, największy indeks skończonego elementu tablicy t_n będzie nam mówić, jaki jest rozmiar najdłuższego podciągu rosnącego ciągu $a_1 \dots a_n$, co jest naszą szukaną wartością.

Udowodnijmy najpierw kilka rzeczy.

- W tablicy t_k dla danego prefiksu $a_1 \dots a_k$ wszystkie skończone elementy będą ustawione rosnąco.

  Dowód. Jeśli w prefiksie $a_1 \dots a_k$ mamy podciąg długości i zakończony przez $t(i)$, to oczywiście w $a_1 \dots a_k$ mamy podciąg długości $i-1$ zakończony $t(i)$, bo wystarczy usunąć pierwszy element podciągu długości i. Zachodzi więc $t(i-1) < t(i)$. Oczywiście, jeśli $t(i) = \infty$ nierówność jest trywialna. □

- Każde dwie tablice t_{k-1} oraz t_k różnią się w dokładnie jednym miejscu.

  Dowód. Rozważmy poszerzenie naszego prefiksu o a_k. Innymi słowy zmianę z t_{k-1} na t_k. Skoro ciąg jest rosnący to istnieje taki indeks tablicy, powiedzmy j, że elementy $t(1) \dots t(j)$ są mniejsze od a_k a elementy $t(j+1) \dots t(n)$ są większe od a_k. Teraz tak:

  1. Wiemy, że elementów $t(1) \dots t(j)$ nie poprawimy elementem a_k, ponieważ są tam już podciągi odpowiedniej długości, a zastąpienie ostatniego elementu przez a_k nie ma sensu, jako, że jest on większy od każdego z tych elementów.
  2. $t(j)$ mówi nam, że w prefiksie $a_1 \dots a_{k-1}$ jest podciąg rosnący o długości j zakończony elementem $t(j)$, a skoro $t(j) < a_k$ to na koniec tego podciągu możemy dodać a_k i otrzymamy podciąg długości $j+1$ zakończony elementem $a_k < t(j+1)$, czyli $t(j+1) = a_k$
  3. Elementy $t_{j+1} \dots t_n$ są większe od a_k więc nie możemy przedłużyć tych podciągów za pomocą a_k, więc nie zmienimy elementów $t_{j+2} \dots t_n$

  □

- Skoro zachodzą dwa powyższe warunki to możemy znajdować indeks do zmiany elementu w tablicy za pomocą wyszukiwania binarnego.

Skoro tak, to mamy algorytm:

Dane: Tablica liczb $a_1 \dots a_n$.
Wynik: Liczba s taka, że s jest rozmiarem najdłuższego podciągu rosnącego w $a_1 \dots a_k$.

Niech t - tablica rozmiaru n;
for $i \leftarrow 1$ **to** n **do**
  $t[i] \leftarrow \infty$;
end
for $i \leftarrow 1$ **to** n **do**
  $j \leftarrow BinarySearch(a_i, t)$;
  $\backslash\backslash BinarySearch(a_i, t)$ zwraca najmniejszy taki indeks j, że $a_i < t[j]$
  $t[j+1] \leftarrow a_i$;
end
$s \leftarrow 1$;
while $t[s] \neq \infty$ && $s < n$ **do**
  $s{+}{+}$;
end
return s

Algorytm 1: Długość LIS

Mamy pętlę przechodzącą n razy, a w każdej z nich dominującą operacją będzie wyszukiwanie binarne - $\log n$. Skoro tak, to złożoność czasowa naszego algorytmu to $O(n \log n)$. Przy złożoności pamięciowej potrzebujemy jednej tablicy t rozmiaru n, bo elementy możemy na bieżąco wrzucać do tablicy, co daje złożoność $O(n)$.