A simple algorithm for Lempel-Ziv factorization

*

19 maja 2022

Faktoryzacja Lempel-Ziv'a dla słowa w jest takim rozkładem $u_0u_1...u_k = w$, że każde u_i, za wyjątkiem możliwie ostatniego, jest albo najdłuższym prefiksem $u_iu_{i+1}...u_k$ i występuje jako podsłowo w $u_0u_1...u_i$, ale nie tylko jako sufiks, albo jest pojedynczym symbolem, gdy takiego prefiksu nie ma.

Authorzy proponują algorytm pozwalający obliczać faktoryzację w czasie liniowym i pamięci $o(n)$. Jeszcze poprzedni wynik tych samych autorów osiągał liniowy czas i pamięć, natomiast różnica pomiędzy dużym $O(n)$ tamtego algorytmu, i małym $o(n)$ dzisiejszego, jest na tyle istotna, że nowy algorytm został opublikowany.

Algorytm ten, tak jak i poprzedni, korzysta z tablicy Longest Previous Factor. Aby zrozumieć co to jest, weźmy taki najdłuższy czynnik słowa $w[1..i]$, równy m. Wtedy m musi być najdłuższym podsłowem słowa $w[1..i + |m| - 1]$, i to jego długość będzie występować w tej tablicy, na pozycji i-tej.

Gdy posiadamy tablicę LPF, wyznaczanie faktoryzacji nie jest trudne. Łatwo zauważyć, że "najdłuższy poprzedni czynnik", to dokładnie taki czynnik jakiego potrzebujemy do faktoryzacji. Wystarczy zatem przejść po tablicy LPF zwracając kolejne czynniki, pomijając przy tym czynniki pośrednie, występujące pomiędzy tymi z faktoryzacji, oraz zamieniając wszystkie zera na jedynki w tablicy LPF, ponieważ faktoryzacja nie zawiera słów pustych. **Algorithm 1** jest implementacją powyższej logiki.

Pozostaje wyznaczenie LPF. Do tego autorzy korzystają z tablic SA, i LCP – z uporządkowanej tablicy sufiksów i tablicy najdłuższych prefiksów między nimi.

Algorithm 1 lempel_ziv_factorization

Require: LPF, n
Ensure: LZ
  LZ ← [ ]
  pos ← 1
  while pos ⩽ n **do**
    push(max(1, LPF[pos]), LZ)
    pos ← pos + max(1, LPF[pos])
  end while