Lista 4, zadanie 6.4 - Tomasz Woszczyński

Treść: Ułóż algorytm, który dla danych podciągów x i y rozwiązuje problem znajdowania najdłuższego wspólnego podciągu niezawierającego podsłowa "egzamin".

Rozwiązanie: Rozważmy najpierw problem znajdowania zbioru najdłuższych wspólnych podciągów $x[1...i]$ oraz $y[1...j]$. Wykorzystamy do tego dwie funkcje opisane poniżej, pod implementacją:

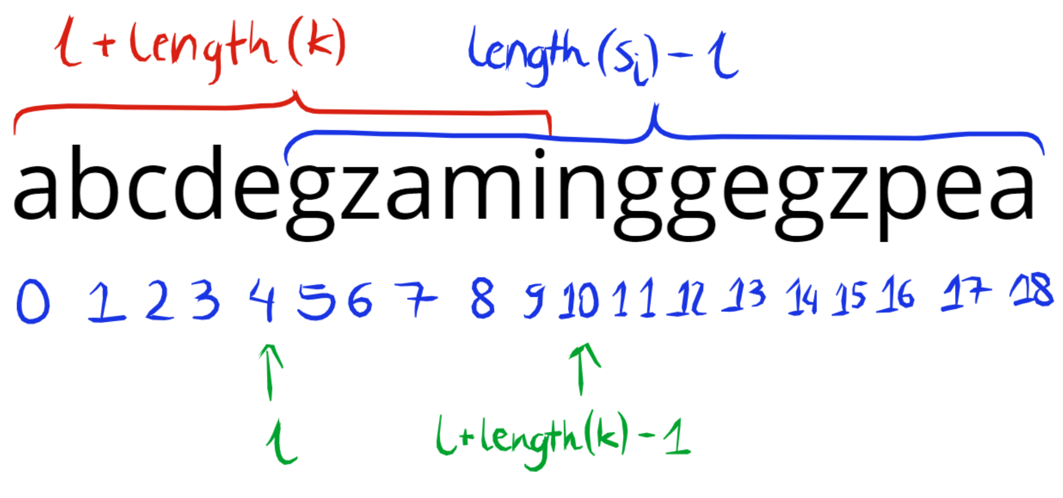
```
function LCS(x: string, y: string, m: length(x), n: length(y))
    for i ← 0 to m do
        for j ← 0 to n do
            if i = 0 or j = 0 then
                L[i][j] ← 0
            else if x[i − 1] = y[j − 1] then
                L[i][j] ← L[i − 1][j − 1] + 1
            else
                L[i][j] ← max(L[i − 1][j], L[i][j − 1])
            end if
        end for
    end for
    return L[m][n]
end function

function FindLCS(x: string, y: string, m: length(x), n: length(y))
    S ← ∅          ▷ do tego zbioru będziemy zapisywać kolejne najdłuższe podciągi
    if i = 0 or j = 0 then          ▷ w tym miejscu dochodzimy do końca stringa
        S ∪ ε
        return S
    end if
    if x[m − 1] = y[n − 1] then          ▷ ostatni znak x i y jest taki sam
        T ← FindLCS(x, y, m - 1, n - 1)          ▷ wywołujemy się rekurencyjnie
        for all str in T do          ▷ do wszystkich LCS dodajemy ostatni znak
            S ∪ (str + x[m − 1])
        end for
    else          ▷ ostatnie znaki podciągów x i y są różne
        if L[m − 1][n] ⩾ L[m][n − 1] then          ▷ gdy LCS jest "od góry" w tablicy
            S ← FindLCS(x, y, m - 1, n)          ▷ to idziemy do góry tablicy
        end if
        if L[m][n − 1] ⩾ L[m − 1][n] then          ▷ gdy LCS jest "od lewej" w tablicy
            T ← FindLCS(x, y, m, n - 1)          ▷ to idziemy w lewą stronę tablicy
            for all str in T do          ▷ przechodzimy po wszystkich znalezionych LCS
                S ∪ str          ▷ i dodajemy je do S
            end for
        end if
    end if
    return S          ▷ otrzymujemy zbiór wszystkich najdłuższych podciągów x i y
end function
```

Funkcja LCS znajduje długość najdłuższego podciągu ciągów x i y w czasie $O(m \cdot n)$, gdzie m i n to długości tych ciągów. Po wywołaniu tej funkcji otrzymujemy tablicę $L[m][n]$, dzięki której możemy w łatwy sposób rekurencyjnie odtworzyć wszystkie możliwe do uzyskania najdłuższe podciągi x i y - zajmuje się tym funkcja FindLCS.

Załóżmy, że w szukanym podciągu nie chcemy napotkać słowa k, będącego w naszym przypadku słowem "egzamin". Mając zbiór S wszystkich najdłuższych s_i podciągów, możemy rozpatrzyć trzy możliwe scenariusze.

1. podsłowo k jest na początku s_i,
2. podsłowo k jest na końcu s_i,
3. podsłowo k jest w środku s_i.

Pierwsze dwie możliwości są trywialne, gdyż po przejrzeniu całego stringa s_i wystarczy usunąć pierwszą lub ostatnią, w zależności od przypadku, literę, aby uzyskać najdłuższy podciąg bez podsłowa k. Wtedy w otrzymanym podciągu możemy znów natrafić na rozpatrzone już przypadki, lub na przypadek ostatni, który jest trochę bardziej złożony. Jeśli w trakcie przechodzenia przez s_i napotkamy podsłowo, którego chcemy się pozbyć, zapamiętujemy indeks l pierwszego znaku k w s_i, dzięki czemu będziemy w łatwy sposób mogli operować na tym podciągu. Następnie musimy sprawdzić, który z możliwych podciągów będzie dłuższy: ten bez pierwszej litery k, czy ten bez ostatniej litery k.



Naszym zadaniem jest teraz porównanie wartości $l + \text{length}(k)$ z $\text{length}(s_i) - l$, wybieramy większą z nich i zapamiętujemy długość tego podciągu, jak i sam podciąg. W taki sposób musimy porównać wszystkie podciągi $s_i \in S$, a następnie wybrać najdłuższy z nich (gdy jest ich kilka, wybieramy dowolny). Złożoność czasowa algorytmu znajdowania najdłuższego podciągu z danego zbioru S bez wybranego podsłowa k wynosi $O(|S| \cdot \text{length}(s_1))$.