Zadanie 6 d)

Dawid Żywczak, 05.06.2020

Aby znaleźć najdłuższy wspólny podciąg dwóch ciągów $x[1...m]$ oraz $y[1...n]$ musimy wypełnić tablicę T, gdzie $T[i][j]$ oznacza długość najdłuższego wspólnego podciągu prefiksów $x[1...i]$ i $y[1...j]$. Tablicę wypełniamy w następujący sposób:

1. Jeśli $x[i] \neq y[j]$ to $T[i][j] = max(T[i-1][j], T[i][j-1])$
2. wpp. $T[i-1][j-1]$+1

Zastanówmy się teraz jak podobną ideę wykorzystać do znalezienia najdłuższego wspólnego podciągu nie zawierającego podsłowa "egzamin"? Najpierw rozważmy w jaki sposób znaleźć najdłuższy wspólny podciąg zawierający podsłowo "egzamin".

Aby to zrobić rozważmy tabele $T_0, T_1, \ldots, T_8$, gdzie w tabeli T_0 obliczamy najdłuższy wspólny podciąg opisaną wyżej metodą, tabela T_1 oblicza najdłuższy wspólny podciąg kończący się na literze e, w T_2 obliczamy najdłuższy wspólny podciąg kończący się na słowem eg i tak aż do T_8, w którym obliczamy najdłuższy wspólny podciąg zawierający podsłowo egzamin. Tworząc tabelę T_1 rozważamy następujące przypadki:

1. Jeśli $x[i] \neq y[j]$ to $T_1[i][j] = max(T_1[i-1][j], T_1[i][j-1])$ (nie mamy zgodności, bierzemy długość ostatniego wspólnego podciągu)
2. Jeśli $x[i] = y[j] \neq e$ to $T_1[i][j] = T_1[i-1][j-1]$
3. Jeśli $x[i] = y[j] = e$ to $T_1[i][j] = T_0[i-1][j-1] + 1$ (ponieważ znaleźliśmy podciąg spełniający nasze założenia co do tabeli T_1)

Postępujemy podobnie dla tabel $T_2, \ldots, T_7$ zastępując literę e, literą która kończy słowo, mające znajdować się na końcu wspólnego podciągu, a T_0 tabelą o numerze o jeden mniejszym niż obecnie obliczana. Każdą z tabel obliczamy jednocześnie, to znaczy w trakcie obrotu pętli dla wartości x i y, obliczamy $T_0[x][y]$, $T_1[x][y]$, itd. Złożoność czasowa takiego rozwiązania to $O(n * m)$ (gdzie n i m to długości ciągów). Złożoność pamięciowa wyraża się jako $O(n * m)$ ze względu na trzymane tablice. Jak takie podejście wykorzystać w naszym zadaniu?

Podobnie zdefiniujmy sobie tabele: T_1 - najdłuższy wspólny podciąg nie zawierający podsłowa “egzamin” i niekończący się na e, T_2 - najdłuższy wspólny podciąg nie zawierający podsłowa “egzamin” i niekończący się na eg itd. aż do T_7 - najdłuższy wspólny podciąg nie zawierający podsłowa egzamin. Podobnie jak w przypadku wyżej, wszystkie tabele wypełniamy jednocześnie. W jaki sposób?

Dla T_1:

1. Jeśli $x[i] \neq y[j]$ to $T_1[i][j] = max(T_1[i-1][j], T_1[i][j-1])$ (nie mamy zgodności, bierzemy długość ostatniego wspólnego podciągu)
2. Jeśli $x[i] = y[j] = e$ to $T_1[i][j] = T_1[i-1][j-1]$ (nie możemy dołożyć e, co wynika z założeń tabeli T_1)
3. Jeśli $x[i] = y[j] = n$ to $T_1[i][j] = max(T_6[i-1][j-1] + 1, T_1[i-1][j-1])$ (możemy dołożyć do ciągu podciągu z T_6 ponieważ, mamy założenie, że nie kończy się na "egzami", zatem nie dostaniemy podsłowa "egzamin")
4. Jeśli $x[i] = y[j] \notin \{e, n\}$ $T_1[i][j] = T_7[i-1][j-1] + 1$

Rozważmy przypadki również dla T_2:

1. Jeśli $x[i] \neq y[j]$ to $T_2[i][j] = max(T_2[i-1][j], T_1[i][j-1])$

2. Jeśli $x[i] = y[j] = g$ to $T_2[i][j] = max(T_1[i-1][j-1] + 1, T_2[i-1][j-1])$ (możemy dołożyć g do T_1 ponieważ wiemy, że nie ma tam e na końcu, więc nie dostaniemy prefiksu słowa egzamin)
3. Jeśli $x[i] = y[j] = n$ to $T_2[i][j] = max(T_6[i-1][j-1] + 1, T_2[i-1][j-1])$ (możemy dołożyć do ciągu podciągu z T_6 ponieważ, mamy założenie, że nie kończy się na "egzami", zatem nie dostaniemy podsłowa "egzamin")
4. Jeśli $x[i] = y[j] \notin \{g, n\}$ $T_1[i][j] = T_7[i-1][j-1] + 1$

Analogiczne rozumowanie przeprowadzić można dla tabel $T_3, \ldots, T_6$. Przyjrzyjmy się jeszcze tabeli T_7.

1. Jeśli $x[i] \neq y[j]$ to $T_7[i][j] = max(T_7[i-1][j], T_7[i][j-1])$ (nie mamy zgodności, bierzemy długość ostatniego wspólnego podciągu)
2. Jeśli $x[i] = y[j] = n$ to $T_7[i][j] = max(T_7[i-1][j-1], T_6[i-1][j-1] + 1)$ (podobnie jak wyżej możemy bez obaw dołożyć n na koniec najdłuższego podciągu wyznaczonego w T_6)
3. Jeśli $x[i] = y[j] \neq n$ $T_7[i][j] = T_7[i-1][j-1] + 1$

Zatem rozwiązanie zadania to wypełnienie tablic zgodnie z podanym wyżej schematem oraz odtworzenie wyniku tak jak w algorytmie szukania LCS. Złożoność pamięciowa to $O(n * m)$, podobnie złożoność pamięciowa. Czy to w ogóle działa? Można się tutaj powołać na zasadę optymalności, ponieważ kolejno budujemy optymalne rozwiązania podproblemów od których bezpośrednio zależą kolejne wyniki, rozważając przy tym wszystkie możliwe stany, co ostatecznie prowadzi do znalezienia optymalnego rozwiązania.

(Algorytm można zmodyfikować, zwiększając ilość tabel odpowiednio względem długości podsłowa, którego najdłuższy wspólny podciąg ma nie zawierać. Oczywiście trzeba również zamienić warunki odnośnie liter i prefiksów, na te odpowiadające naszemu podsłowu. Wtedy złożoność pamięciowa i czasowa rozwiązania tego problemu dla podsłowa o długości k wynosi $O(k * m * n)$)