# LCS - NAJDŁUŻSZY WSPÓLNY PODCIĄG

30 stycznia 2006

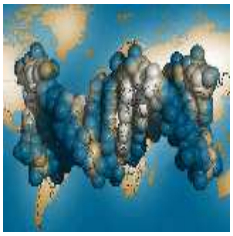
# Rozdział 1

# Wprowadzenie

Problem znalezienia Najdłuższego Wspólnego Podciągu (ang. LCS - Longest Common Subsequences) pojawia się w kilku praktycznych zastosowaniach :

- poszukiwanie "podobieństwa" dwóch pary nici DNA jako jednej z miar stopnia pokrewieństwa odpowiadających im organizmów
- wyłapywanie plagiatów muzycznych poprzez analizę ich linii melodycznych
- Porównywanie zawartości plików
- Odświeżanie tekstu w niektórych edytorach tekstów pracujących w powolnych sieciach terminali (no, ten problem jest już dzisiaj chyba mało aktualny)
- No i wreszcie w kryptografii

Najefektywniejsza metoda jego rozwiązania opiera się na programowaniu dynamicznym.



Rysunek 1.1: DNA a LCS

# Rozdział 2

# Najdłuższy Wspólny Podciąg - trochę teorii

Dowolny podciąg danego ciągu można otrzymać przez usunięcie z niego niektórych elementów (w skrajnym przypadku żadnego). Dla danego ciągu X=($x_1$,$x_2$,.....,$x_m$) inny ciąg Z=($z_1$,$z_1$,.....,$z_k$) jest **podciągiem** X, jeśli istnieje ściśle rosnący ciąg indeksów ($i_1$,$i_2$,....,$i_k$) ciągu X, taki, że dla każdego j-1,2,.....,k zachodzi równość $x_{ij}$=$z_j$. Na przykład Z=(B,C,D,B) jest podciągiem ciągu X=(A,B,C,B,D,A,B), a odpowiadający mu ciąg indeksów to (2,3,5,7). Mówimy, że ciąg Z jest **wspólnym podciągiem ciągów** X i Y, jeśli Z jest zarówno podciągiem X, jak i Y.

Na przykład dla danych ciągów X=(A,B,C,B,D,A,B) i Y=(B,D,C,A,B,A) ciąg (B,C,A) jest wspólnym podciągiem X i Y. Ciąg (B,C,A) nie jest jednak LCS ciągów X i Y, ponieważ ma długość 3, a ciąg(B,C,B,A) o długości 4 też jest wspólnym podciągiem ciągów X i Y. Podciągi (B,C,B,A) oraz (B,D,A,B) są **najdłuższymi wspólnymi podciągami ciągów** X i Y, ponieważ nie istnieje wspólny podciąg X i Y o długości większej od 4.

Danymi wejściowymi dla problemu LCS są dwa ciągi X=($x_1$,$x_2$,.....,$x_m$) oraz Y=($y_1$,$y_2$,.....,$y_n$). Rozwiązanie problemu LCS polega na znalezieniu ich najdłuższego wspólnego podciągu.

## 2.1 Charakterystyka LCS

Czas działania naiwnego rozwiązania problemu LCS polegającego na wygenerowaniu wszystkich podciągów X (jest ich $2^m$) jest wykładniczy. Przy dłuższych ciągach wymagania czasowe są nie do przyjęcia. Na szczęście ten problem ma **własność optymalnej podstruktury** i naturalna przestrzeń podproblemów odpowiada w tym przypadku parom "prefiksów" ciągów wejściowych. Konkretniej - dla danego ciągu X=($x_1$,$x_2$,.....,$x_m$) definiujemy i-ty **prefiks** ciągu X, dla i=0,1,.....,m jako $X_i$=($x_1$,$x_2$,.....,$x_i$). Przykładowo : jeśli X=(A,B,C,B,D,A,B), to $X_4$=(A,B,C,B), a $X_0$ jest ciągiem pustym.

**TWIERDZENIE nr 1** - o optymalnej podstrukturze LCS

Niech X=($x_1$,$x_2$,.....,$x_m$) oraz Y=($y_1$,$y_2$,.....,$y_n$)będą ciągami, a Z=($z_1$,$z_2$, .....,$z_k$) dowolnym LCS ciągów X i Y.

- 1. Jeśli $x_m$=$y_n$, to $z_k$=$x_m$=$y_n$ i $Z_{k-1}$ jest LCS ciągów $X_{m-1}$ i $Y_{n-1}$.
- 2. Jeśli $x_m \neq y_n$ i $z_k \neq x_m$, to Z jest LCS podciągów $X_{m-1}$ i Y.
- 3. Jeśli $x_m \neq y_n$ i $z_k \neq y_n$, to Z jest LCS ciągów X i $Y_{n-1}$.

**DOWÓD**

(1) (nie wprost) Jeśli $Z_k \neq X_m$, to moglibyśmy dołączyć $X_m$=$Y_n$ do Z, uzyskując wspólny podciąg X i Y o długości k+1, co przeczy założeniu, że Z jest najdłuższym podciągiem X i Y. Muszą więc zachodzić równości $Z_k$=$X_m$=$Y_n$. Prefiks $Z_{k-1}$ o długości k-1 jest z kolei wspólnym podciągiem ciągów $X_{m-1}$ i $Y_{n-1}$ Pokażemy teraz, że jest to LCS. Załóżmy przeciwnie, że istnieje wspólny podciąg W ciągów $X_{m-1}$ i $Y_{n-1}$ o długości większej niż k-1. Wtedy dodając do W element $X_m$=$Y_n$, otrzymujemy wspólny podciąg ciągów X i Y o długości większej niż k. Otrzymujemy sprzeczność.

(2) Jeśli $Z_k \neq$ m, to Z jest wspólnym podciągiem ciągów $X_{m-1}$ i Y. Gdyby istniał wspólny podciąg W ciągów $X_{m-1}$ i Y o długości większej niż k wtedy W byłby jednocześnie podciągiem $X_m$ i Y, co jest w sprzeczności z założeniem, że Z stanowi LCS ciągów X i Y.

(3) Dowód jest podobny do przypadku (2)

$\diamondsuit$

Charakterystyka powyższego twierdzenia pozwala stwierdzić, że problem LCS ma własność optymalnej podstruktury. Widać, że LCS dwóch ciągów zawiera LCS prefiksów tych ciągów.

## 2.2 Rozwiązanie rekurencyjne

Z twierdzenia nr 1 wynika, że szukając LCS ciągów X=($x_1$,$x_2$,.....,$x_m$) oraz Y=($y_1$,$y_2$,.....,$y_n$) należy rozpatrzyć jeden lub dwa problemy. Jeśli $X_m$=$Y_n$, to wystarczy znaleźć LCS ciągów $X_{m-1}$ i $Y_{n-1}$. Dołączając $X_m$=$Y_n$ do tego LCS otrzymujemy LCS ciągów X i Y. Jeśli natomiast $X_m \neq Y_n$, to musimy rozwiązać dwa podproblemy : znaleźć LCS ciągów $X_{m-1}$ i Y oraz znaleźć LCS ciągów X i $Y_{n-1}$. Dłuższy z nich stanowi LCS ciągów X i Y. Ponieważ te przypadki wyczerpują wszystkie możliwości, wiemy, że jedno z optymalnych rozwiązań musi być użyte w LCS ciągów X i Y. Oto rekurencyjny wzór na koszt optymalnego rozwiązania LCS. Niech c[i,j] będzie długością LCS ciągów $X_i$ i $Y_j$. Jeśli i=0 lub y=0, to jeden z podciągów ma długość 0 a w konsekwencji ich LCS ma także długość 0.

`c[i,j] = 0, jeśli i = 0 lub j = 0`
`c[i,j] = c[i-1,j-1]+1, jeśli i,j > 0 i` $X_i$ `=` $Y_j$
`c[i,j] = max(c[i,j-1],c[i-1,j]), jeśli i,j > 0 i` $X_i \neq Y_j$

Powyższa zależność pozwala nie rozpatrywać wszystkich podproblemów co jest niewątpliwym zyskiem czasowym. W przypadku, gdy $X_i$=$Y_j$ rozważamy podproblem znajdowania LCS ciągów $X_{i-1}$ i $Y_{j-1}$. W przeciwnym razie rozważamy dwa inne podproblemy znajdowania LCS ciągów $X_i$ i $Y_{i-1}$ oraz $X_{i-1}$ i Y.

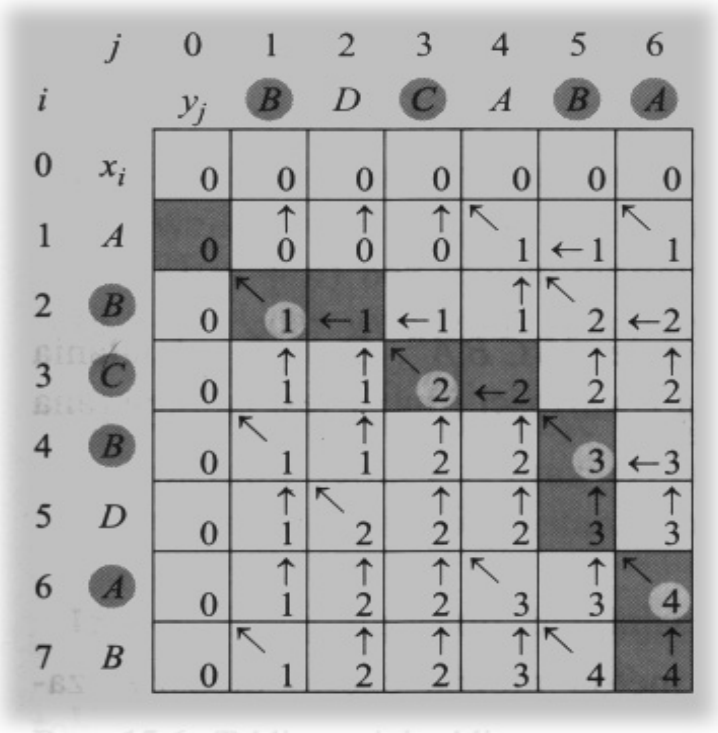
## 2.3 Obliczanie długości LCS

Aby obliczyć długość LCS musimy rozważyć (mn) problemów. Ponieważ jest to ilość nieduża, tak więc możemu użyć programowania dynamicznego do obliczania ich rozwiązań metodą wstępującą.

Procedura **LCS-Ile** dla danych ciągów X=($x_1$,$x_2$,.....,$x_m$) i Y=($y_1$,$y_2$,.....,$y_n$) oblicza wartości c[i,j] i zapamiętuje je w tablicy c[0..m,0..n]. Obliczenia te wykonywane są wzdłuż wierszy, a w każdym wierszu od strony lewej do prawej. Przy okazji tworzona jest tablica b[1..m,1..n] ułatwiająca konstrukcję otymalnego rozwiązania. Wartość b[i,j] wskazuje na pole w tablicy c odpowiadające wybranemu podczas obliczania c[i,j] optymalnemu rozwiązaniu problemu. Procedura **LCS-Ile** generuje tablice b i c, a pole c[m,n] zawiera długość LCS ciągów X i Y.

**LCS-Ile (X,Y)**

```
m=len(X)
n=len(Y)
for i = 1 to m
    do c[i,0] = 0
for j = 0 to n
    do c[0,j] = 0
for j = 0 to m
    do for j = 1 to n
        do if X_i = Y_i
            then c[i,j] = c[i-1,j-1]+1
                b[i,j] = "↖"
            else if c[i-1,j] >= c[i,j-1]
                    then c[i,j] = c[i-1,j]
                        b[i,j] = "↑"
                    else c[i,j] = c[i,j-1]
                        b[i,j] = "←"
return c i b
```




Rysunek 2.1: Konstrukcja LCS

Na rysunku 2.1 widać tablice obliczone przez procedurę **LCS-Ile** dla ciągów X=(A,B,C,B,D,A,B) oraz Y=(B,D,C,A,B,A) . Czas działania tej procedury wynosi $\vartheta$(mn), ponieważ obliczenie każdego z pól tablic c i b zajmuje $\vartheta(1)$ jednostek czasu.

## 2.4 Konstrukcja LCS

Korzystając z tablicy b obliczonej przez **LCS-Ile** możemy szybko obliczyć LCS ciągów X=($x_1$,$x_2$,.....,$x_m$) oraz Y=($y_1$,$y_2$,.....,$y_n$). Wystarczy rozpocząć od b[m,n], a następnie kierować się zgodnie ze strzałkami w tablicy b. Każda strzałka typu "$\nwarrow$" w polu b[i,j] oznacza, że $X_i$=$Y_j$ należy do LCS. Postępując w ten sposób, otrzymujemy elementy LCS w odwróconej kolejności.

# Rozdział 3

# Czy da się to policzyć sprytniej ?

Jak już wspomniałem przedstawiony algorytm zajmuje $\vartheta$(mn) pamięci i może obliczyć żądany LCS w czasie $\vartheta$(mn). Istnieje jednak jeszcze sprytniejszy algorytm, który potrzebuje jedynie $\vartheta$(m+n) pamięci. Jego autorem jest Hirschberg. Więcej szczegółów na jego temat można znaleźć na

*http://www.csse.monash.edu.au/~lloyd/tildeAlgDS/Dynamic/Hirsch*

# Rozdział 4

# Czyli zakończenie

Dokument ten powstał na bazie wykładu zamieszczonego we "Wprowadzeniu do algorytmów" autorstwa Thomasa H. Cormena, Charlsa E. Leisersona, Ronalda L. Rivesta oraz Clifforda Steina.

Bartek Nowowiejski