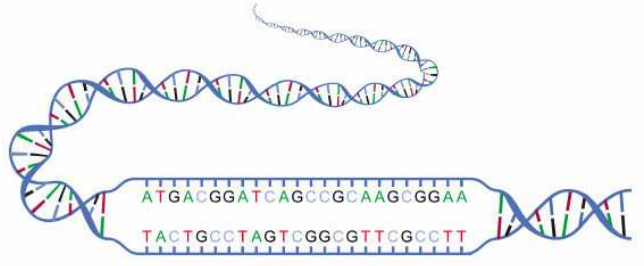
Najdaljši skupni podniz

Igor Pesek

Uvod

Genetika se ukvarja s preučevanjem dedovanja, lastnosti genov in DNK. Čeprav so celotni človekov genom uspeli določiti, ostaja še veliko vprašanj neodgovorjenih. V iskanju odgovorov raziskovalci večkrat primerjajo dva DNK niza, ki sta predstavljena s kombinacijo črk A, C, G, T (več v [1]). Zanima jih ujemanje nizov oziroma njihova podobnost. To slednje bo tema tega prispevka, kjer bomo ugotavljali podobnost dveh nizov z iskanjem najdaljšega skupnega podniza.

Definirajmo najprej osnovne pojme, ki jih bomo potrebovali. *Niz* S je urejen seznam znakov, ki so običajno neprekinjeno zapisani od leve proti desni. Niz S vsebuje *podniz* $S[i..j]$, ki se začne na indeksu i in konča na indeksu j. Iščemo takšen strnjeni podniz, ki je najdaljši in je vsebovan v nizih S in T dolžin m in n.



SLIKA 1.
Vijačnica z DNK nizi

Najdaljši skupni podniz

Najpreprostejša bi bila primerjava vseh možnih podnizov prve besede z vsemi podnizi druge besede. Ta pristop bi lahko opisali kot preiskovanje s surovo silo, ki pa je časovno zelo zahteven. Za ilustracijo si poglejmo iskanje podnizov v nizih $AAGGCT$ in $CCGCT$, kjer sta dolžini prvega niza $m = 6$ in drugega $n = 5$. Vseh podnizov prvega niza je $m^2 = 6^2 = 36$ in podnizov drugega niza $n^2 = 5^2 = 25$, kar pomeni, da bi morali primerjati $36 \cdot 25 = 900$ podnizov. Časovna zahtevnost takšnega pristopa je $\mathcal{O}(m^2 \cdot n^2)$. V nadaljevanju bomo predstavili algoritem, ki poišče najdaljši podniz s časovno zahtevnostjo $\mathcal{O}(m \cdot n)$, obstaja pa tudi algoritem, ki najde podniz v linearnem času.

Algoritem je nastal s pomočjo dinamičnega programiranja, vendar nekaterih podrobnosti ne bomo razložili, prav tako tudi ne bomo strogo dokazali, da algoritem deluje pravilno [2]. Oboje namreč presega okvir tega prispevka.

Zapišimo niza S in T, v katerih iščemo podniz, v tabelo L velikosti $(m+1) \times (n+1)$, kjer ima zaporedoma vsak znak niza S svoj stolpec tabele, vsak znak niza T pa svojo vrstico v tabeli. V tabelo smo dodali še eno vrstico in stolpec, ker nam bo to kasneje koristilo pri izračunih. Primer je prikazan v tabeli 1, kjer je S_i i-ti znak niza S in T_j j-ti znak niza T.

Sedaj tabelo L napolnimo po naslednjem pravilu:

- $$L(i,j) = \begin{cases} 0 & i = 0 \text{ ali } j = 0, \\ 0 & S[i] \neq T[j] \\ 1 + L(i-1, j-1) & S[i] = T[j]. \end{cases}$$

Razmislimo sedaj, zakaj takšno pravilo. Druga vrstica pravi, da ko sta znaka $S[i]$ in $T[j]$ različna, potem na tem mestu zagotovo ne more obstajati podniz. Ko pa sta znaka enaka, potem pogledamo mesto, kjer smo v obeh nizih primerjali njun predhodni znak, to je $S[i-1]$ in $T[j-1]$ oziroma v tabeli $L(i-1, j-1)$. Vrednost na tem mestu nam pove, koliko ujemanj po znakih je bilo do tega znaka. Če

		T_1	T_2	$\cdots$	T_j	$\cdots$	T_m
S_1							
S_2							
$\vdots$							
S_i							
$\vdots$							
S_n							

TABELA 1.
Primer tabele

je vrednost 0, potem ni bilo ujemanja, v nasprotnem primeru pa nam vrednost pove, koliko znakov se je že zaporedoma ujemalo do predhodnega znaka. Ker se tudi trenutna ujemata, vrednost povečamo za 1.

Kot neke vrste inicializacijo v tabelo najprej zapišemo 0 v prvo vrstico in prvi stolpec. Pomembno je, da pri izpolnjevanju tabele upoštevamo vrstni red izpolnjevanja od leve zgoraj proti desni spodaj, saj se nam v nasprotnem primeru lahko zgodi napaka, da vrednost ne obstaja. Razmislite, zakaj in kdaj bi se to zgodilo. Tretja vrstica je tudi razlog, zakaj imamo na začetku dodatni stolpec in vrstico. Vrednost $L(i-1, j-1)$ namreč dostopa do predhodne vrstice in stolpca, ki pa sploh ne bi nujno obstajala. Ko je tabela v celoti izpolnjena, poiščemo v tabeli največjo vrednost, ki nam pove dolžino najdaljšega podniza in je delna rešitev našega problema. Določitev podniza pa bomo razložili ob primeru v nadaljevanju.

Poglejmo si sedaj na primeru nizov $ACCA$ in $CACC$, kako je izpolnjena tabela L, kot je prikazana v tabeli (2).

Metoda »hitrega očesa« lahko brez tabele hitro ugotovi, da je najdaljši skupni podniz ACC dolžine 3, kar pa je tudi enako največji vrednosti v tabeli.

		A	C	C	A
	0	0	0	0	0
C	0	0	1	1	0
A	0	1	0	0	1
C	0	0	2	1	0
C	0	0	1	3	0

TABELA 2.
Z uporabo pravila napolnjena tabela

Kako ta podniz razberemo iz tabele? Poglejmo še enkrat tabelo (2). Če poznamo pravilo, kako se je tabela ustvarila, hitro opazimo, da za rekonstrukcijo podniza potrebujemo največjo vrednost k in indeksa i ali j v tabeli. Potem je najdaljši skupni podniz enak $S[i-k..i]$ ali $T[j-k..j]$. V našem primeru je $k = 3$ in $j = 3$ oziroma $T[j-k..j] = T[0..3] = "ACC"$.

Zapišimo sedaj najprej algoritem, ki nam izpolni tabelo in vrne najdaljši niz.

```
NAJDALJŠI SKUPNI PODNIZ (S,m,T,n)
    for i := 0 to m do L(i,0) := 0
    for j := 0 to n do L(0,j) := 0
    dolzina := 0
    resitev := (0,0)
    for i := 1 to m do
       for j := 1 to n do
          if S[i] != T[j] then
             L(i,j) := 0
          else
             L(i,j) := 1 + L(i-1,j-1)
             if L(i,j) > dolzina then
                dolzina := L(i,j)
                resitev = (i,j)
     j = resitev[1]
     return T[j-dolzina, j]
```

Naloga Z uporabo opisanega algoritma poiščite najdaljši podniz v nizih $AAACCACA$ in $ACACCCACCAA$. Rešitev naloge bo podana na koncu prispevka.