Meklēšana III

1. Uzdevuma nostādne

Šajā lekcijā tiks apskatīta simbolu virkņu aptuvenā salīdzināšana (angliski: sequence alignment). Šim uzdevumam ir vairāki varianti:

1. Dotas simbolu virknes T[0]...T[n-1] un T'[0]...T'[m-1]. Atļautas sekojošas operācijas:

   - Viena simbola aizstāšana ar citu simbolu;
   - Jebkura viena simbola izdzēšana;
   - Jauna simbola iespraušana patvaļīgā vietā.

   Kāds ir mazākais šādu operāciju skaits, ar ko var pārveidot T par T'?

2. Kāds ir mazākais šādu operāciju skaits, ar ko var pārveidot T par virkni, kas ir T' apakšvirkne?

3. Šiem diviem uzdevumiem ir iespējami sarežģītāki varianti:

   - Aizstāšanas operācijas izmaksas var būt atkarīgas no tā, kāds simbols tiek aizstāts ar kādu.
   - Var būt iespējams izdzēst vai iespraust k simbolu apakšvirkni vienā operācijā ar izmaksām f(k), kur f(k)<k (t.i. apakšvirknes iespraušana izmaksā lētāk nekā k atsevišķu simbolu iespraušana).

Motivācija. Uzdevums parādās, piemēram, algoritmiskajā bioloģijā, kur tiek pētīts, cik mutāciju nepieciešams, lai viena DNS (ģenētiskās informācijas) virkne pārvērstos par otru virkni.

Vienkāršības labad, mēs tālāk apskatīsim tikai pirmo uzdevuma paveidu.

2. Rekursīvs algoritms

Operāciju skaitu var izrēķināt šādi:

Ja T[n-1]=T'[m-1], tad Opt(T[0...n-1], T'[0...m-1]) = Opt(T[0...n-2], T'[0...m-2]).

Citādi, Opt(T[0...n-1], T'[0...m-1]) =
min(Opt(T[0...n-2], T'[0...m-2])+1,
Opt(T[0...n-2], T'[0...m-1])+1,
Opt(T[0...n-1], T'[0...m-2])+1).

Var diezgan vienkārši ar indukciju pēc virkņu garuma pierādīt, ka šis algoritms dod pareizu rezultātu, bet tā darbības laiks var būt eksponenciāls.

3. Dinamiskā programmēšana

Pseidokods:

```
for i=0 to max (n, m)
        M[0, i] = 0;
        M[i, 0] = 0;
for i = 1 to n
        for j = 1 to m
                if (A[i] = B[j]) then M[i, j] = M[i-1, j-1]
                else M[i, j] = min (M[i-1, j]+1, M[i, j-1]+1, M[i-1, j-1]+1);
```

Šī algoritma darbības laiks ir O(n▾m).

Darbības piemērs:

Ja T = atcaca un T’ = tcat, tad tabula aizpildās šādi:

	-	A	T	C	A	C	A
-	0	1	2	3	4	5	6
T	1	1	1	2	3	4	5
C	2	2	2	1	3	3	4
A	3	2	3	2	1	4	3
T	4	3	2	3	2	2	3

Pirmkārt, no šīs tabulas mēs uzzinām, ka minimālais operāciju skaits ir 3. Otrkārt, no tās var atjaunot optimālo operāciju virkni. To dara secībā no pēdējās operācijas uz pirmo:

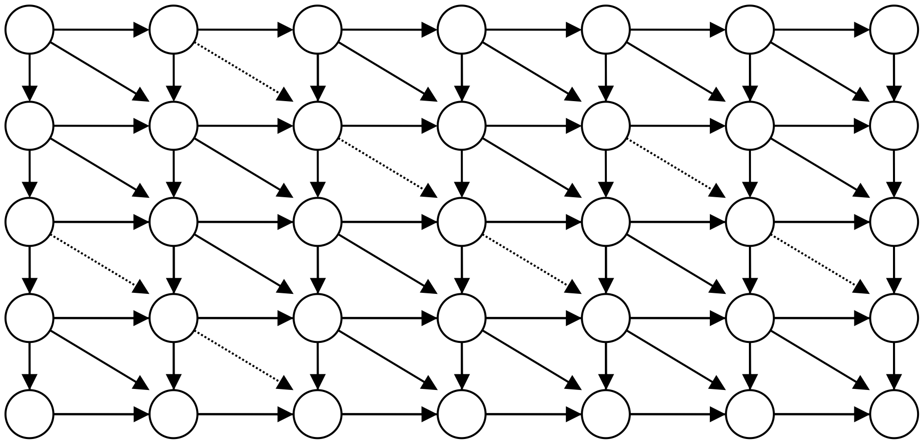
1. Apskatot labo apakšējo stūri (M[4, 6]) un tā blakus rūtiņas, secinām, ka M[4, 6]=3 iegūts, pieskaitot 1 pie M[4, 5]=2.
2. M[4, 5] = 2 savukārt iegūts pieskaitot 1 pie M[3, 4] = 1.
3. M[3, 4] = 1 iegūts no M[2, 3] =1, kas iegūts no M[1, 2] =1, kas iegūts no M[0, 1] =1.

Tas nozīmē, ka ATCACA no TCAT var iegūt šādi:

TCAT => ATCAT => ATCAC => ATCACA.

4. Ukkonena algoritms

Tabulu var reprezentēt grafa formā šādā veidā:



Grafa virsotnes (i, j) atbilst tabulas elementiem M[i, j]. Virsotņu pāri, kas sastāv no (i, j-1) un (i, j), vai no (i-1, j) un (i, j), ir savienoti ar šķautni garumā 1. Virsotņu pāri, kas sastāv no (i-1, j-1) un (i, j), ir savienoti ar šķautni garumā 0 (zīmējumā – pārtraukta līnija), ja T[i] = T'[j] un šķautni garumā 1 (nepārtraukta līnija), ja T[i] $\neq$ T'[j].

Tagad, M[i, j] ir vienāds ar īsākā ceļa no (0, 0) uz (m, n) garumu. To var izrēķināt, izmantojot Daisktras algoritmu priekš īsākā ceļa atrašanas (skat. nākošo sadaļu), laikā O(n D), kur D – ceļa garums. Ja minimālais operāciju skaits D ir būtiski mazāks par m un n (t.i. virknes ir līdzīgas), tad tas ir ātrāk nekā O(n m) priekš dinamiskās programmēšanas parastās realizācijas.

5. Daikstras algoritms

Daikstras algoritms atrod īsākos ceļus no vienas virsotnes grafā uz katru no pārējām virsotnēm. Šajā konkrētajā gadījumā Daikstras algoritms ir vienkāršaks nekā vispārējā gadījumā:

1. i = 0;
2. Atkārto:
    a. Izveido sarakstu S_i ar visām virsotnēm attālumā i no (0, 0);
    b. i = i +1;

    līdz (m, n) $\in$ S_{i-1}.

Sarakstus savukārt veido šādi:

<u>Saraksta S_0 izveide.</u>

1. i = 0;
2. Atkārto:
    a. Pievieno (i, i) sarakstam S_0;
    b. i = i+1;

    līdz brīdim kad T[i] $\neq$ T'[i].

<u>Saraksta S_i izveide (priekš i>0).</u>

Priekš katra iepriekšējā saraksta S_{i-1} elementa (j, k):

1. Ja (j+1, k) nav nevienā no sarakstiem S_0, S_1, ... S_{i-1}, tad:
    a. Pievieno (j+1, k) sarakstam S_i.
    b. Priekš katram r>0, kuram apakšvirkne T[j+2]...T[j+r+1] sakrīt ar T'[k+1]...T'[k+r], pievieno (j+r+1, k+r) sarakstam S_i.
2. Ja (j, k+1) nav nevienā no sarakstiem S_0, S_1, ... S_{i-1}, tad:
    a. Pievieno (j, k+1) sarakstam S_i.
    b. Priekš katram r>0, kuram apakšvirkne T[j+1]...T[j+r] sakrīt ar T'[k+2]...T'[k+r+1], pievieno (j+r, k+r+1) sarakstam S_i.
3. Ja (j+1, k+1) nav nevienā no sarakstiem S_0, S_1, ... S_{i-1}, tad:
    a. Pievieno (j+1, k+1) sarakstam S_i.
    b. Priekš katram r>0, kuram apakšvirkne T[j+2]...T[j+r+1] sakrīt ar T'[k+2]...T'[k+r+1], pievieno (j+r+1, k+r+1) sarakstam S_i.