KODIRANJE

Stiskanje z enakim številom bitov: vsak ASCII znak ima 8 bitov oz. 1 bajt. Genom je velik cca 3 milijarde znakov, to je 3×10^9 in posledično 2.4×10^{10} bitov (3×10^9 bajtov). (3GB) Primer, ko ima vsak nukleotid svoje bite, ki so njegova koda:

A: 1 0 0 0 C: 0 1 0 0 G: 0 0 1 0 T: 0 0 0 1 ⇒ pol manj kot prej (iz 8 v 4 bite), kar je 1.5×10^9 bajtov = 1.5 GB.

A: 0 0 C: 1 0 G: 0 1 T: 1 1 ⇒ 1 bajt na 4 znake ⇒ $3 \cdot 1/4 \times 10^9 = 0.75$ GB

↳ Primer zapisa: AGAGCCT = 00 01 00 01 10 10 11

Stiskanje z različnim številom bitov: Npr. dajmo vsakemu znaku neko kodo naključno:

A: 0 C: 0 1 G: 0 0 1 T: 0 0 0 ⇒ Problem tu je da ne vemo kdaj prebereno A (0) ali C (01) ali T (000). Ne vemo kdaj nehat brat kodo za znak. Problem je, da je A pripona vsem (vsi se začnejo z 0), ali pa npr. AC = 001 = G ?! Problem pri dešifriranju.

Iščemo tako stiskanje, da bo predponsko prosto.

HUFFMANOVO KODIRANJE: efektivno zakodira podatke glede na frekvence znakov, zato je odvisno od značilnosti besedila (ne zakodira vseh besedil enako). Znakom s manjšo frekvenco dodeli daljšo kodo in obratno.

PORAVNAVA ZAPOREDIJ

LEVENSTHEINOVA RAZDALJA: Število razlik med dvema nizoma oz. najmanjše število sprememb potrebnih, da enega pretvorimo v drugega. Npr. niza s = ATATAT- in t = -TATATATA:

s	A	T	A	T	A	T	A	T	-
t	-	T	A	T	A	T	A	T	A

⇒ d(s,t) = 2

HAMMINGOVA RAZDALJA: Število mest na katerih se niza razlikujeta. Za zgornji primer je to 2 (začetek in konec), za niza s' = ACT in t' = GTA pa je 3 (vse črke). Problem: Hammingova razdalja je lahko velika, čeprav sta si zaporedji zelo podobni:

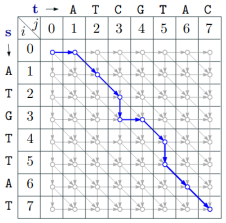
s	A	T	A	T	A	T
t	T	A	T	A	T	A

⇒ d(s,t) = 8

Leto	Avtorji	Algoritem	Pristop
1970	Needleman & Wunsch	Globalna poravnava	Optimalen
1981	Smith & Waterman	Lokalna poravnava	Optimalen
1988	Pearson & Lipman	FASTA	Hevristične
1990	Altschul idr.	BLAST	Hevristične

Hevristični algoritmi so hitrejši od **optimalnih**, toda ne izračunajo optimalne rešitve. Poravnave ovrednotimo tako, da **večje vrednosti ustrezajo boljšim poravnavam.**
Problem poravnave lahko modeliramo kot **iskanje najdaljše poti v acikličnem grafu** (predstavljaj si matriko) ⇒ vsaka pot v grafu od (0,0) do (i,j) predstavlja neko poravnavo podzaporedij s[1,..,i] in t[1,...,j] in obratno. Če povezave utežimo z ustreznimi nagradami in kaznimi, je največja vrednost poravnave zaporedij s in t (optimalna) enaka dolžini najdaljše poti.
Izračun poteka v treh korakih:

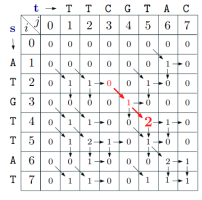
1. Inicializacija tabele
2. Izračun vrednosti v tabeli
3. Izračun ene (ali več) optimalnih poravnav zaporedij

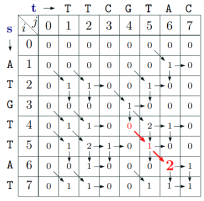


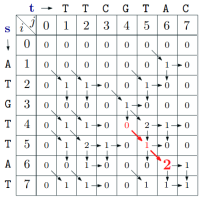
Algoritem:
Needleman - Wunsch

Motivacija za lokalne poravnave: globalne ne odkrijejo podobnih podzaporedij. Algoritem: Smith - Waterman.
Vsaka pot od elementa 0 do elementa z največjo vrednostjo (i,j) predstavlja optimalno lokalno poravnavo.

Primer:







GRUČENJE

Kako razdeliti množico eksperimentalno pridobljenih podatkov na skupine (gruče), tako da bodo podatki znotraj ene gruče čim bolj podobni drug drugemu in podatki v različnih gručah čim bolj različni? Primeri uporabe: identifikacija družin genov s podobnimi funkcijami, rekonstrukcija filogenetskih dreves, poravnave več zaporedij ...

Ugotavljanje funkcij genov poteka dandanes s pomočjo **mikromrež** (DNA microarrays), ki analizirajo ravni izražanja genov (količino proizvedene mRNA) pod različnimi pogoji in ob različnih časih.

Na ta način ugotovimo, kateri geni so ob katerem času aktivni v celici.

Podatki takega eksperimenta so zbrani v t.i.

matriki izražanja, $\mathcal{I}$,

ki ima n vrstic, po eno za vsak gen, in m stolpcev, po enega za vsako meritev.

Element matrike

$$\mathcal{I}(i,j)$$

predstavlja raven izražanja i-tega gena v j-tem poskusu.

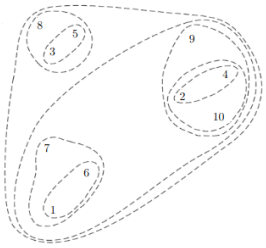
Celotna i-ta vrstica predstavlja t.i. **vzorec izražanja** i-tega gena.

Ideja: če imata dva gena podobna vzorca izražanja, potem sta zelo verjetno tudi biološko povezana (bodisi opravljata isto funkcijo ali sta prisotna pri istem biološkem procesu).

Gručenje je dobro, ko zagotavlja **Homogenost** (vrednosti D_{ij} morajo biti majhne za vsak par (i,j) iz iste gruče) ter **Separacijo** (vrednosti D_{ij} morajo biti velike za vsak par (i,j) iz različnih gruč).

Hierarhično gručenje: elemente organizira v drevo. Ne predstavlja ene razdelitve množice na gruče, temveč družino takih razdelitev.

→ Geni so v listih
→ Vsaka povezava ima neko dolžino
→ Razdalje med listi so sorazmerne z elementi matrike razdalj



Metode:

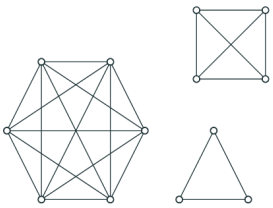
- **UPGMA** (Unweighted Pair Group Method with Arithmetic Averages)
- **NJ** (Neighbor Joining) – to metodo uporabljajo programi *Clustal* (vrsta programov za poravnavo več zaporedij)
- **Rekonstrukcija dreves iz aditivnih matrik**

  (ta metoda je direktno uporabna za konstrukcijo filogenetskih dreves, ob t.i. predpostavki *molekularne ure*)

Nehierarhično gručenje: npr. metoda voditeljev, ki zagotavlja homogenost vendar ne dobre separacije. Za te probleme ne obstajajo učinkoviti algoritmi, le hevristike – metoda poškodovanih klik,[1] CAST.[2]

(1)

- Klika v grafu je poln podgraf (poln graf je tak v katerem imata vsaki dve točki povezavo).
- Klični graf ima za povezane komponente same polne grafe. (ti ustrezajo gručam)
- V idealnem primeru dobimo klični graf iz katerega lahko razberemo gručenje, toda zaradi eksperimentalnih napak pogosto dobimo graf z dodatnimi povezavami ali z manj povezavami, zaradi česar ne dobimo kličnega grafa, therefore **problem poškodovanih klik**.
  ↳ Določi najmanjše število povezav, ki jih moramo odvzeti/dodati grafu, da dobimo klični graf. NP-težek problem, le hevristike.



(2)

- **CAST** izračuna gruče glede na razdaljni graf ter konstanto Θ. Iterativno izračuna particijo množice genov tako da na vsakem koraku najde gručo C, za katero velja:
  (i) $\exists i \notin C$ tako da je $d(i,C) < \Theta$ (bližnji gen izven gruče ne obstaja)
  (ii) $\exists i \in C$ tako da je $d(i,C) > \Theta$ (noben gen v gruči ni oddaljen)
- Nima zagotovljene kvalitetne rešitve ali zagotovljenega ustavitvenega pogoja, vender se v praksi kar dobro obnese.

PORAVNAVA VEČ ZAPOREDIJ