Bioinformatika 1

Samostojeći indeksi

Mirjana Domazet-Lošo
FER, 2020./2021.

Creative Commons Imenovanje-Nekomercijalno-Dijeli pod istim uvjetima 3.0

Što je indeks i čemu služi?

- količina digitalno dostupnih podataka rasla je eksponencijalno u posljednjem desetljeću
  - procjena za 2020. godinu: > 40 ZB (*zettabyte* = 10^{21} *bytes*) (International Data Corporation iView)
  - nakon 2020. očekuje se udvostručavanje otprilike svake 2 godine (Data Age 2025; Reinsel et al., 2017., IDC White Paper)

- kako omogućiti učinkovito pretraživanje teksta (ali i drugih digitalnih podataka)?
  - izgradnja indeksa
  - indeks = struktura podataka koja omogućuje učinkovit dohvat podataka

Ideja: indeks u knjizi

J. Pevsner, 2009. Bioinformatics and Functional Genomics, 2nd ed.

Subject Index

AAT program, 666
ABCD1 gene, 734, 857
Aberrations, chromosomal, 683, 863
Ab initio approaches:
  bacterial and archaeal genomes, 618
  gene-finding software, 666, 755
  genome analysis, 782
  prediction of protein structure, 450–451, 455
Absolute value, 346
Accepted point mutations, 58–63, 217. *See also* PAM matrices
Accession numbers, significance of, 27–28, 33, 37, 51–52, 90, 106, 128, 196, 198, 248
Accuracy, in microarray data analysis, 344–345
Acetylcholine, 768
Acetyltransferases, 39
ACOR, 883
Acrylamide gel, 382–383, 544
Actin, 224, 817
Actinobacteria, 600
Actinomycetes, 603
Acyl-CoA dehydrogenases, 718
Acyrthosiphon pisum, 609
ADAM20, 653
Adaptation, 216
ADE2, 468
Adenine, 64, 110, 242, 286, 545
Adenoma, 874
Adenosine:
  cyclic (cAMP), 398, 756
  monophosphate (AMP), 127
  triphosphate (ATP), 408, 545–546, 732, 858
Adenovirus, 570
Adenylate kinase, 223
Adhesin, 579
Adhesion molecules, 397, 579, 758
ADP, 224
Adrenoleukodystrophy, 718, 848, 850
Advanced database searching:
  characteristics of, 141–142
  hidden Markov models (HMM), 152, 156–161, 174
  gene discovery using BLAST, 169–173
  pitfalls of, 174
  position-specific iterated BLAST (PSI-BLAST), 145–156, 174
  PSSM, 146–153, 174
  rapid search of genomic DNA, 161–169
  SAM, 174
  specialized BLAST sites, 142–145
  web resources, 175
Aedes spp., 762, 764
Aeropyrum pernix, 533, 601, 608
Affinity chromatography, 499–500
Affymetrix, 315–317, 332, 333–335, 337, 343, 345–346
Affymetrix GeneChip, 343
Agarose gel, 293
Agglomerative hierarchical clustering, 355–357
Agilent, 317, 335
Agricultural issues, 541
Agrobacterium tumefaciens, 483
AIDS, 541, 579, 583, 585, 876
Ajellomyces capsulatus, 716
Akaike information criterion (AIC), 253
Alanine, 50–61, 63, 65, 68, 92, 94, 148, 154, 382, 428, 544
Albinism, 843, 847
Albumins, 305, 378, 380
Alcohol dehydrogenase, 702
Algae, 530, 746. *See also* Brown algae; Green algae
Algorithms, *see specific algorithms*
  advanced database searches, 141, 144
  applications, 5, 55, 161–162
  BLAST search, 115, 174
  defined, 55
AliBaba2, 670
ALIGN, 93
Align-m, 195
Alignment, significance of:
  advanced database searches, 141–142
  gapped, 120–121, 123
  pairwise, *see* Pairwise sequence alignment
  phylogenetic analysis, 60–61
  protein, 47–49
  score, 110
  ungapped, 119–120, 123
Alkaline phosphatase, 397
Alkaptonuria, 842–843
Alleles/allelic:
  functional genomics, 475, 478, 492, 508
  human disease, 863, 877
  human genome, 826
  single nucleotide polymorphisms (SNPs), 684–686
  variants, 27
AllGenes, 754
Allomyces macrogynus, 717
Allopolyploids, 753
Alpha crystallin A chain, 223
Alpha globin, 33

Bioinformatics and Functional Genomics, Second Edition. By Jonathan Pevsner
Copyright © 2009 John Wiley & Sons, Inc.

Tipovi indeksnih struktura

- potpuni indeks (eng. *full-text index*) – indeks koji omogućuje dohvat cijelog teksta ili bilo kojeg njegovog dijela
  - sufiksno stablo
  - sufiksno polje
  - → prostorna složenost teksta: $O(n \log |\Sigma|)$ bita
  - → prostorna složenost indeksa: $\Theta(n \log n)$ bita

- potpuni indeks ≠ samostojeći indeks (eng. *self-index*)
  - samostojeći indeks - proporcionalan veličini komprimiranog teksta

Samostojeći indeks

- samostojeći indeks (eng. *self-index*; *compressed self-index*)
  - indeksira tekst
  - zamjenjuje tekst, tj. indeks sam omogućuje pristup tekstu ili dijelovima teksta (podnizovima) nad kojima je indeks izgrađen
  - prostorna složenost: proporcionalna veličini komprimiranog teksta (eng. *compressed text*), tj. sublinearna složenost u odnosu na originalni tekst
  - problem: memorijsko zauzeće za izgradnju takvog indeksa može biti 5*n*-9*n* (Ferragina *et al*. 2008)
  - prvi takav indeks: FM-indeks (eng. *FM-index*; Ferragina i Manzini 2000.)

Podjela samostojećih indeksa

Podjela u 3 osnovne skupine (Ferragina *et al*., 2008):

1. FM-skupina indeksa
2. Indeksi temeljeni na *CSA* (*Compressed Suffix Arrays*)
3. LZ-skupina indeksa (Lempel-Zivovo sažimanje)

- teorijska prostorna složenost za niz S duljine n:

  $O(n \cdot H_k(S)) + o(n)$ bita
- za brojanje pojavljivanja podniza P u S:

  $O(|P| \cdot \log|\Sigma|)$
- pronalazak svakog pojavljivanja P u S:

  $O(\log^{1+\varepsilon}|S|)$

Entropija

- Primjer: bacanje kocke (6 mogućih događaja)
  - vrijedi: $P(X=1) + P(X=2) + P(X=3) + P(X=4) + P(X=5) + P(X=6) = 1$
  - promatramo 2 slučaja:

(i) nepristrana kocka:

$P(X=1) = P(X=2) = P(X=3) = P(X=4) = P(X=5) = P(X=6) = 1/6$

(ii) pristrana kocka – npr. uvijek dobijemo 6:

$P(X=6) = 1$, $P(X \neq 6) = 0$

- statistička entropija (Shannon, 1948): $H(S) = -\sum_i p_i \ln(p_i)$
  - H je maksimalna, ako su svi mogući ishodi jednako vjerojatni: $H = \ln 6 = 1.79$ (slučaj (i))
  - ako je moguć samo jedan ishod (slučaj (ii)): $H = 0$

Definicija nulte entropije za niz

- Neka je zadan niz *S* duljine *n*, i neka je n_i broj pojavljivanja znaka $i \in \Sigma$ u *S*.
- $H_0(S) = -\sum_i \frac{n_i}{n} log \frac{n_i}{n}, \quad n = \sum_i n_i$ (log → $\log_2$)

- <u>Primjer:</u>
  - S_1 = ACCA, $n = 4$, $n_A = n_C = 2$

    $H_0(S_1) = -(\frac{n_A}{n} log \frac{n_A}{n} + \frac{n_C}{n} log \frac{n_C}{n}) = -(\frac{2}{4} log \frac{2}{4} + \frac{2}{4} log \frac{2}{4}) = 1$

  - S_2 = ACCC, $n = 4$, $n_A = 1$, $n_C = 3$

    $H_0(S_2) = -(\frac{n_A}{n} log \frac{n_A}{n} + \frac{n_C}{n} log \frac{n_C}{n}) = -(\frac{1}{4} log \frac{1}{4} + \frac{3}{4} log \frac{3}{4}) = 0.811$

Definicija *k*-te entropije niza (1)

- Neka je zadan niz S duljine n, i neka je n_i broj pojavljivanja znaka $i \in \Sigma$ u S.
- Σ^k je skup svih nizova duljine k čiji su znakovi iz Σ.
- Neka je *con* podniz od S duljine k ($con \in \Sigma^k$): *kontekst* (eng. *context*). Neka je S^{con} niz koji sačinjavaju znakovi koji se pojavljuju u S iza *con* gledano slijeva nadesno.

Definicija *k*-te entropije niza (2)

- $H_k(S)$ je *k*-ti red entropije od *S* (eng. *the k-th order entropy of S*), $k \geq 0$
- $H_0(S) = -\sum_i \frac{n_i}{n} log \frac{n_i}{n}, \quad n = \sum_i n_i$
- $H_k(S) = \frac{1}{n} \sum_{con \in \Sigma^k} |S^{con}| H_0(S^{con})$

- Vrijedi:
  $0 \leq H_k(S) \leq H_{k-1}(S) \leq \ldots \leq H_1(S) \leq H_0(S) \leq \log|\Sigma|$

Definicija *k*-te entropije niza - primjer

- S = ACCA, $n = |S| = 4$
- con_1 = A, S^{con1} = C
- con_2 = C, S^{con2} = AC
- $\boldsymbol{H_0(S)} = -\sum_i \frac{n_i}{n} log \frac{n_i}{n} = -(\frac{n_A}{n} log \frac{n_A}{n} + \frac{n_C}{n} log \frac{n_C}{n}) = -2\,(\frac{2}{4} log \frac{2}{4}) = 1$
- $H_k(S) = \frac{1}{n} \sum_{con \in \Sigma^k} |S^{con}| H_0(S^{con})$
  - $H_0(S^{con1}) = -(\frac{1}{1} log \frac{1}{1}) = 0$
  - $H_0(S^{con2}) = -2\,(\frac{1}{2} log \frac{1}{2}) = 1$
- $\boldsymbol{H_1(S)} = 0.25 \cdot (|S^{con1}| \cdot H_0(S^{con1}) + |S^{con2}| \cdot H_0(S^{con2})) = 0.5$

FM-skupina indeksa

- prvi samostojeći indeks uopće: FM-indeks (Ferragina i Manzini 2000.; 2005.)
- temelji se na Burrows-Wheelerovoj transformaciji teksta (Burrows i Wheeler, 1994.)
- memorijski zahtjevi: proporcionalno *k*-toj entropiji teksta
- još neki indeksi iz ove skupine:
  - *Succinct Suffix Array* (SSA) (Mäkinen i Navarro, 2005.)
  - *Alphabet-Friendly FM-index* (AF) (Ferragina *et al*. 2004.)

Samostojeći indeksi temeljeni na *CSA*

- temelje se na komprimiranim/sažetim sufiksnim poljima (eng. *compressed suffix arrays*)
  - originalni CSA (Grossi i Vitter, 2000.) nije bio samostojeći indeks (memorijski zahtjevi: $O(n \cdot \log|\Sigma|)$)
  - Sadakane je unaprijedio inicijalno rješenje kako bi dobio samostojeći indeks (Sadakane, 2003.)
  - za sažimanje SA koristi se Ψ funkcija, koja koristi uočene pravilnosti teksta

Samostojeći indeksi temeljeni na LZ-sažimanju

- temeljeni na LZ78-sažimanju (Ziv i Lempel, 1978.)
- primjer indeksa iz ove skupine:
  - Navarro (2004.)

Table II. Ideal compressibility of our indexed texts. For every k-th order model, with $0 \leq k \leq 4$, we report the number of distinct contexts of length k, and the empirical entropy H_k, measured as number of bits per input symbol.

Text	$\log \sigma$	H_0	1st order		2nd order		3rd order		4th order	
			H_1	#	H_2	#	H_3	#	H_4	#
dna	4.000	1.974	1.930	16	1.920	152	1.916	683	1.910	2222
english	7.814	4.525	3.620	225	2.948	10829	2.422	102666	2.063	589230
pitches	7.055	5.633	4.734	133	4.139	10946	3.457	345078	2.334	3845792
proteins	4.644	4.201	4.178	25	4.156	607	4.066	11607	3.826	224132
sources	7.845	5.465	4.077	230	3.102	9525	2.337	253831	1.852	1719387
xml	6.585	5.257	3.480	96	2.170	7049	1.434	141736	1.045	907678

Table III. Real compressibility of our indexed texts, as achieved by the best-known compressors: gzip (option -9), bzip2 (option -9), and PPMDi (option -l 9).

Text	H_4	gzip	bzip2	PPMDi
dna	1.910	2.162	2.076	1.943
english	2.063	3.011	2.246	1.957
pitches	2.334	2.448	2.890	2.439
proteins	3.826	3.721	3.584	3.276
sources	1.852	1.790	1.493	1.016
xml	1.045	1.369	0.908	0.745

Ferragina et al. 2008. *Compressed Text Indexes: From Theory to Practice*

Tekst (200 MB)	**Veličina abecede**
dna	16
english	225
pitches	133
proteins	25
sources	230
xml	96

$$0 \leq H_k(T) \leq H_{k-1}(T) \leq \ldots \leq H_1(T) \leq H_0(T) \leq \log \sigma$$

Usporedba samostojećih indeksa (1)

- analiza koji su proveli Ferragina, González, Navarro i Venturini (2008):
  - usporedba sufiksnog polja i predstavnika svake od skupina samostojećih indeksa
  - promatrane su dvije tipične operacije:
  1. pronalaženje broja pojavljivanja uzorka *P* u tekstu *S*
  2. dohvaćanje svih pojavljivanja uzorka *P* u tekstu *S*

Usporedba samostojećih indeksa (2)

- analiza koji su proveli Ferragina, González, Navarro i Venturini (2008):
  - brzina (traženje uzorka u nizu) - samostojeći indeksi su sporiji 1-3 reda veličine od SA
  - memorijsko zauzeće samostojećih indeksa: može biti i do reda veličine manje od SA
    - usporedba: SA + tekst → 5n okteta, $n = |S|$
  - FM- i CSA-temeljeni indeksi zahtijevaju izgradnju SA polja kako bi se izgradio sam indeks, a LZ-indeks zahtijeva još neke pomoćne strukture
  - za izgradnju samostojećih indeksa potrebno 5n - 9n okteta

Table V. Time and peak of main memory usage required to build the various indexes over the 200 MB file `english`. The indexes are built using the default value for the locate tradeoff (that is, $s_A = 64$ for `AF-index` and `SSA`; $s_A = 64$ and $s_\Psi = 128$ for `CSA`; and $\epsilon = \frac{1}{4}$ for the `LZ-index`).

Index	Build Time (sec)	Main Memory Usage (MB)
`AF-index`	772	1,751
`CSA`	233	1,801
`LZ-index`	198	1,037
`SSA`	217	1,251

Table VI. Experiments on the counting of pattern occurrences. Time is measured in microseconds per pattern symbol. The space usage is expressed as a fraction of the original text size. We put in boldface those results that lie within 10% of the best space/time tradeoffs.

	`SSA`		`AF-index`		`CSA`		`LZ-index`		`plain SA`	
Text	Time	Space	Time	Space	Time	Space	Time	Space	Time	Space
`dna`	**0.956**	**0.29**	1.914	**0.28**	5.220	0.46	43.896	0.93	0.542	5
`english`	**2.147**	0.60	2.694	**0.42**	4.758	**0.44**	68.774	1.27	0.512	5
`pitches`	**2.195**	0.74	2.921	**0.66**	3.423	**0.63**	55.314	1.95	0.363	5
`proteins`	**1.905**	**0.56**	3.082	**0.56**	6.477	0.67	47.030	1.81	0.479	5
`sources`	**2.635**	0.72	2.946	0.49	4.345	**0.38**	162.444	1.27	0.499	5
`xml`	2.764	0.69	**2.256**	0.34	4.321	**0.29**	306.711	0.71	0.605	5

Ferragina *et al.* 2008. *Compressed Text Indexes: From Theory to Practice*

FM-indeks

- **pretraživanje unatrag** (eng. ***backwards search***)
- temelji se na podatkovnim strukturama SA i BWT (eng. *Burrows-Wheeler Transform* ili *block-sorting compression;* Burrows & Wheeler; 1994.)
  - BWT - koristi se kod sažimanja podataka, npr. bzip2 (Seward, 2007)
  - sublinearno memorijsko zauzeće:
    $O(nH_k(S)) + o(n)$ bita za niz S duljine n
  - $H_k(S)$ je k-ti red entropije od S (eng. *the k-th order entropy*)

BWT (1)

Izgradnja niza *BWT* rotacijom niza *S* = ACCA$. $ je abecedno najmanji znak.

i	Cikličke rotacije (CR) niza *S*	Abecedno poredane cikličke rotacije niza *S*	*B*[*i*]
0	ACCA$	$ACC**A**	A
1	CCA$A	A$AC**C**	C
2	CA$AC	ACCA**$**	$
3	A$ACC	CA$A**C**	C
4	$ACCA	CCA$**A**	A

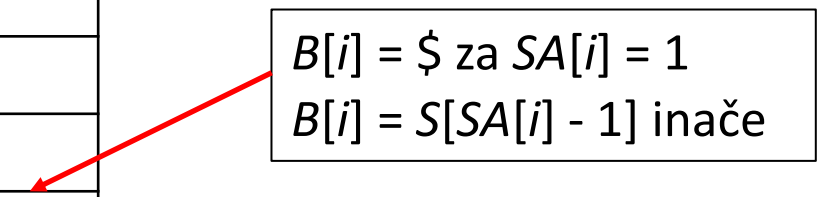
BWT (2)

BWT(*Burrows-Wheeler Transform*):

- određuje se u $O(n)$ vremenu iz *SA*
- memorijsko zauzeće: n okteta

Konstrukcija niza *BWT* za niz *S* = ACCA$ ($ je abecedno najmanji znak) korištenjem SA = [5, 4, 1, 3, 2]

i	SA[*i*]	Abecedno poredani sufiksi od *S*	*B*[*i*]
1	5	$	A
2	4	A$	C
3	1	ACCA$	$
4	3	CA$	C
5	2	CCA$	A

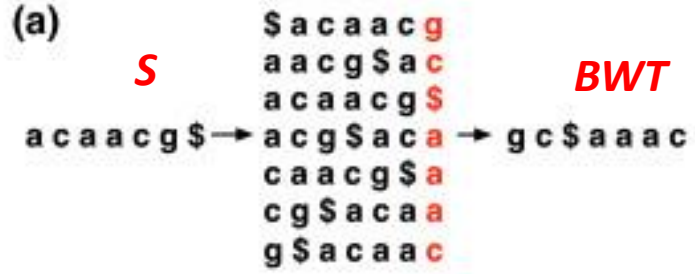


Reverzibilnost BWT-a

LF-mapiranje (eng. *LF-mapping*; *Last-to-Front mapping; Last-to-First mapping*)

i-ta pojava znaka X u zadnjem stupcu (*L*) odgovara *i*-toj pojavi znaka X u prvom stupcu (*F*), tj. *BWT*[*i*] se nalazi u stupcu *F* na mjestu *LF*[*i*].

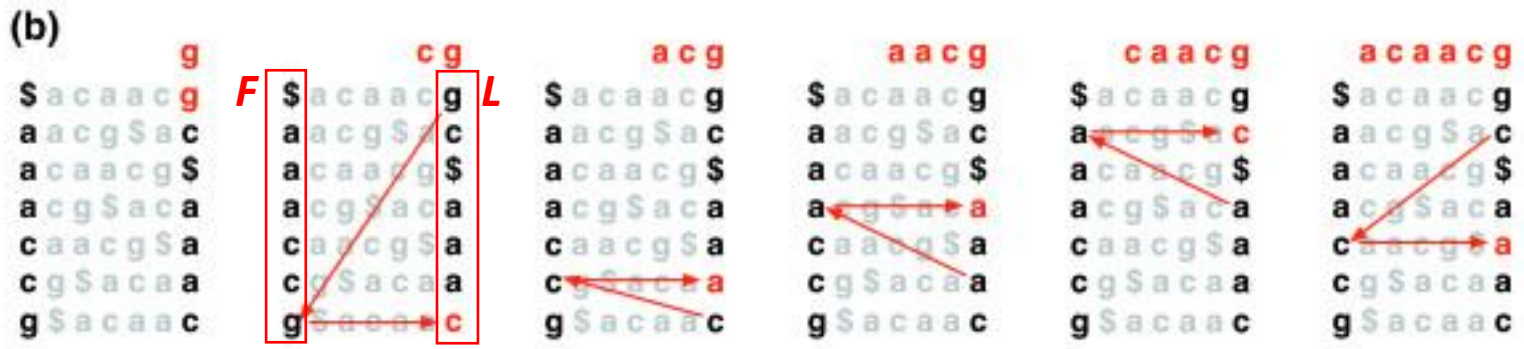
Primjer: prvi znak **g** u stupcu *L* se mapira na prvi znak **g** u stupcu *F* (vidjeti sliku).



***C*[*c*]** je broj znakova u *S*[1, |*S*|-1] koji su abecedno prije *c* (uključujući ponavljanja znakova).

***Occ*(*c*, *i*)** je broj pojavljivanja znaka *c* u *BWT*[1, *i*].

***LF*[*i*]** = *C*[*BWT*[*i*]] + *Occ*(*BWT*[*i*], *i*)



Langmead *et al*. 2009. *Ultrafast and memory-efficient alignment of short DNA sequences to the human genome.*

Interval sufiksnog polja

- Neka je zadan niz S.
- Sva pojavljivanja podniza P u S određena su intervalom sufiksnog polja $[L_P, R_P]$:

  L_P = **min** {k: P je prefiks sufiksa $s_{SA[k]}$}

  R_P = **max** {k: P je prefiks sufiksa $s_{SA[k]}$}

- Poseban slučaj:
  ako je $P = \emptyset$, onda je $[L_P, R_P] = [1, |S|]$

Primjer: S = ACCA\$, P = A → $[L_P, R_P] = [2, 3]$

i	SA[*i*]	*S*[SA[*i*], *n*]
1	5	\$
2	4	**A**\$
3	1	**A**CCA\$
4	3	CA\$
5	2	CCA\$

Pretraživanje unatrag (1)

Pretraživanje unatrag (eng. *backward search*)

- Neka je B = BWT(S), a c ϵ Σ.

- Tada su definirane funkcije C i Occ na sljedeći način:
  - ***C*[*c*]** je broj znakova u S[1, |S|-1] koji su abecedno prije c (uključujući i ponavljanja znakova)
  - ***Occ*(*c*, *i*)** je broj pojavljivanja znaka c u B[1, i], gdje i = 1, .., |S|

Pretraživanje unatrag (2)

- Neka je *P* neki podniz koji tražimo u *S* i c ϵ Σ.
- Tada je *P* sufiks podniza *cP* kojeg tražimo u *S* nakon što smo pronašli *P*.
- Vrijedi sljedeće (Ferragina i Manzini, 2000):

  $\boldsymbol{L_{cP}} = C[c] + Occ(c, L_P - 1) + 1$

  $\boldsymbol{R_{cP}} = C[c] + Occ(c, R_P)$

  $Occ(c, i)$ = broj pojavljivanja *c* u $B[1, i]$; $i = 1, .., |S|$

- *BW_Count* algoritam (Ferragina i Manzini, 2000)
  - kreće s pretraživanjem počevši od zadnjeg znaka od *P*
  - ako *P* postoji u *S*, algoritam vraća broj pojavljivanja od *P*, a 0 inače

Algoritam *BW_Count*

```
Ulaz: C, Occ, P
Izlaz: broj pojavljivanja P u S
i = |P| /* kreće se od zadnjeg mjesta u P */
c = P[|P|] /* zadnji znak u P */
L_P = C[c] + 1
R_P = C[csljed] /* csljed je znak nakon c u Σ */
dok ((L_P < R_P) i (i ≥ 2)) ponavljaj
   c = P[i - 1]; /* sljedeći znak iz P */
   L_P = C[c] + Occ(c, L_P - 1) + 1
   R_P = C[c] + Occ(c, R_P)
   i = i - 1
ako (R_P < L_P) onda vrati 0
inače vrati (R_P - L_P + 1)
```

C[*c*] → broj znakova koji su abecedno prije *c*

Occ(*c*, *i*) → broj pojavljivanja *c* u *B*[1, *i*]

Algoritam *BW_Count* – primjer (1)

- Neka je zadan *S* = ACCA$.
  Postoji li podniz *P* = CA u *S*?
- Rješenje:
  P je *S*[3, 4] (sufiks s_3 sadrži prefiks koji je jednak *P*-u).
- 1. korak: određujemo polje *C*
  C['A'] = 1
  C['C'] = 3

C[*c*] je broj znakova u *S* koji su abecedno prije *c*

- 2. korak: određujemo *Occ*

Occ('A', 1) = 1; *Occ*('A', 2) = 1; itd.
Occ('C', 1) = 0; *Occ*('C', 2) = 1; itd.

Occ(*c*, *i*) = broj pojavljivanja *c* u *B*[1, *i*]

	1	2	3	4	5
B	A	C	$	C	A

Algoritam *BW_Count* – primjer (2)

- Neka je zadan S = ACCA\$ (B = AC\$CA). Postoji li podniz P = CA u S?
- Traženje P u S počinjemo s 'A':

  C['A'] = 1

  $L_P = C['A'] + 1 = \mathbf{2}$

  $R_P = C$[csljed] = C['C'] = **3**
- Nastavljamo pretraživanje s 'C':

  $L_P = C['C'] + Occ('C', L_P - 1) + 1 =$

  $3 + 0 + 1 = \mathbf{4}$

  $R_P = C['C'] + Occ('C', R_P) = 3 + 1 = \mathbf{4}$
- Funkcija vraća vrijednost:

  $R_P - L_P + 1 = 4 - 4 + 1 = 1$

  P se pojavljuje samo jedanput u S.

(i) $\boldsymbol{L_P} = C[c] + 1$; $\boldsymbol{R_P} = C[csljed]$

(ii) $\boldsymbol{L_P} = C['C'] + Occ('C', L_P - 1) + 1$

$\boldsymbol{R_P} = C['C'] + Occ('C', R_P)$

i	SA[*i*]	*S*[SA[*i*], *n*]
1	5	\$ACC**A**
2	4	**A**\$AC**C**
3	1	**A**CCA**\$**
4	3	CA\$A**C**
5	2	CCA\$**A**

Occ(*c*, *i*) = broj pojavljivanja *c* u *B*[1, *i*]

Algoritam *BW_Count* – primjer (3)

i-ta pojava znaka X u zadnjem stupcu odgovara *i*-toj pojavi znaka X u prvom stupcu, tj. *BWT*[*i*] se nalazi u prvom stupcu (F) na mjestu *LF*[*i*]; $\boldsymbol{LF[i]} = C[BWT[i]] + Occ(BWT[i], i)$

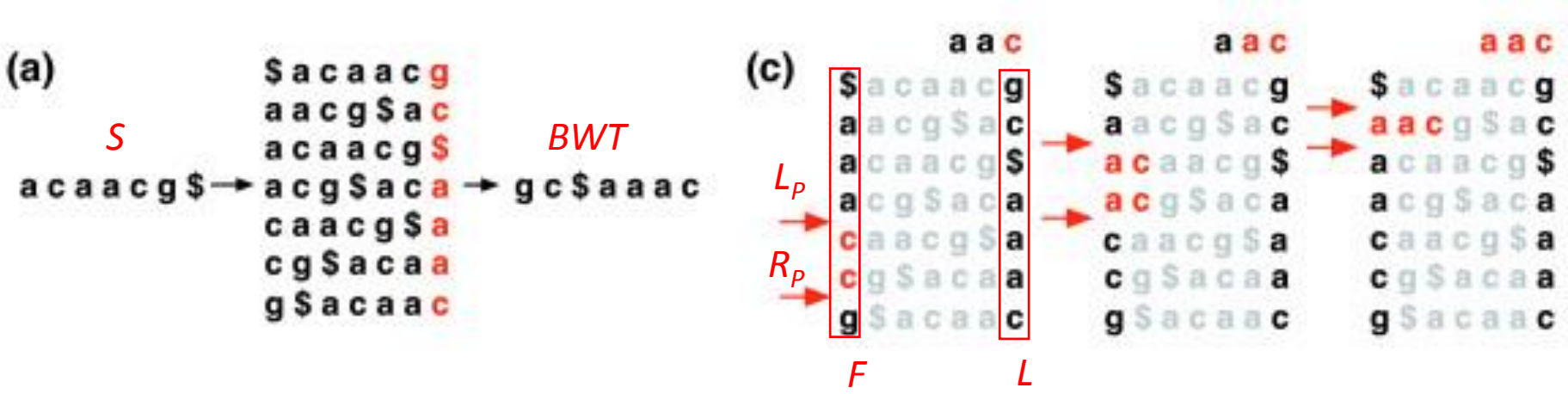
***C*(*c*)** je broj znakova u *S* koji su abecedno prije *c* (uključujući i ponavljanja znakova)

***Occ*(*c*, *i*)** je broj pojavljivanja znaka *c* u *BWT*[1, *i*]

(i) $\boldsymbol{L_P} = C[c] + 1$; $\boldsymbol{R_P} = C[csljed]$

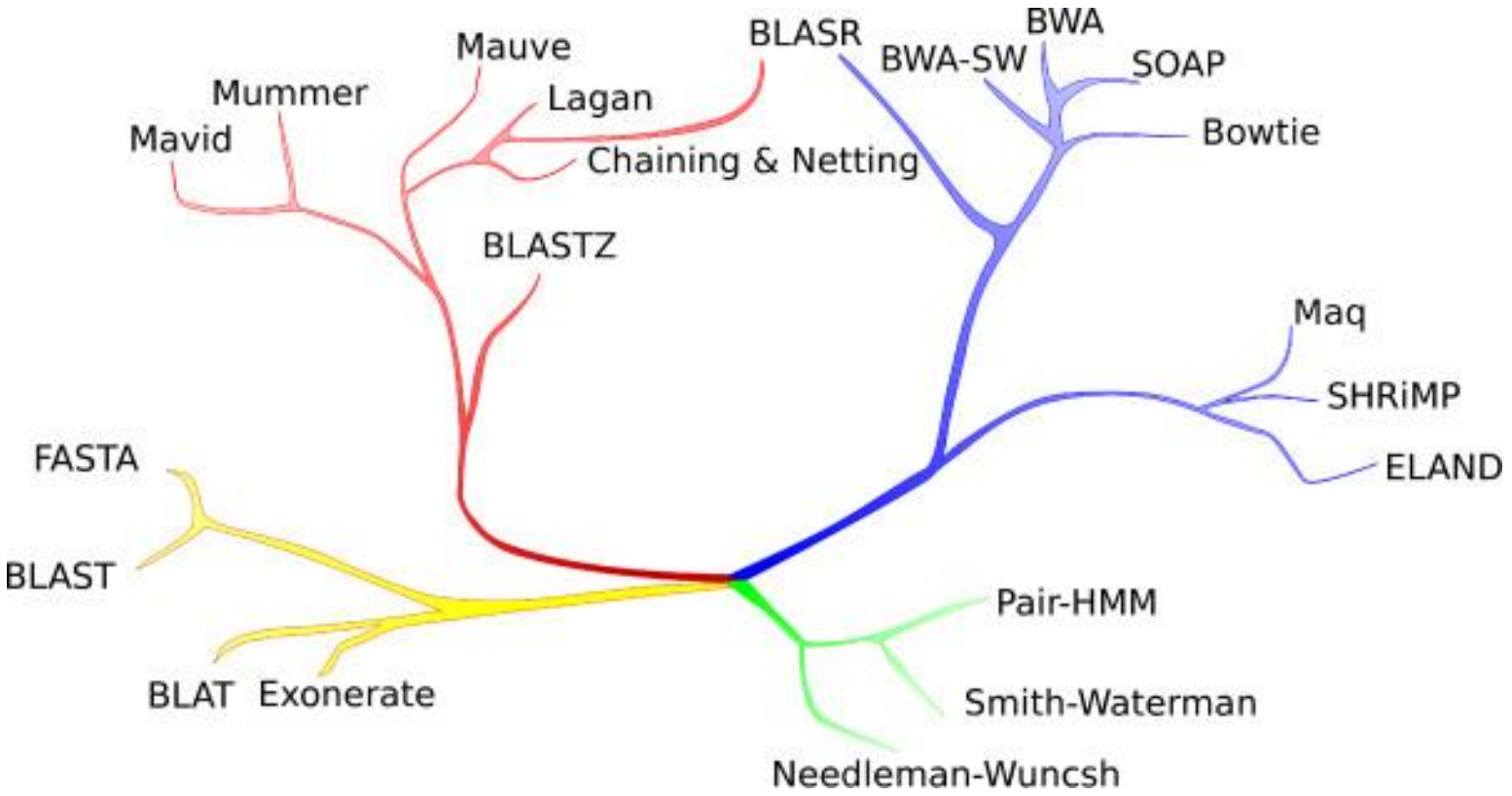
(ii) $\boldsymbol{L_P} = C[c] + Occ(c, L_P - 1) + 1$

$\boldsymbol{R_P} = C[c] + Occ(c, R_P)$



Langmead *et al*. 2009. *Ultrafast and memory-efficient alignment of short DNA sequences to the human genome.*

Pregled različitih metoda poravnavanja



Chaisson and Tesler, *BMC Bioinformatics* 2012, **13**:238