Outline

Řetězce a algoritmy na řetězcích

- Základní pojmy
- Základní algoritmy
- Algoritmus využívající analýzu hledaného motivu
- Algoritmus využívající nalýzu prohledávaného řetězce

Hledání opakování

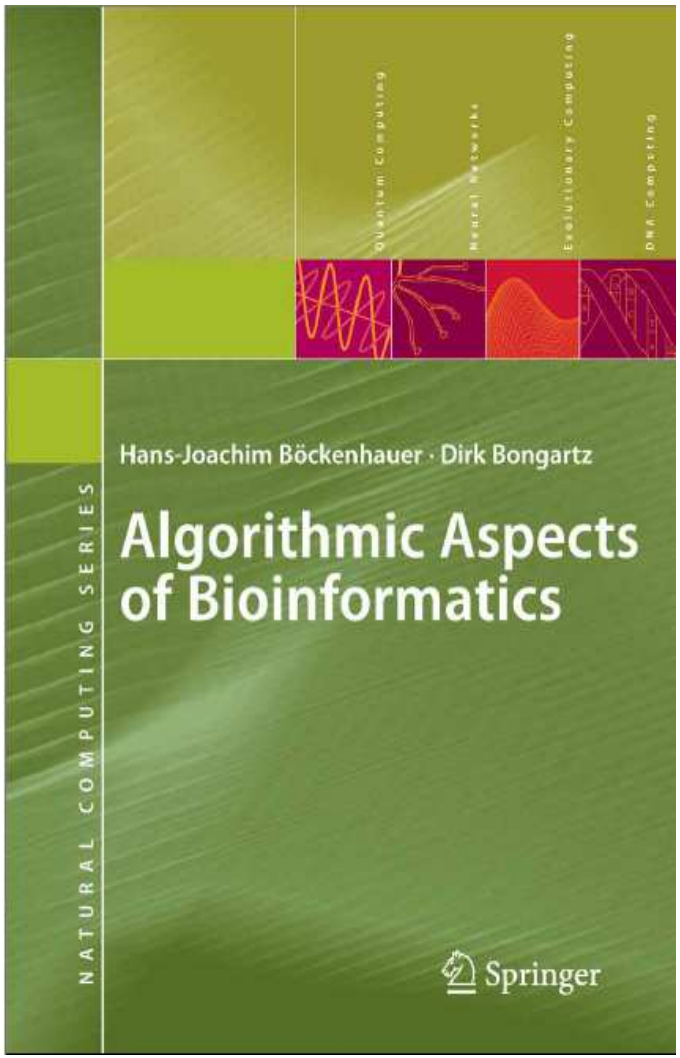
- Tandemové opakování
- Palindromy

Srovnávání dvou sekvencí

- DP - Needleman-Wunsch
- Vylepšení pro maximálně k chyb
- video HHMI

Řetězce a algoritmy na řetězcích
Základní pojmy
Základní algoritmy
Algoritmus využívající analýzu hledaného motivu
Algoritmus využívající nalýzu prohledávaného řetězce
Hledání opakování
Tandemové opakování
Palindromy
Srovnávání dvou sekvencí
Vylepšení pro maximálně k chyb
Příště
video HHMI

Podrobné informace



Řetězce a algoritmy na řetězcích
Základní pojmy
Základní algoritmy
Algoritmus využívající analýzu hledaného motivu
Algoritmus využívající nalýzu prohledávaného řetězce
Hledání opakování
Tandemové opakování
Palindromy
Srovnávání dvou sekvencí
Vylepšení pro maximálně k chyb
Příště
video HHMI

Základní pojmy

Řetězce a algoritmy na řetězcích
Základní pojmy
Základní algoritmy
Algoritmus využívající analýzu hledaného motivu
Algoritmus využívající nalýzu prohledávaného řetězce
Hledání opakování
Tandemové opakování
Palindromy
Srovnávání dvou sekvencí
Vylepšení pro maximálně k chyb
Příště
video HHMI

abeceda	$\{\epsilon, a, c, g, t\}$
podřetězec	aag**gtacg**cgt
prefix	**gtacg**cgtgtt
suffix	cgtat**gtacg**

Základní operace

Řetězce a algoritmy na řetězcích
Základní pojmy
Základní algoritmy
Algoritmus využívající analýzu hledaného motivu
Algoritmus využívající nalýzu prohledávaného řetězce
Hledání opakování
Tandemové opakování
Palindromy
Srovnávání dvou sekvencí
Vylepšení pro maximálně k chyb
Příště
video HHMI

konkatenace	`x=cggat y=att x.y=cggatatt`
průnik	`x=cggat y=att Over(x,y)=at`
sjednocení	`x=cggat y=att ⟨x,y⟩=cggatt`

Výskyt sekvenčních motivů v databázích

Řetězce a algoritmy na řetězcích
Základní pojmy
Základní algoritmy
Algoritmus využívající analýzu hledaného motivu
Algoritmus využívající nalýzu prohledávaného řetězce
Hledání opakování
Tandemové opakování
Palindromy
Srovnávání dvou sekvencí
Vylepšení pro maximálně k chyb
Příště
video HHMI

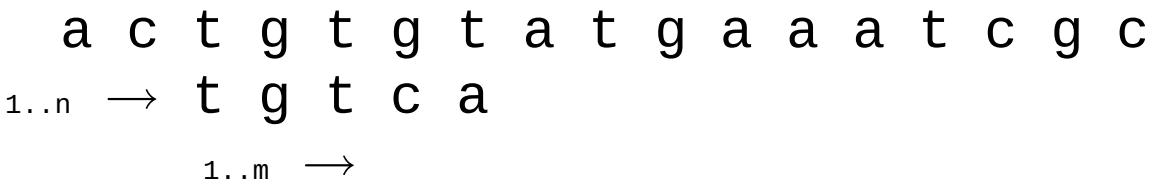
Cílem je zjistit všechny pozice delšího řetězce, na kterých se vyskytuje kratší řetězec

- přesný výskyt
- přibližný výskyt

řetězec t dlouhý (n), např genomová sekvence

motiv p krátký (m), např `cgcggctggtggctcg`

Naivní algoritmus



Složitost: O(mn)

Řetězce a algoritmy na řetězcích
Základní pojmy
Základní algoritmy
Algoritmus využívající analýzu hledaného motivu
Algoritmus využívající nalýzu prohledávaného řetězce
Hledání opakování
Tandemové opakování
Palindromy
Srovnávání dvou sekvencí
Vylepšení pro maximálně k chyb
Příště
video HHMI

Boyer-Moore

Řetězce a algoritmy na řetězcích
Základní pojmy
Základní algoritmy
Algoritmus využívající analýzu hledaného motivu
Algoritmus využívající nalýzu prohledávaného řetězce
Hledání opakování
Tandemové opakování
Palindromy
Srovnávání dvou sekvencí
Vylepšení pro maximálně k chyb
Příště
video HHMI

```
a c t g t g t a t g a a a t c g c
    → g a t c a t
              × ↑ ↑ ←
```

máme v motivu další `t`*?*

```
a c t g t g t a t g a a a t c g c
  +1 → g a t c a t
```

kde máme v motivu další výskyt suffixu `at`*?*

```
a c t g t g t a t g a a a t c g c
      +3 → g a t c a t
```

Realizujeme krok, který je větší

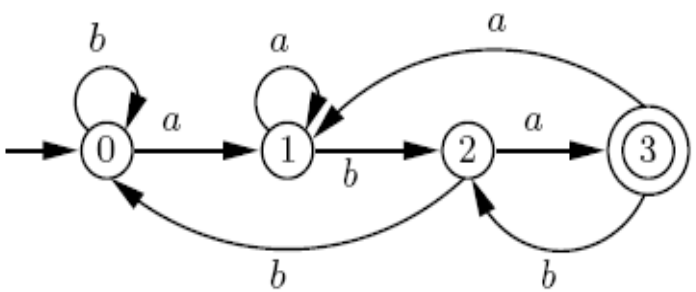
Složitost

konstrukce: O(‖*abeceda*‖.m)

hledání: O(mn) (v praxi ale blíže k O(n))

Automat pro hledání řetězce aba

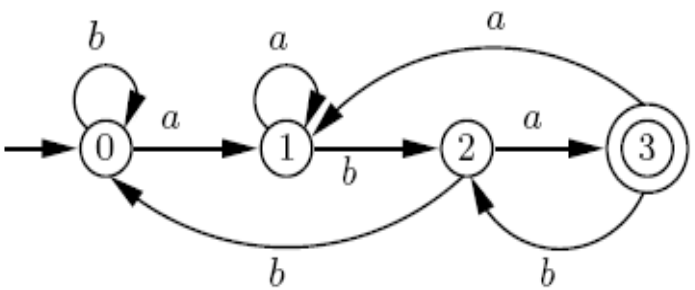
Řetězce a algoritmy na řetězcích
Základní pojmy
Základní algoritmy
Algoritmus využívající analýzu hledaného motivu
Algoritmus využívající nalýzu prohledávaného řetězce
Hledání opakování
Tandemové opakování
Palindromy
Srovnávání dvou sekvencí
Vylepšení pro maximálně k chyb
Příště
video HHMI



Automat vytvořen z motivu p postupně čte symboly z řetězce `m`. Koncový stav automatu dosáhneme po načtení celého hledaného motivu.

t=bababaa p=aba

ϵ	0
b	0
ba	1
bab	2
b**aba**	**3**
babab	2
bab**aba**	**3**
bababaa	1



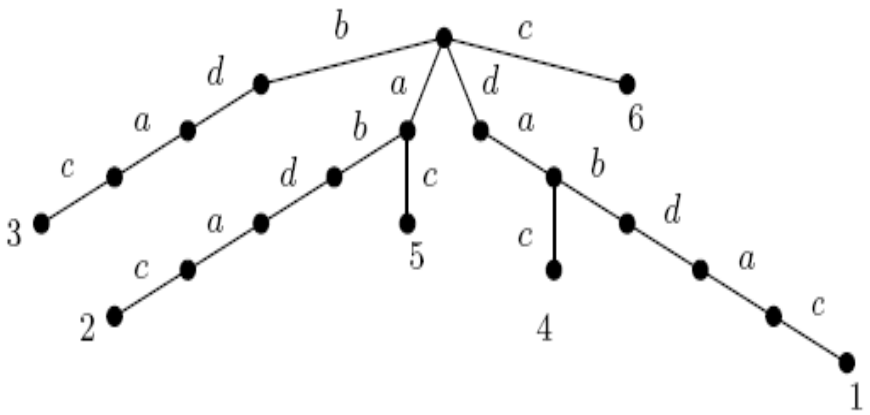
Složitost

konstrukce: naivní O(m^3); optimální O(||*abeceda*||.m)

hledání: O(n)

Řetězce a algoritmy na řetězcích
Základní pojmy
Základní algoritmy
Algoritmus využívající analýzu hledaného motivu
Algoritmus využívající nalýzu prohledávaného řetězce
Hledání opakování
Tandemové opakování
Palindromy
Srovnávání dvou sekvencí
Vylepšení pro maximálně k chyb
Příště
video HHMI

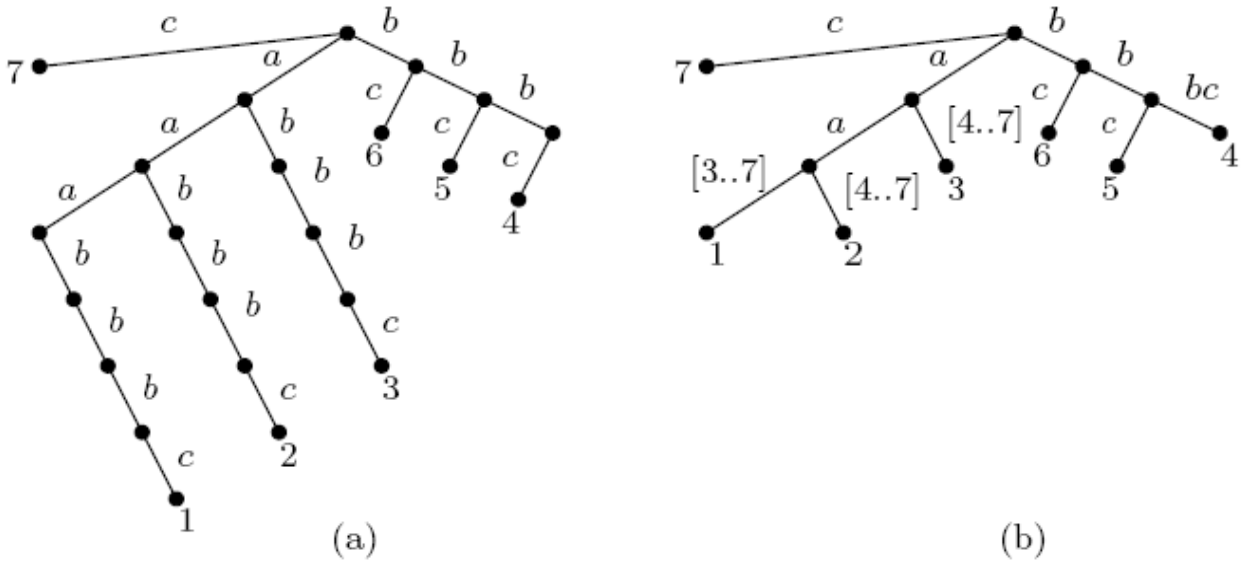
Suffixový strom pro řetězec dabdac



Řetězce a algoritmy na řetězcích
Základní pojmy
Základní algoritmy
Algoritmus využívající analýzu hledaného motivu
Algoritmus využívající nalýzu prohledávaného řetězce
Hledání opakování
Tandemové opakování
Palindromy
Srovnávání dvou sekvencí
Vylepšení pro maximálně k chyb
Příště
video HHMI

Kompaktní suffixový strom pro řetězec aaabbbc

Řetězce a algoritmy na řetězcích
Základní pojmy
Základní algoritmy
Algoritmus využívající analýzu hledaného motivu
Algoritmus využívající nalýzu prohledávaného řetězce
Hledání opakování
Tandemové opakování
Palindromy
Srovnávání dvou sekvencí
Vylepšení pro maximálně k chyb
Příště
video HHMI



Konstrukce: O(n.log*n*)
Hledání: O(m.||*abeceda*||+k)

Sufixové pole - ukazovatele na polohy suffixů seřazené lexikograficky

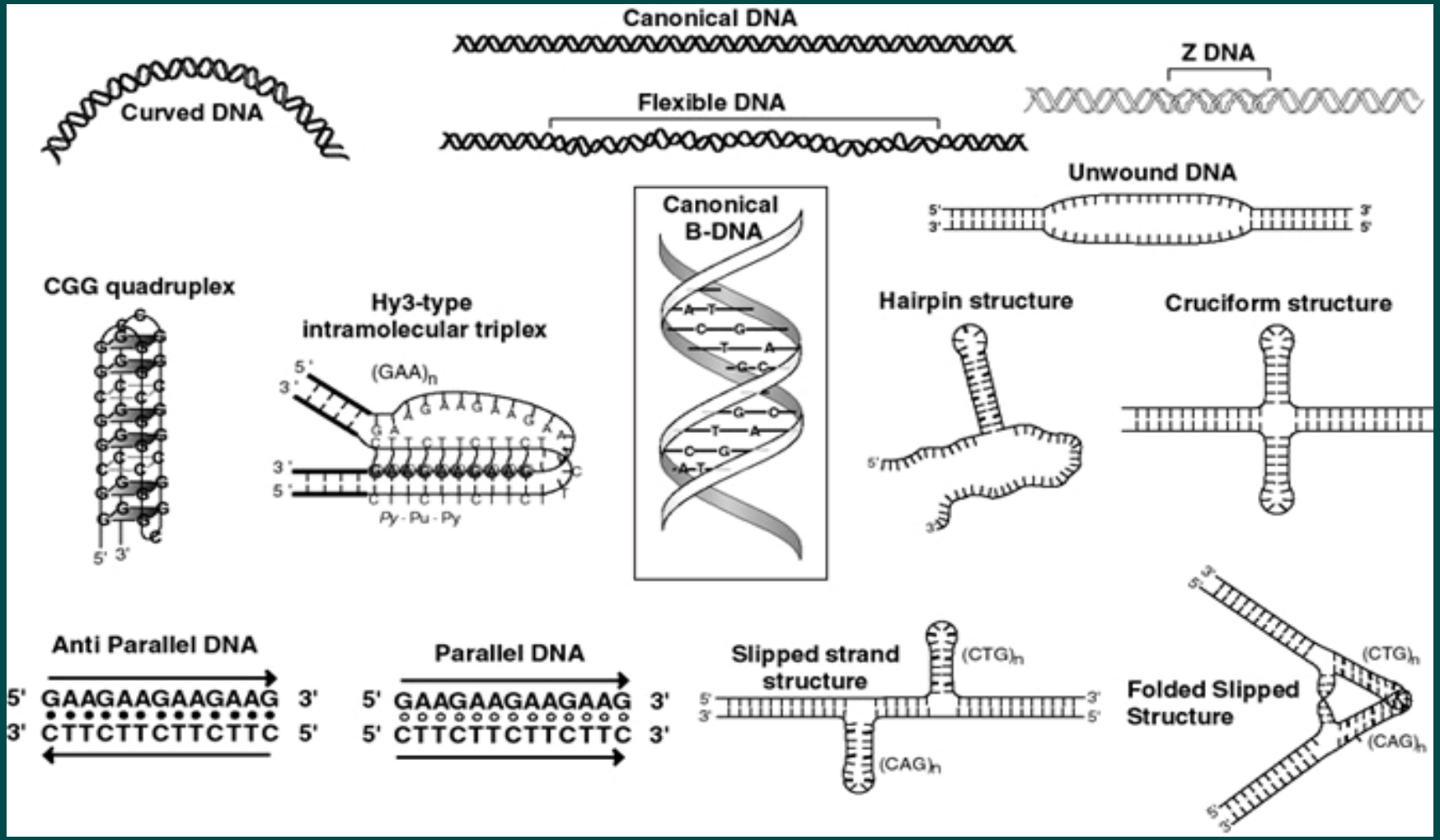
Řetězce a algoritmy na řetězcích
Základní pojmy
Základní algoritmy
Algoritmus využívající analýzu hledaného motivu
Algoritmus využívající nalýzu prohledávaného řetězce
Hledání opakování
Tandemové opakování
Palindromy
Srovnávání dvou sekvencí
Vylepšení pro maximálně k chyb
Příště
video HHMI

Dlouho bylo považováno za méně kvalitní datovou strukturu, protože neobsahuje přímo informace o společných prefixech. Ty lze vsak spočítat do lcp pole (least common prefix) tak, že konstrukce pole i stromu má stejnou složitost.

t = dabdac
sa(t) = 6,1,4,2,5,0,3
rank(t) = 5,1,3,6,2,4,0
lcp(t) = 0,0,1,0,0,0,2

```
6 0
1 0 abdac
4 1 ac
2 0 bdac
5 0 c
0 0 dabdac
3 2 dac
```

Sufixové pole



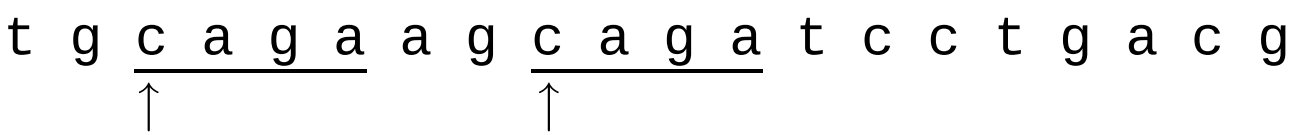
Tandemová a palindromická opakování nesou biologický i praktický význam

Řetězce a algoritmy na řetězcích
Základní pojmy
Základní algoritmy
Algoritmus využívající analýzu hledaného motivu
Algoritmus využívající nalýzu prohledávaného řetězce
Hledání opakování
Tandemové opakování
Palindromy
Srovnávání dvou sekvencí
Vylepšení pro maximálně k chyb
Příště
video HHMI

palindrom	možná sekundární struktura DNA nebo RNA
tandem	regulace genů, telomery, identifikace jedinců z DNA

Nejdelší společný prefix dvou pozic

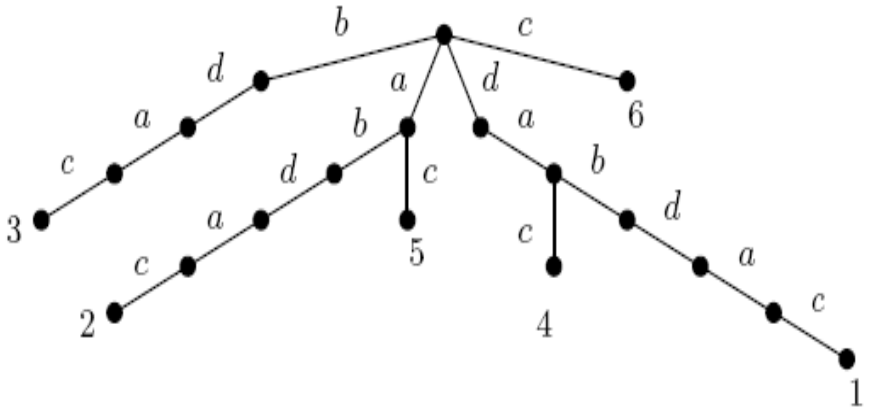
Řetězce a algoritmy na řetězcích
Základní pojmy
Základní algoritmy
Algoritmus využívající analýzu hledaného motivu
Algoritmus využívající nalýzu prohledávaného řetězce
Hledání opakování
Tandemové opakování
Palindromy
Srovnávání dvou sekvencí
Vylepšení pro maximálně k chyb
Příště
video HHMI



Složitost naivního algoritmu O(n^3)

Posuzování tandemových opakování pomocí suffixových stromů, příp. polí (t=dabdac)

Řetězce a algoritmy na řetězcích
Základní pojmy
Základní algoritmy
Algoritmus využívající analýzu hledaného motivu
Algoritmus využívající nalýzu prohledávaného řetězce
Hledání opakování
Tandemové opakování
Palindromy
Srovnávání dvou sekvencí
Vylepšení pro maximálně k chyb
Příště
video HHMI



lcp(1,4)=?

Nalezneme větve označené 1 a 4

lcp(1,4)=da

Hledání tandemových opakování

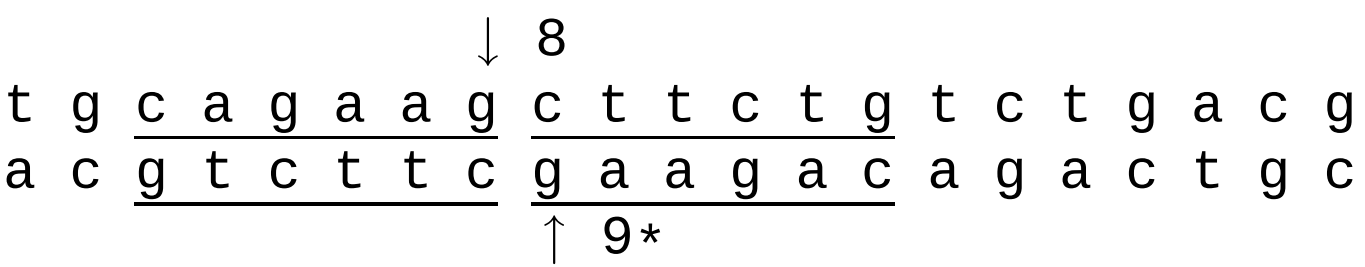
Řetězce a algoritmy na řetězcích
Základní pojmy
Základní algoritmy
Algoritmus využívající analýzu hledaného motivu
Algoritmus využívající nalýzu prohledávaného řetězce
Hledání opakování
Tandemové opakování
Palindromy
Srovnávání dvou sekvencí
Vylepšení pro maximálně k chyb
Příště
video HHMI

- konstrukce stromu: O(n.*log*n)
- hledání lcp pro dvě konkrétní pozice O(n.*log*n)
- Prohledávání sekvence

Složitost: $O(n.(logn)^2+p)$

Palindromy - nejdelší společný prefix mezi originální a komplementární sekvencí umožňuje urychlení hledání podobně jako pro tandemové opakování

Řetězce a algoritmy na řetězcích
Základní pojmy
Základní algoritmy
Algoritmus využívající analýzu hledaného motivu
Algoritmus využívající nalýzu prohledávaného řetězce
Hledání opakování
Tandemové opakování
Palindromy
Srovnávání dvou sekvencí
Vylepšení pro maximálně k chyb
Příště
video HHMI



Složitost naivního algoritmu O(n^3)

Složitost naivního algoritmu O(nlp) (pro omezenou vzdálenost a délku)
Složitost s použitím suffixových struktur O(n)

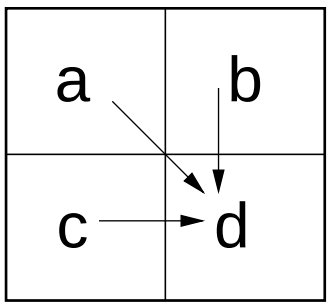
Využití DP pro identifikaci palindromů

Řetězce a algoritmy na řetězcích
Základní pojmy
Základní algoritmy
Algoritmus využívající analýzu hledaného motivu
Algoritmus využívající nalýzu prohledávaného řetězce
Hledání opakování
Tandemové opakování
Palindromy
Srovnávání dvou sekvencí
Vylepšení pro maximálně k chyb
Příště
video HHMI

	m	i	s	s	i	s	s	i	p	p	i
i										0	0
p									0	0	1
p								0	0	0	1
i							0	0	1	1	0
s						0	0	1	1	2	1
s					0	0	0	1	2	3	2
i				0	0	1	1	0	1	2	3
s			0	0	1	0	1	1	1	2	3
s		0	0	0	1	1	0	1	2	2	3
i	0	0	1	1	0	1	1	0	1	2	2
m	0	1	1	2	1	1	2	1	1	2	3

match

a)

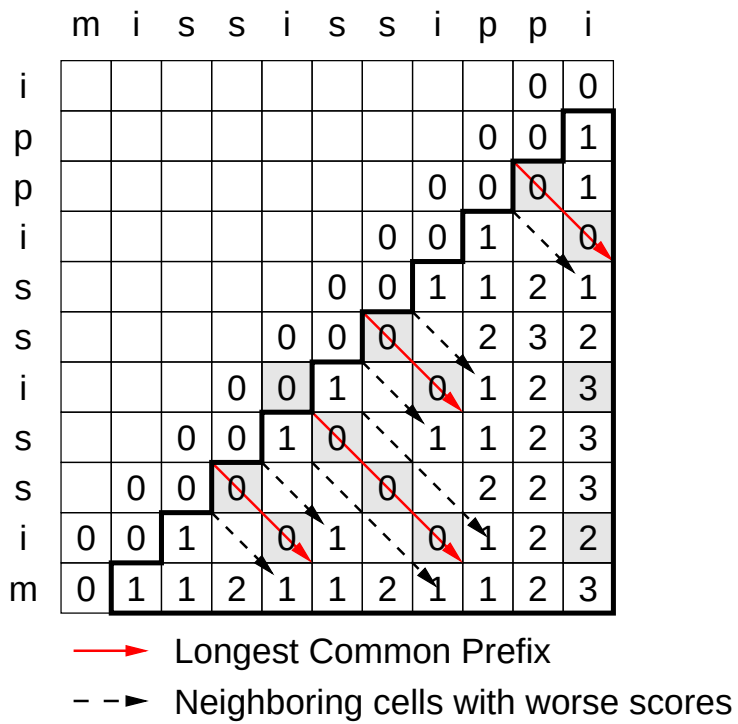


$$d = \min \begin{cases} a & \text{match} \\ a+1 & \text{mismatch} \\ b+1 & \text{insertion} \\ c+1 & \text{deletion} \end{cases}$$

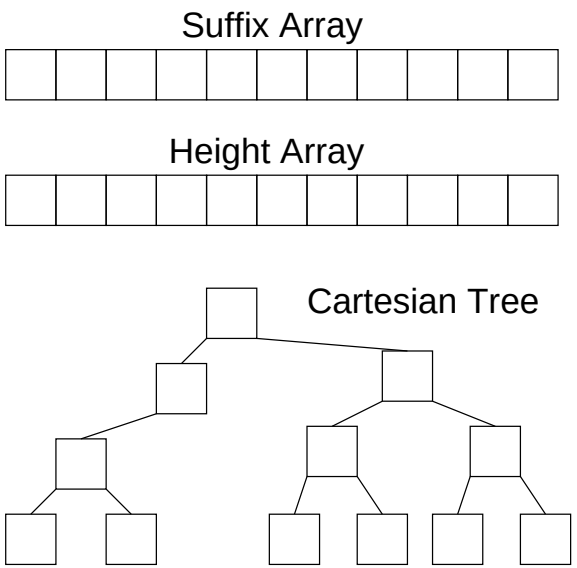
b)

Využití SA a LCP k rychlému postupu po diagonále

Řetězce a algoritmy na řetězcích
Základní pojmy
Základní algoritmy
Algoritmus využívající analýzu hledaného motivu
Algoritmus využívající nalýzu prohledávaného řetězce
Hledání opakování
Tandemové opakování
Palindromy
Srovnávání dvou sekvencí
Vylepšení pro maximálně k chyb
Příště
video HHMI



a)



b)

Stupeň podobnosti dvou sekvencí

IDENTITA

```
MASAQSFYLL
||||||||||
MASAQSFYLL
```

SUBSTITUCE

```
MASAQSFYLL        MASAQSFYLL
||||||:|||        |||||| |||
MASAQSWYLL        MASAQSTYLL
```

INZERCE/DELECE

```
MASAQSFYLL
|||||| |||
MASAQS-YLL
```

TRANSPOZICE

```
MASAQSFYLL
|||||  |||
MASAQFSYLL
```

Stupeň podobnosti dvou sekvencí

Netriviální hodnocení substitucí u proteinů (matice PAM250)

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	2																			
R	-2	6																		
N	0	0	2																	
D	0	-1	2	4																
C	-2	-4	-4	-5	12															
Q	0	1	1	2	-5	4														
E	0	-1	1	3	-5	2	4													
G	1	-3	0	1	-3	-1	0	5												
H	-1	2	2	1	-3	3	1	-2	6											
I	-1	-2	-2	-2	-2	-2	-2	-3	-2	5										
L	-2	-3	-3	-4	-6	-2	-3	-4	-2	-2	6									
K	-1	3	1	0	-5	1	0	-2	0	-2	-3	5								
M	-1	0	-2	-3	-5	-1	-2	-3	-2	2	4	0	6							
F	-3	-4	-3	-6	-4	-5	-5	-5	-2	1	2	-5	0	9						
P	1	0	0	-1	-3	0	-1	0	0	-2	-3	-1	-2	-5	6					
S	1	0	1	0	0	-1	0	1	-1	-1	-3	0	-2	-3	1	2				
T	1	-1	0	0	-2	-1	0	0	-1	0	-2	0	-1	-3	0	1	3			
W	-6	2	-4	-7	-8	-5	-7	-7	-3	-5	-2	-3	-4	0	-6	-2	-5	17		
Y	-3	-4	-2	-4	0	-4	-4	-5	0	-1	-1	-4	-2	7	-5	-3	-3	0	10	
V	0	-2	-2	-2	-2	-2	-2	-1	-2	4	2	-2	2	-1	-1	-1	0	-6	-2	4

Stupeň podobnosti dvou sekvencí

```
        A  G  A  T  A
    0 -2 -4 -6 -8 -10
A -2  2  0 -2 -3 -5
G -4 -1  ?
T -6
C -8
A -10

INDEL=-2 IDENT=2 SUBST=-1

AGATA
||  |    S=4
AGTCA
```

BLAST (basic local alignment search tool)

Co když jsou sekvence dlouhé a máme jich několik milionů ?

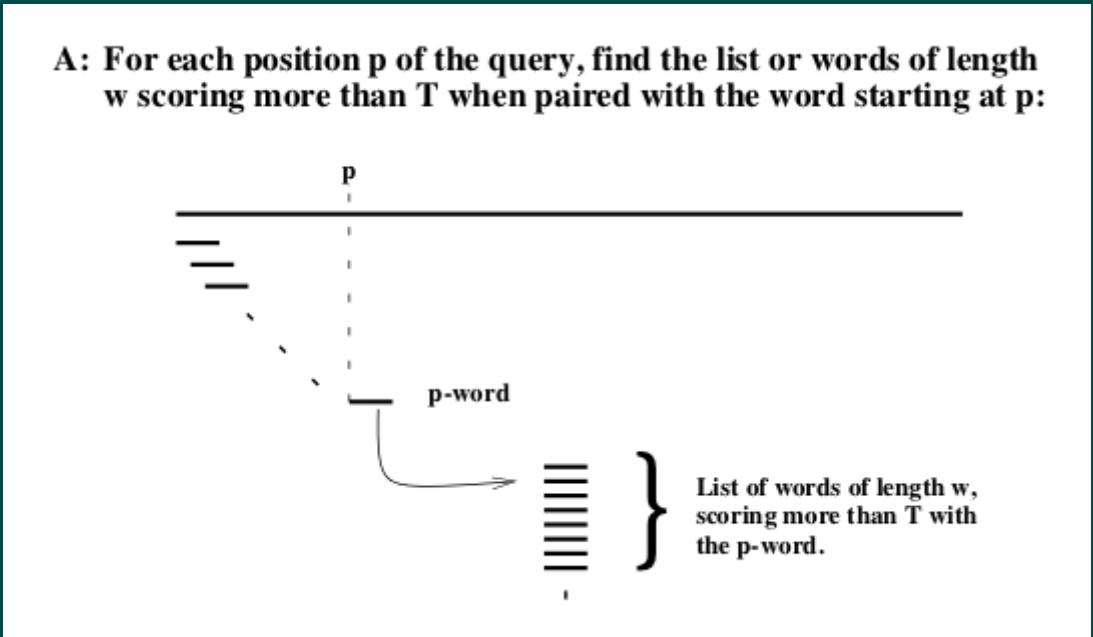
DP nestačí, výpočty trvají příliš dlouho. Alternativou výpočtu by byl předpočítaný soubor podobnosti různých slov v databázi. Problémem indexu je, že je pro dlouhá slova nezvladatelný objemově. Existuje např.

$20^8 = 25\ 600\ 000\ 000$

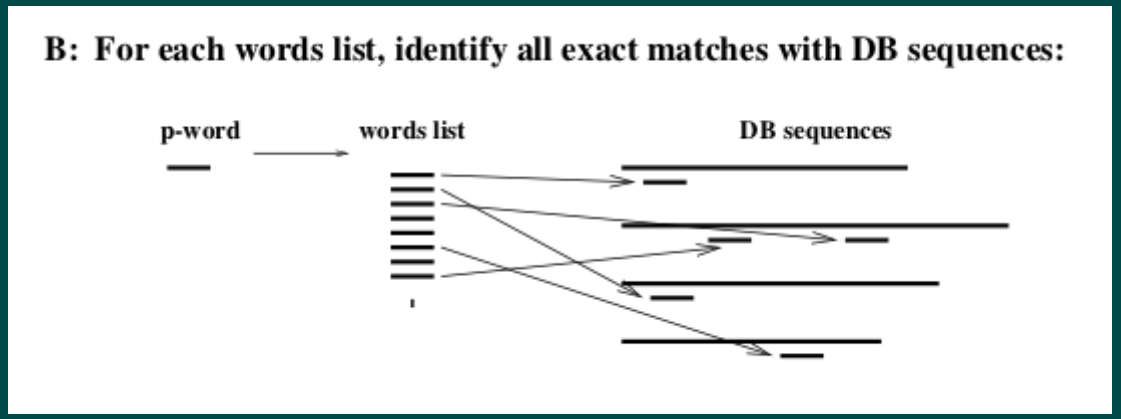
různých uspořádání osmi aminokyselin v řetězci, několik způsobů hodnocení podobnosti atd.

Kompromisem je heuristické řešení. Nalezení tzv. “seeds”, výskytu krátkých řetězců (2-4 aminokyseliny, 7-11 nukleotidů) a hledání podobnosti algoritmem DP jenom v jejich blízkosti.

BLAST (basic local alignment search tool)



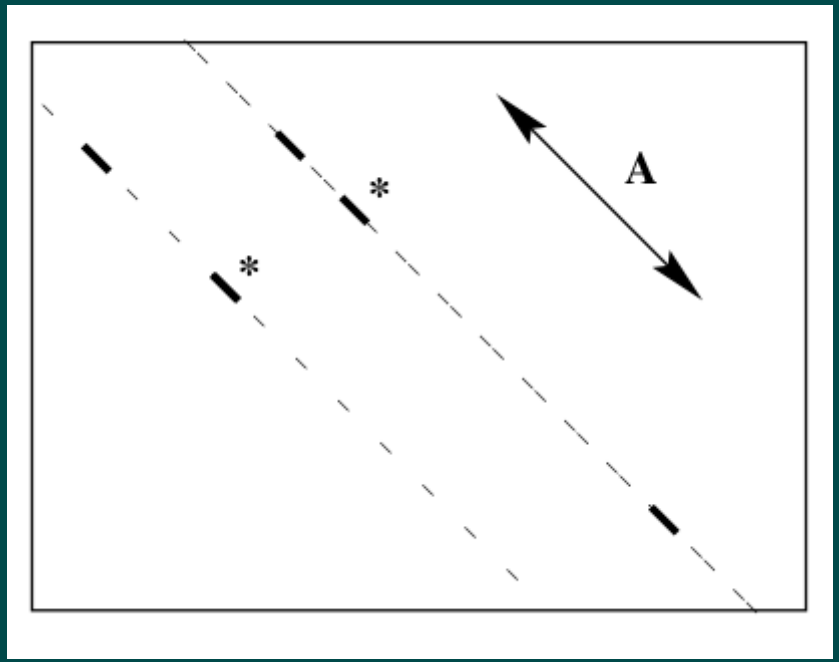
BLAST (basic local alignment search tool)



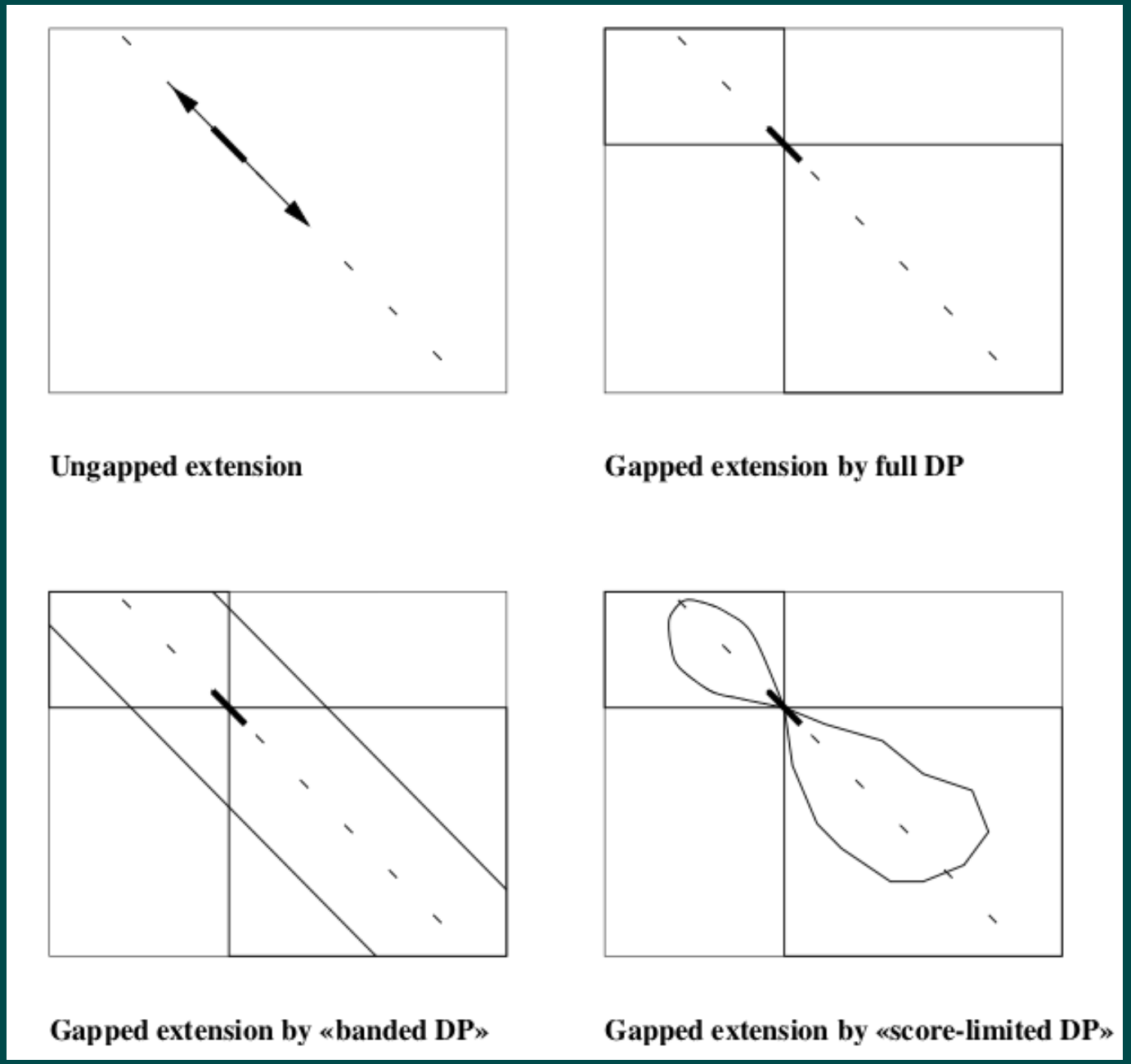
BLAST (basic local alignment search tool)

C: For each word match («hit»), extend ungapped alignment in both directions. Stop when S decreases by more than X from the highest value reached by S.

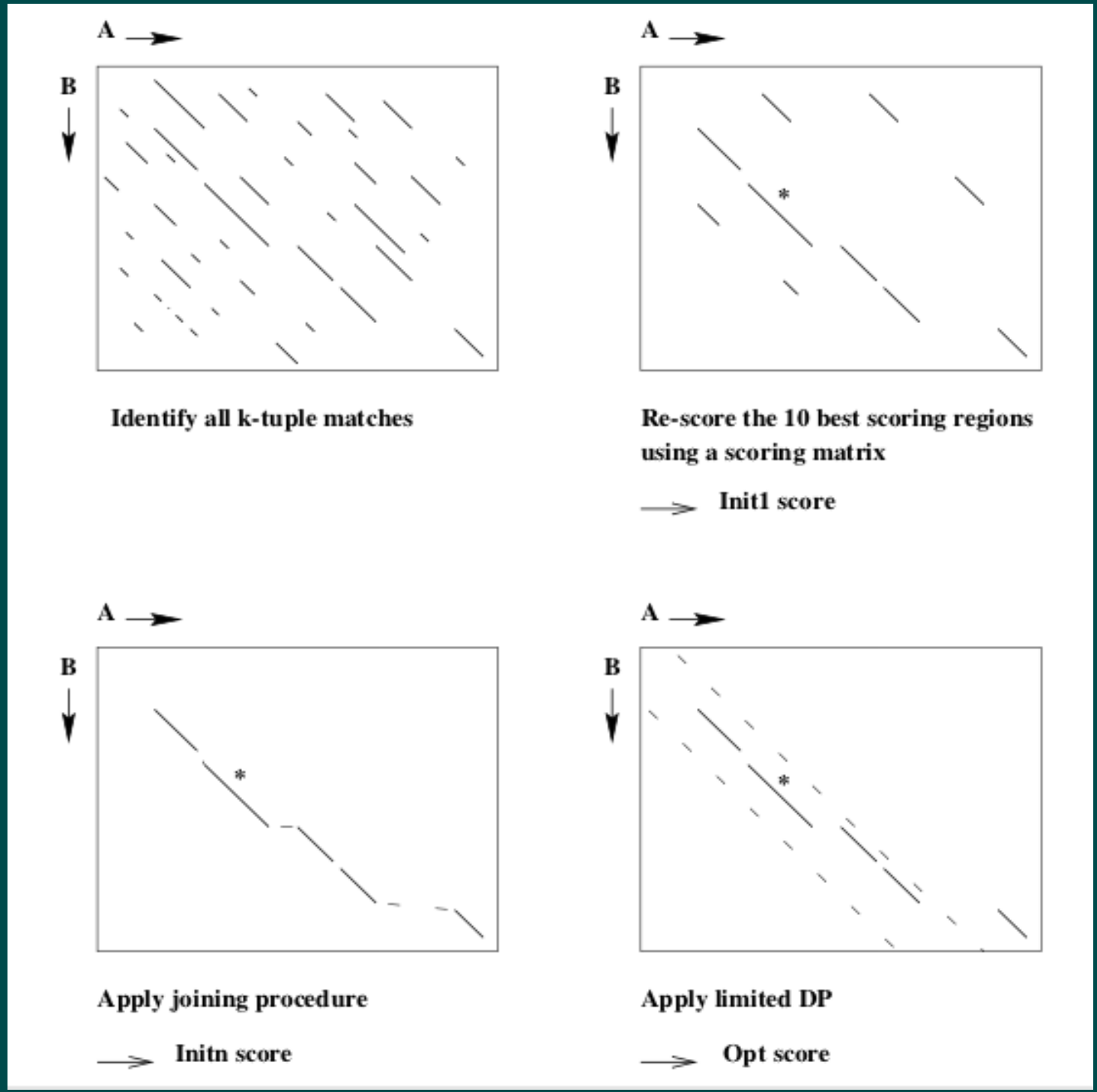
BLAST (basic local alignment search tool)



BLAST (basic local alignment search tool)



FASTA


A
B
Identify all k-tuple matches
A
B
*
Re-score the 10 best scoring regions using a scoring matrix
Init1 score
A
B
*
Apply joining procedure
Initn score
A
B
*
Apply limited DP
Opt score

BLAST (basic local alignment search tool)

P-VALUE . . . P(score>S)

PRAVDĚPODOBNOST VÝSKYTU PODOBNOSTI VĚTŠÍ
NEŽ S V NÁHODNÝCH SEKVENCÍCH URČITÉ DÉLKY

P(MSP(M,N)>S) = 1 - exp(-Kmn.exp(-lambda.S))

E-VALUE

OČEKÁVANÝ POČET PODOBNOSTÍ KDE score>S

Kmn.exp(-lambda.S)

BLAST (basic local alignment search tool)

PAM150
Percent Accepted Mutations

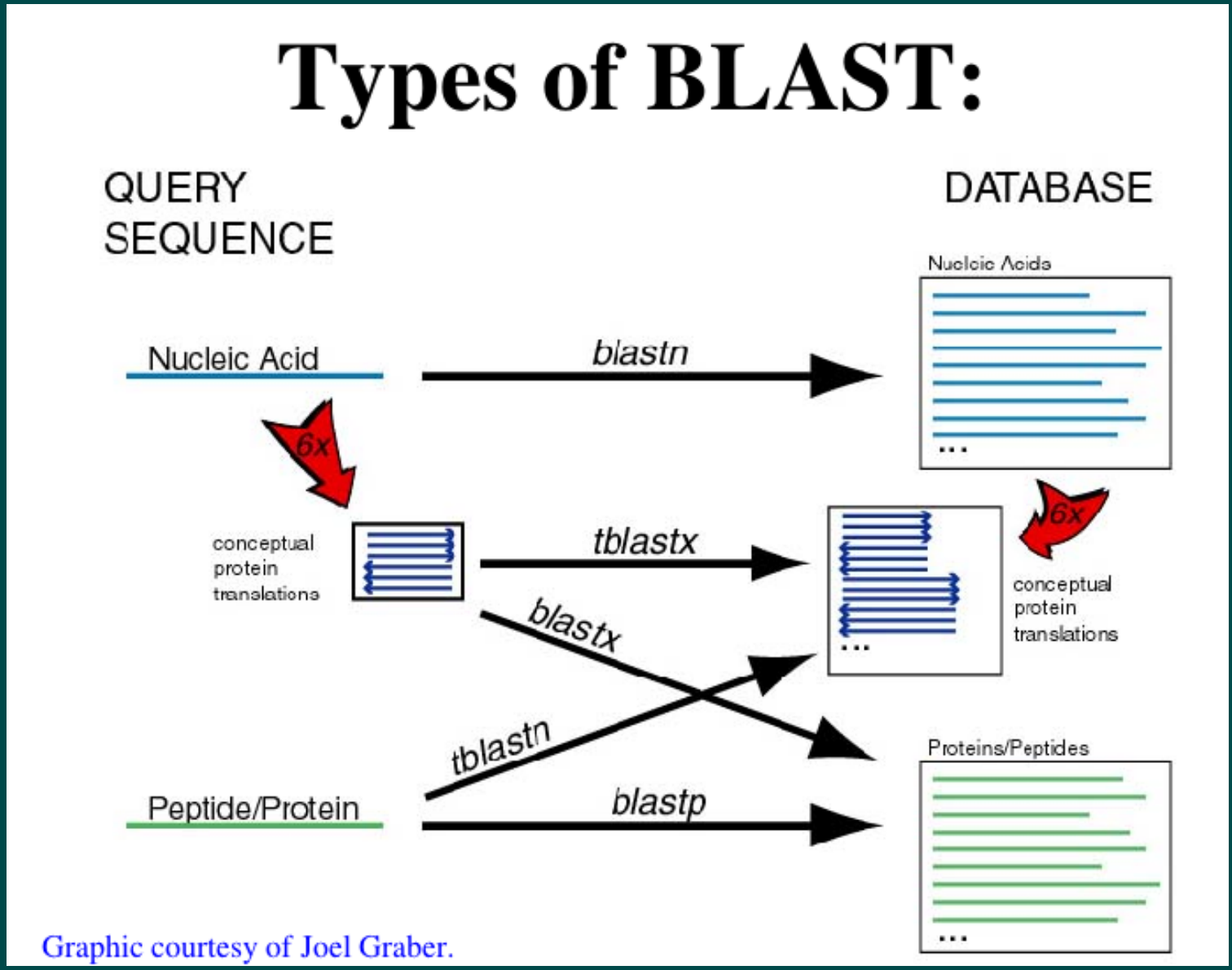
Substituční matice odvozena z předpokladu 150 mutací na 100 pozic sekvence

BLOSUM65
BLOck SUbstitution Matrix

Substituční matice odvozena ze sekvencí se 65% identitou

PAM120 <=> BLOSUM80
PAM250 <=> BLOSUM45

BLAST (basic local alignment search tool)



BLAST (basic local alignment search tool)

```
>gi|50757596|ref|XP_425354.1|similar to protein kinase
 Length = 613

 Score = 50.4 bits (119), Expect(2) = 2e-17
 Identities = 26/54 (48%), Positives = 36/54 (66%), Gaps = 1/54 (1%)

Query: 740 YVMVLEYANEGNLREYLEK-KFDTLQWENKIQMALDITRGLLCLHSRNIIHRDL 582
           Y +V EY +EG+LR YL K +  +L  +  I  ALDI RG+  +HS+ +IHRDL
Sbjct: 250 YCVVTEYLSEGSLRAYLHKLERKSLPLQKLIAFALDIARGMEYIHSQGVIHRDL 303
```

BLAST (basic local alignment search tool)

BLAST (NCBI-BLAST WU-BLAST)
   BLASTN BLASTP BLASTX TBLASTN TBLASTX
MEGABLAST
PSI-BLAST
PHI-BLAST
SNPBLAST
BLASTZ

Pattern Hunter

PATTERN HUNTER **BLAST**

MODEL SLOVA

```
110100110010101111     11111111111
```

ZÁVISLOST POZIC PŘEKRYV

```
110100110010101111     11111111111
 110100110010101111     11111111111     5   10
  110100110010101111     11111111111    5    9
   110100110010101111     11111111111   5    8
    110100110010101111     11111111111  4    7
```

Pattern Hunter

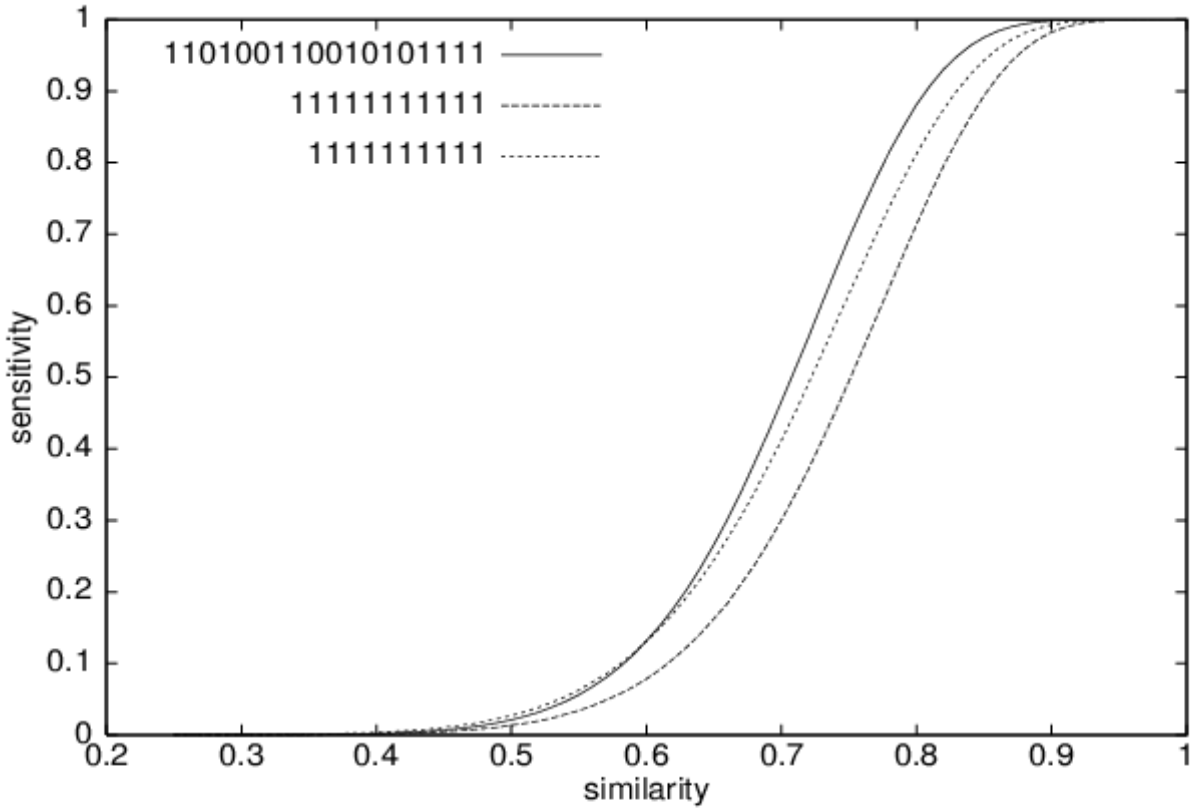


Figure 1: 1-hit performance of weight 11 spaced model versus weight 11 and 10 consecutive models, coordinates in logarithmic scale.

Pattern Hunter

Seq1	Size	Seq2	Size	PH	PH2	MB28	Blastn
M. pneumoniae	828K	*M. genitalium*	589K	10s/65M	4s/48M	1s/88M	47s/45M
E. coli	4.7M	*H. influenza*	1.8M	34s/78M	14s/68M	5s/561M	716s/158M
A.thaliana chr 2	19.6M	*A.thaliana* chr 4	17.5M	5020s/279M	498s/231M	21720s/1087M	∞
H. sapiens chr 22	35M	*H. sapiens* chr 21	26.2M	14512s/419M	5250s/417M	∞	∞

Pattern Hunter

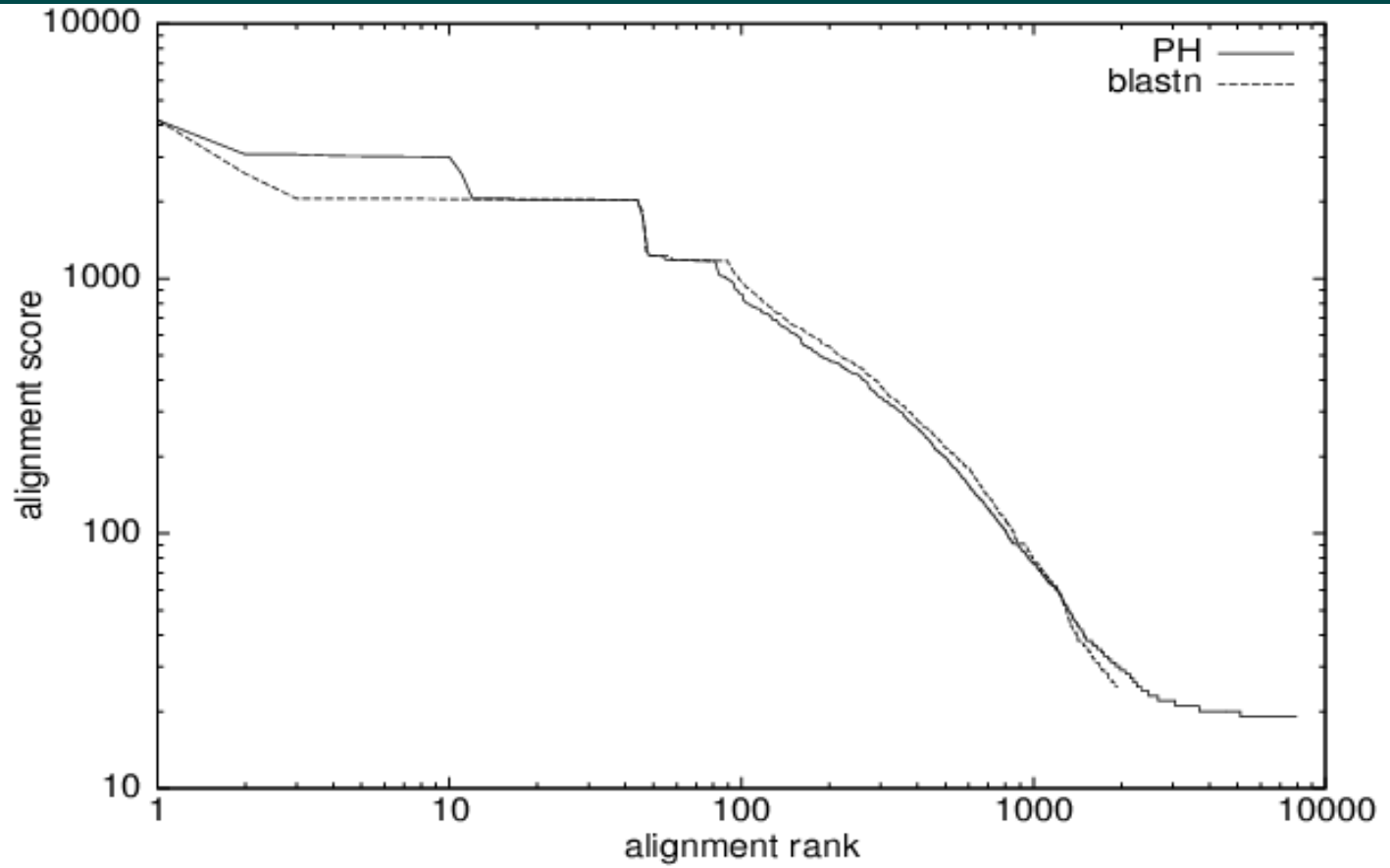


Figure 5: Input: *H. influenza* and *E. coli*. Run times are shown in Figure 7. PatternHunter produces better quality output than Blastn while running 20 times faster.

Pattern Hunter

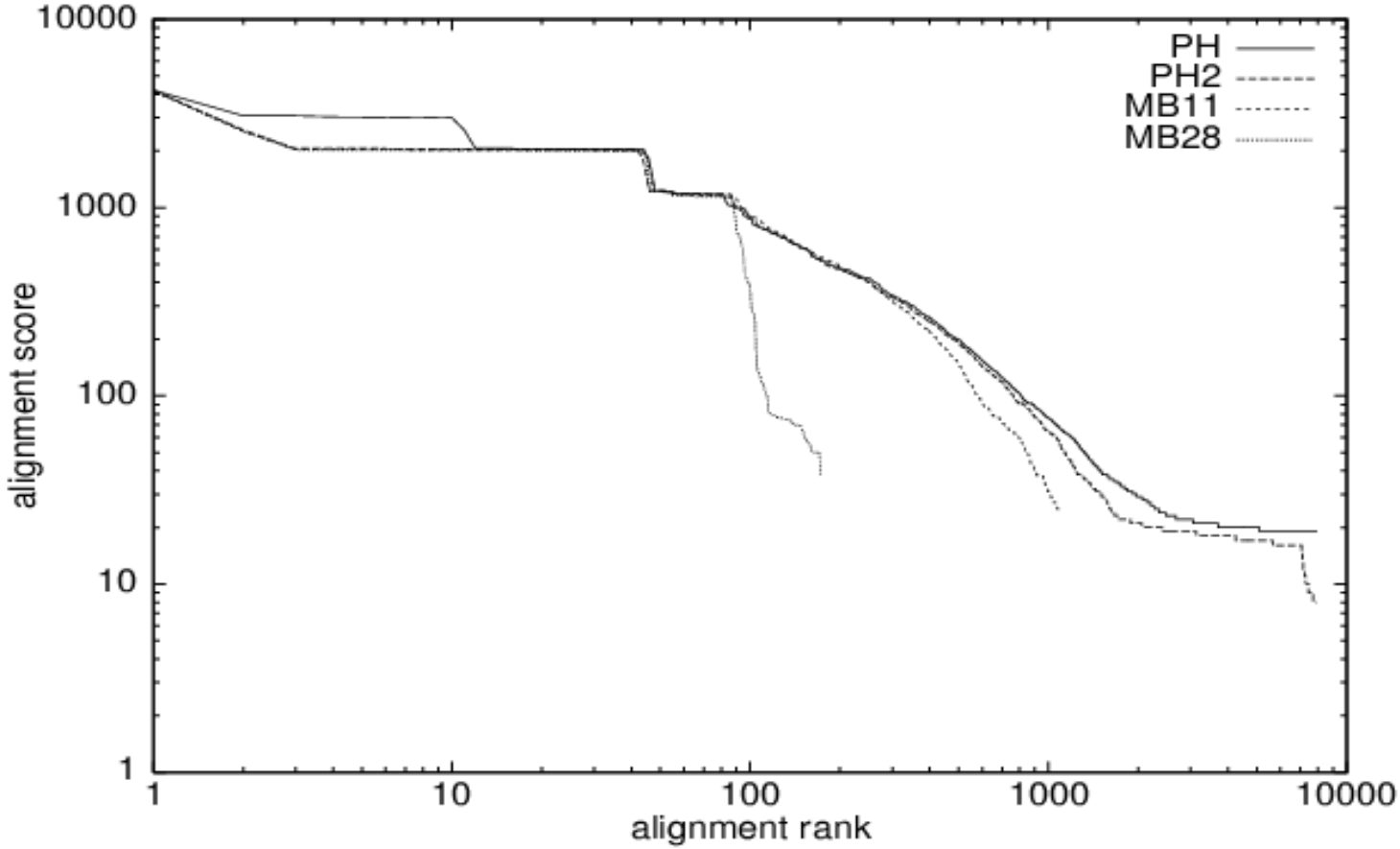


Figure 4: Input: *H. influenza* and *E. coli*. Run times are shown in Figure 7. Score is plotted as a function of the rank of the alignment, with both axes logarithmic. MegaBlast (MB28) misses over 700 alignments

Primex

INDEX — 4^w+ n -> 150Mbp (w=12) -> 167M (~650MB)

FILTRACE — **ACGAGATGACGA***TGACGATGCGAT*

DP (N-W) — Probíhá na omezeném vzorku sekvencí

Primex

ATAGTA*GGTCCG***TCGATA**

18 bp -> 3 slova po 6bp

nastavením *m1*=1 chyby na slovo nalezneme spolehlivě v nejhorším

GT**A**T**T**A*GG*T*A*CG*T**T**G**A**C**A**

i řetězec s 5 chybami

Sekvence se 6 chybami už nemusí být nalezena:

GT**A**T**T**A*G*A*T*A*CG*T**T**G**A**C**A** **nenalezena**

GT**A**T**T**A*GGT*A*CG*T**T**G**A**C**G** **nalezena**

Rozdělení vyhledávacího řetězce na menší segmenty urychlí výpočty tím, že u kratších lze efektivně využít rychlého indexu celé genomové databázy.

Primex

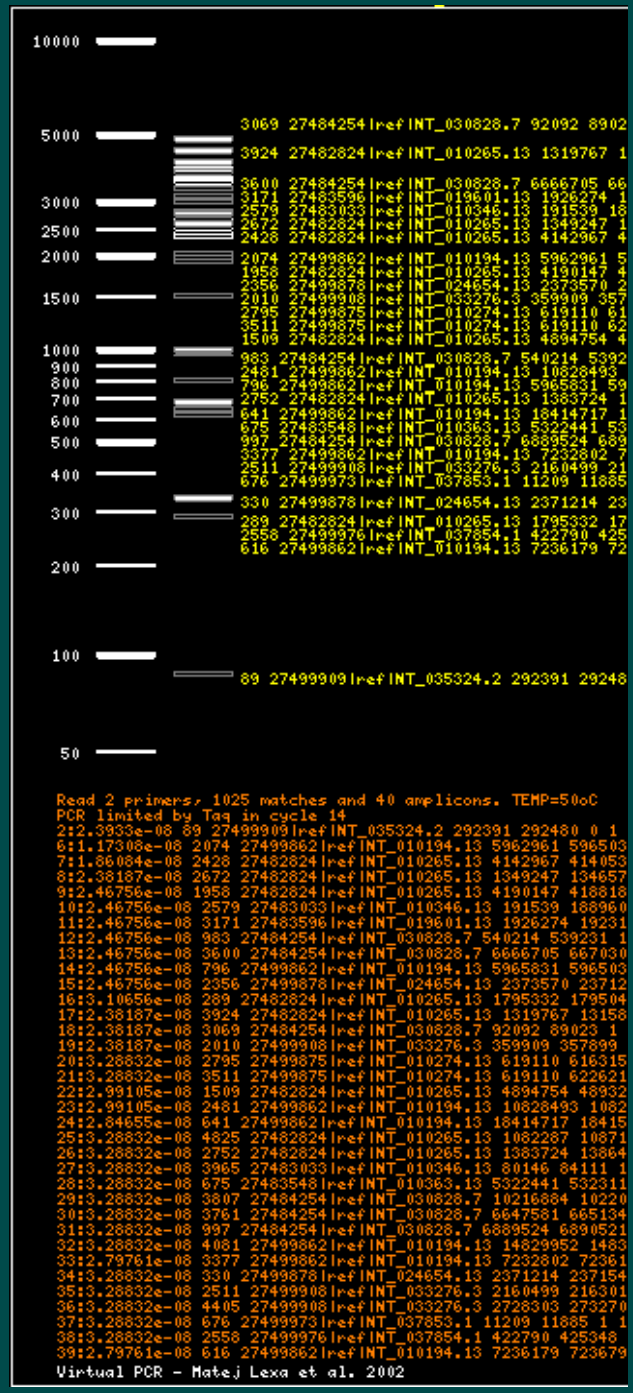
		MISMATCHES						
PROGRAM	TOTAL	0	1	2	3	4	5	6	SEARCH TIME
BLAST	162	1	0						12 s
BLAST-O	2	1							5 s
FASTA	72	1	0	4	13	23	17	12	53 s
FASTA-O	1	1							19 s
BLAT	0	0							80 s
SSAHA	0	0							71 s
SSAHA-O	3	1							10 s
CGC FP-O	1	1							10 s
EMBOSS	983	1	0	5	104	86	8		18 s
EMBOSS-O	1	1							14 s
TACG	1	1							49 s
AGREP	1204	1	0	5	100	779	3632		34 s
AGREP	1	1							2 s
PRIMEX	214	1	0	14	199				56 s
PRIMEX-S (2,5)	12140	1	0	14	199	1686	10240		19 s
PRIMEX-S (2,4)	1900	1	0	14	199	1686			4 s
PRIMEX-S (2,3)	214	1	0	14	199				1 s
PRIMEX-S (2,2)	15	1	0	14					< 1 s
PRIMEX-S (2,1)	1	1	0						<< 1 s
PRIMEX-SO (0,0)	1	1							<< 1 s

Výkony vybraných vyhledávacích programů při hledání výskytu sekvence oligonukleotidu AAAAAATGATCAATTTACAT v genomu Arabidopsis thaliana (cca 100Mbp). Příponou -O jsou označeny programy, které byly nastaveny s nejmenší citlivostí. Přípona -S označuje programy, které v momentě dotazu běžely jako server.

Primex + Virtual PCR

Vyhledané oligonukleotidy lze použít pro simulaci PCR reakce

VPCR Virtual PCR



BLAT

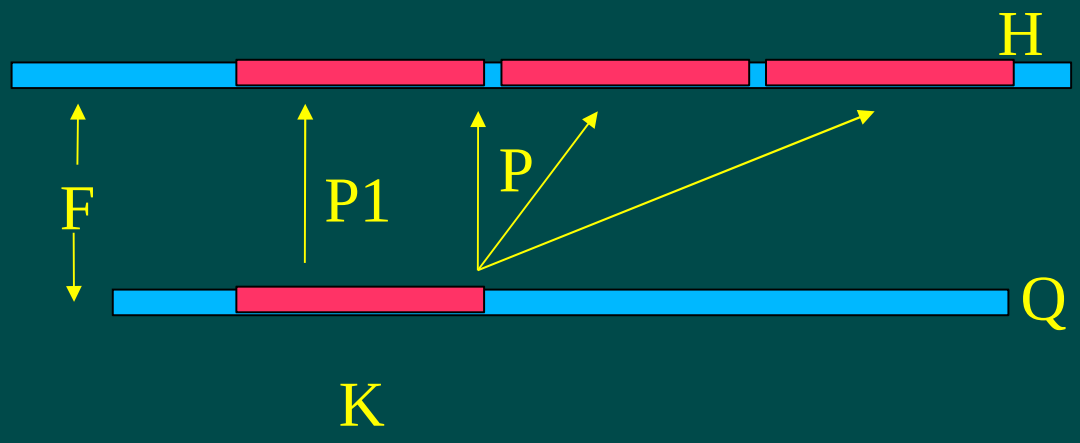
$P1 = M^K$

$T = int(H/K)$

$P = 1 - (1-P1)^T$

$P = 1 - (1-M^K)^T$

$F = (Q-K+1)*(G/K)*(1/A)^K$



P1 – pravdepodonost shody s k-merem
P – pravdepodobnost existence aspon jedne takove shody
F – pravdepodobnost nahodneho vyskytu shody mezi H a Q
K – délka k-meru
H – délka úseku podobnosti (několik 100 bp)
M – zhoda mezi sekvencemi (%identity/100)
G – velikost databázy
Q – velikost vyhledávací sekvence
A – velikost abecedy

BLAT

Table 3. Sensitivity and Specificity of Single Perfect Nucleotide K-mer Matches as a Search Criterion

	7	8	9	10	11	12	13	14
A. 81%	0.974	0.915	0.833	0.726	0.607	0.486	0.373	0.314
83%	0.988	0.953	0.897	0.815	0.711	0.595	0.478	0.415
85%	0.996	0.978	0.945	0.888	0.808	0.707	0.594	0.532
87%	0.999	0.992	0.975	0.942	0.888	0.811	0.714	0.659
89%	1.000	0.998	0.991	0.976	0.946	0.897	0.824	0.782
91%	1.000	1.000	0.998	0.993	0.981	0.956	0.912	0.886
93%	1.000	1.000	1.000	0.999	0.995	0.987	0.968	0.957
95%	1.000	1.000	1.000	1.000	0.999	0.998	0.994	0.991
97%	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.999

B. K	7	8	9	10	11	12	13	14
F	1.3e+07	2.9e+06	635783	143051	32512	7451	1719	399

(*A*) Columns are for K sizes of 7–14. Rows represent various percentage identities between the homologous sequences. The table entries show the fraction of homologies detected as calculated from equation 3 assuming a homologous region of 100 bases. The larger the value of K, the fewer homologies are detected.

(*B*) K represents the size of the perfect match. F shows how many perfect matches of this size expected to occur by chance according to equation 4 in a genome of 3 billion bases using a query of 500 bases.

BLAT

Table 5. Sensitivity and Specificity of Single Near-Perfect (One Mismatch Allowed) Nucleotide K-mer Matches as a Search Criterion

	12	13	14	15	16	17	18	19	20	21	22
A. 81%	0.945	0.880	0.831	0.721	0.657	0.526	0.465	0.408	0.356	0.255	0.218
83%	0.975	0.936	0.904	0.820	0.770	0.649	0.591	0.535	0.480	0.361	0.318
85%	0.991	0.971	0.954	0.900	0.865	0.767	0.719	0.669	0.619	0.490	0.445
87%	0.997	0.990	0.983	0.954	0.935	0.867	0.833	0.796	0.757	0.634	0.591
89%	1.000	0.997	0.995	0.984	0.976	0.939	0.920	0.897	0.872	0.775	0.741
91%	1.000	1.000	0.999	0.996	0.994	0.979	0.971	0.962	0.950	0.890	0.869
93%	1.000	1.000	1.000	0.999	0.999	0.996	0.994	0.991	0.988	0.963	0.954
95%	1.000	1.000	1.000	1.000	1.000	1.000	0.999	0.999	0.999	0.994	0.992
97%	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000

B. K	12	13	14	15	16	17	18	19	20	21	22
F	275671	68775	17163	4284	1070	267	67	17	4.2	1.0	0.3

(*A*) Columns are for K sizes of 12–22. Rows represent various percentage identities between the homologous sequences. The table entries show the fraction of homologies detected as calculated by equation 6 assuming a homologous region of 100 bases. (*B*) K represents the size of the near-perfect match. F shows how many perfect matches of this size expected to occur by chance according to equation 7 in a genome of 3 billion bases using a query of 500 bases.

PSST

Struktura sekvence znázorněna vektorem přítomnosti jednotlivých “slov” v=(1,0,0,1,….,1), kde každá pozice odpovídá určitému “slovu”. pi je normalizovaná frekvence výskytu daného slova a podobnost

$$S(q,t) = \text{suma}\ (q_i * t_i * 1/p_i)$$

Vektory proteinů bez jakékoliv společné podsekvence na dané úrovni jsou ortogonální, S(q,t)=0.

Pro identické proteiny S(q,q)= suma(1/pi). Pokud by všechny slova měla stejnou pravděpodobnost výskytu, pak by to bylo

N*(1/N) = 1

kde N je počet sledovaných nebo existujících slov.