IV107 Bioinformatika I

Přednáška 10

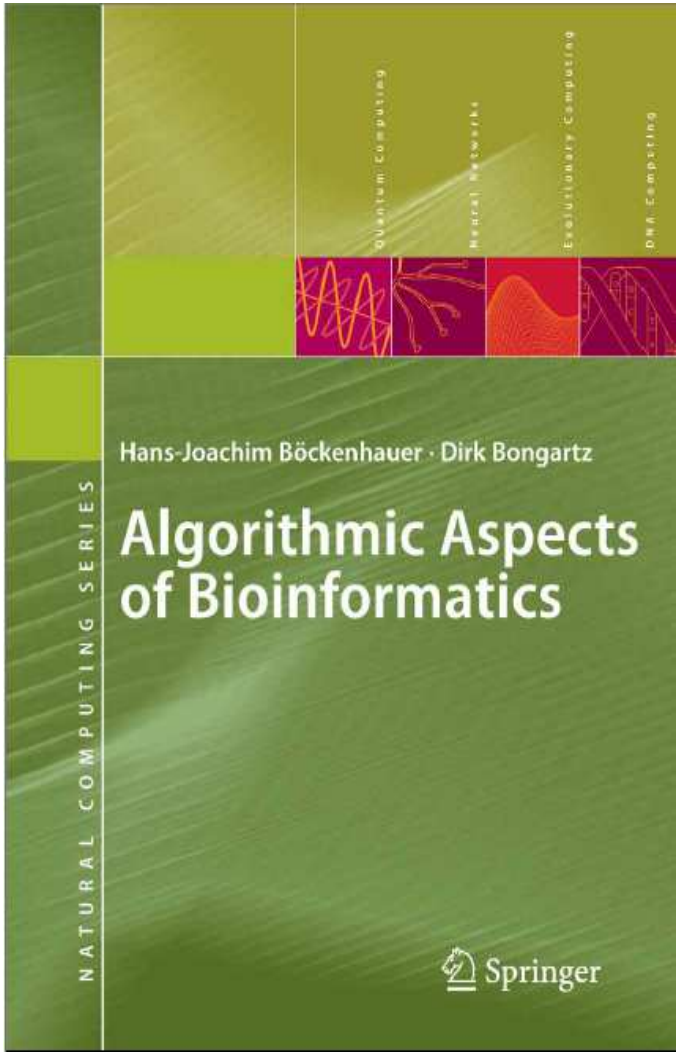
Katedra informačních technologií
Masarykova Univerzita Brno

Jaro 2010

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
Vylepšení pro maximálně k chyb
Příště
video HHMI

Podrobné informace



Základní pojmy

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

Základní operace

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

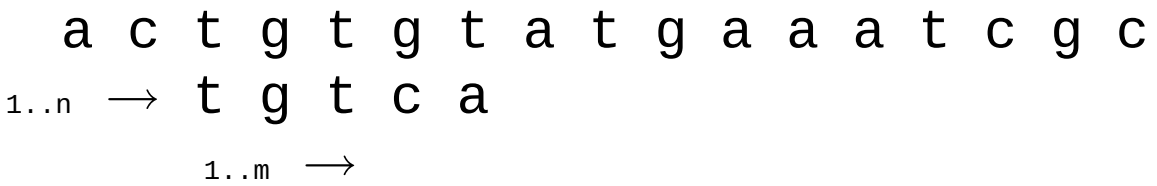
Cílem je zjistit všechny pozice delšího řetězce, na kterých se vyskytuje kratší řetězec

- přesný výskyt
- přibližný výskyt

řetězec t dlouhý (n), např genomová sekvence

motiv p krátký (m), např `cgcggctggtggctcg`

Naivní algoritmus



Složitost: O(mn)

Boyer-Moore


```
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
```

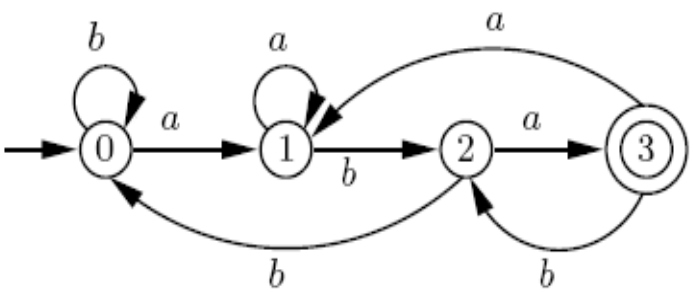
Realizujeme krok, který je větší

Složitost

konstrukce: O(||*abeceda*||.m)

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

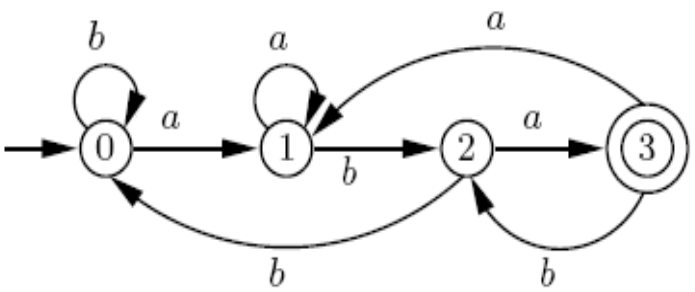
Automat pro hledání řetězce aba



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

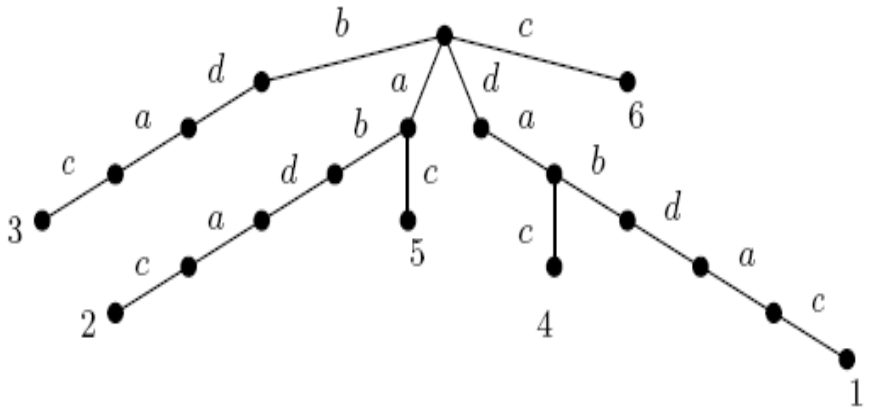


Složitost

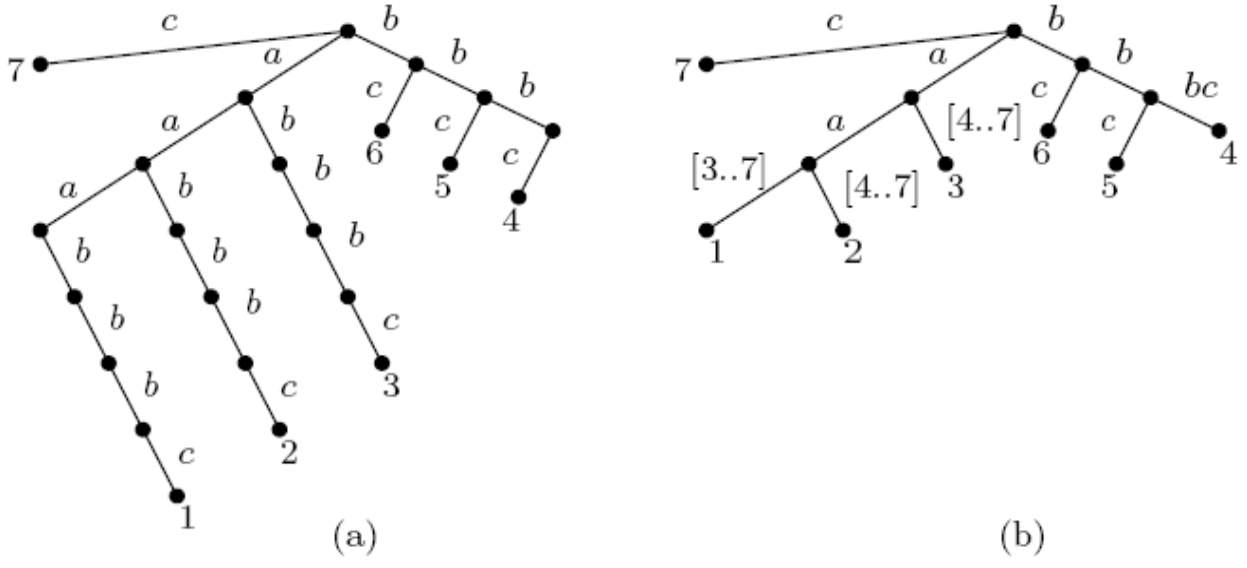
konstrukce: naivní O(m^3); optimální O(||*abeceda*||.m)

hledání: O(n)

Suffixový strom pro řetězec dabdac



Kompaktní suffixový strom pro řetězec aaabbbc



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

Sufixové pole - ukazovatele na polohy suffixů seřazené lexikograficky

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
```

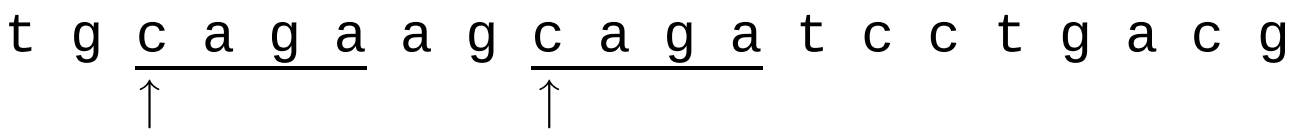
Outline

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
Vylepšení pro maximálně k chyb
Příště
video HHMI

Tandemová a palindromická opakování nesou biologický i praktický význam

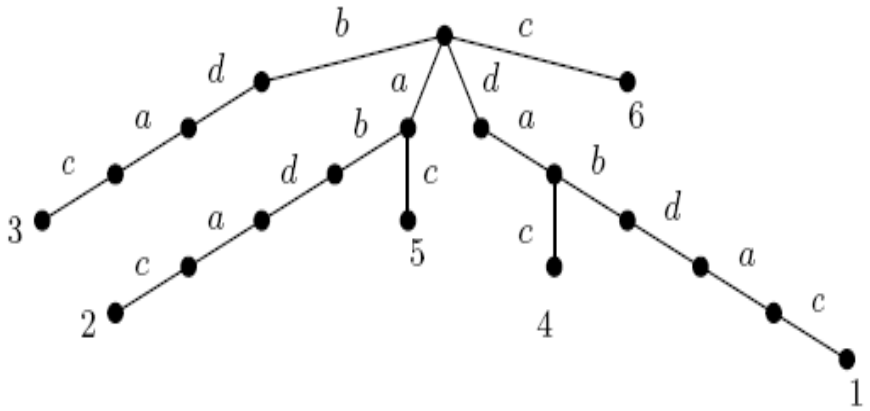
palindrom	možná sekundární struktura DNA nebo RNA
tandem	regulace genů, telomery, identifikace jedinců z DNA

Nejdelší společný prefix dvou pozic



Složitost naivního algoritmu O(n^3)

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



lcp(1,4)=?

Nalezneme větve označené 1 a 4

lcp(1,4)=da

Hledání tandemových opakování

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

Složitost: O(n.$(\log n)^2$+p)

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


```
                    ↓ 8
t g c a g a a g c t t c t g t c t g a c g
a c g t c t t c g a a g a c a g a c t g c
                    ↑ 9*
```

Složitost naivního algoritmu O(n^3)

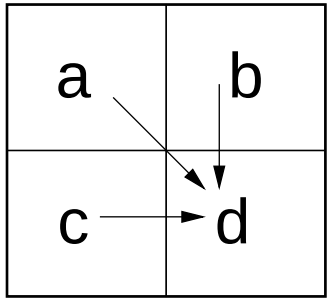
Složitost naivního algoritmu O(nlp) (pro omezenou vzdálenost a délku)
Složitost s použitím suffixových struktur O(n)

Využití DP pro identifikaci palindromů

	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

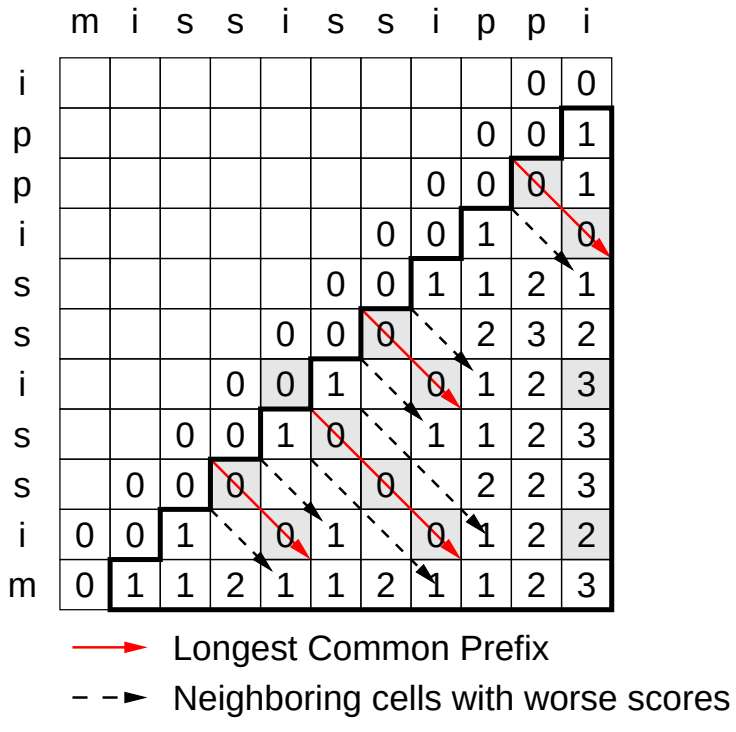
a)



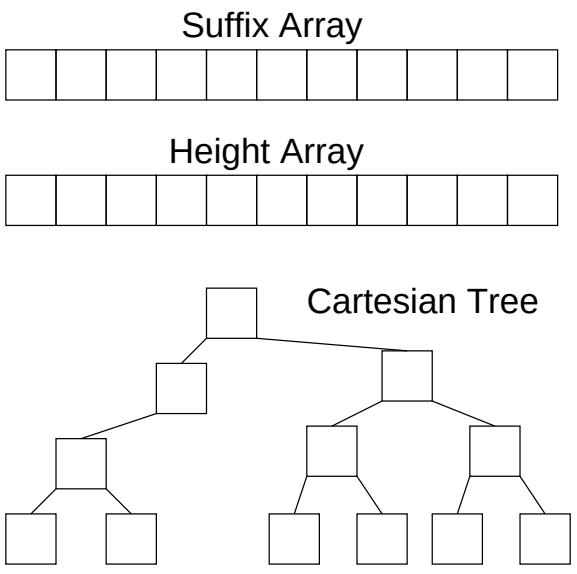
$$d = \min \begin{cases} a & \text{match} \\ a+1 & \text{mismatch} \\ b+1 & \text{insertion} \\ c+1 & \text{deletion} \end{cases}$$

b)

Využití SA a LCP k rychlému postupu po diagonále



a)



b)

Outline

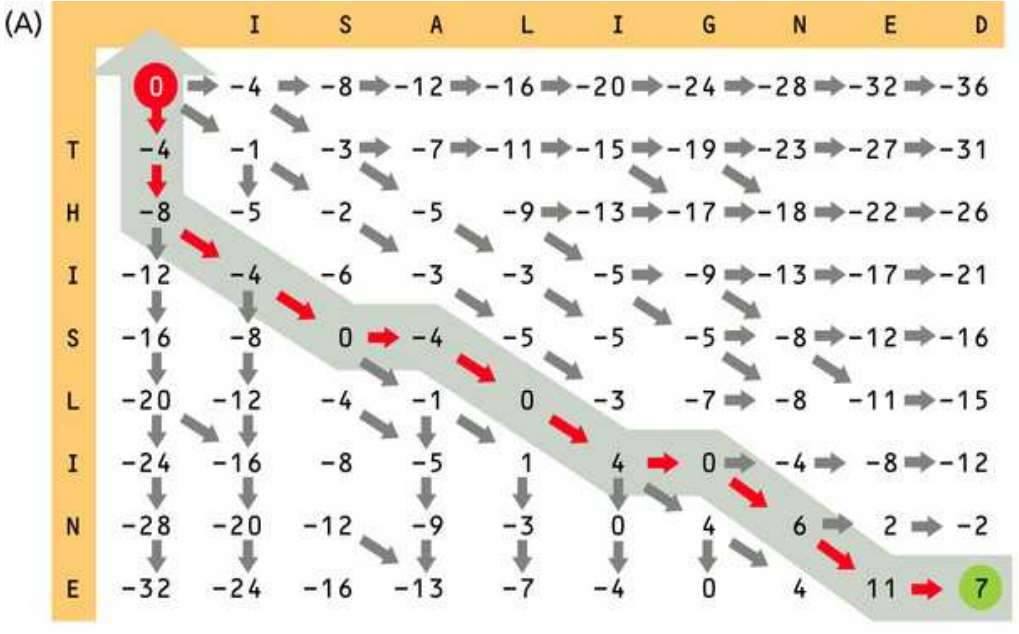
Řetězce a algoritmy na řetězcích
Základní pojmy
Základní algoritmy
Algoritmus využívající analýzu hledaného motivu
Algoritmus využívající nalýzu prohledávaného řetězce
Hledání opakování
Tandemové opakování
Palindromy
Srovnávání dvou sekvencí
Vylepšení pro maximálně k chyb
Příště
video HHMI

Výpočet omezeného počtu buněk v tabulce DP

Stačí počítat 2k+1 diagonál bez ohledu na délku sekvencí

Složitost: O(kn) (naproti O(mn))

Tabulka pro algoritmus dynamického programování



Využití SA a LCP k rychlému postupu po diagonále

Složitost: $O(k^2)$ (viz pou/vzit/'i pro palindromy)

Příště

Video HHMI

Outline

Dodatek

For Further Reading

X