Časté podsekvence, epizodální pravidla

Jiří Kléma

Katedra kybernetiky,
FEL, ČVUT v Praze

http://ida.felk.cvut.cz

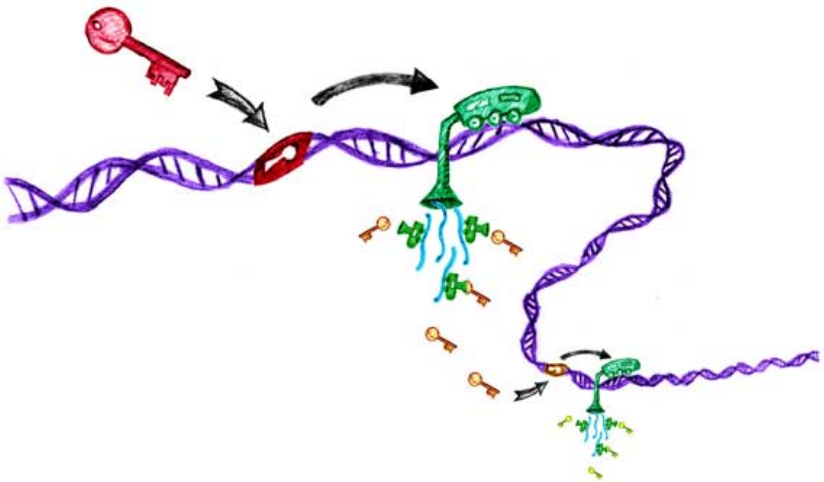
Struktura přednášky

- Motivace pro hledání častých podsekvencí
  - praktické příklady, variabilita v zadání,
- co už vlastně umíme řešit?
  - souvislost s množinami položek, co se změnilo?
  - orientované sekvence, bez šumu/mezer a času,
- proč je to někdy složitější?
  - neorientované sekvence a jejich kanonická forma,
  - (plná) transakční reprezentace a s ní spojené definice,
  - algoritmus GSP (Agrawalovo zobecnění APRIORI),
  - další algoritmy – FreeSpan, PrefixSpan,
- shrnutí
  - kategorizace přístupů dle typu sekvencí a vyhledávaných vzorů,
- STULONG – případová studie.

Časté podsekvence – ilustrace 1: DNA

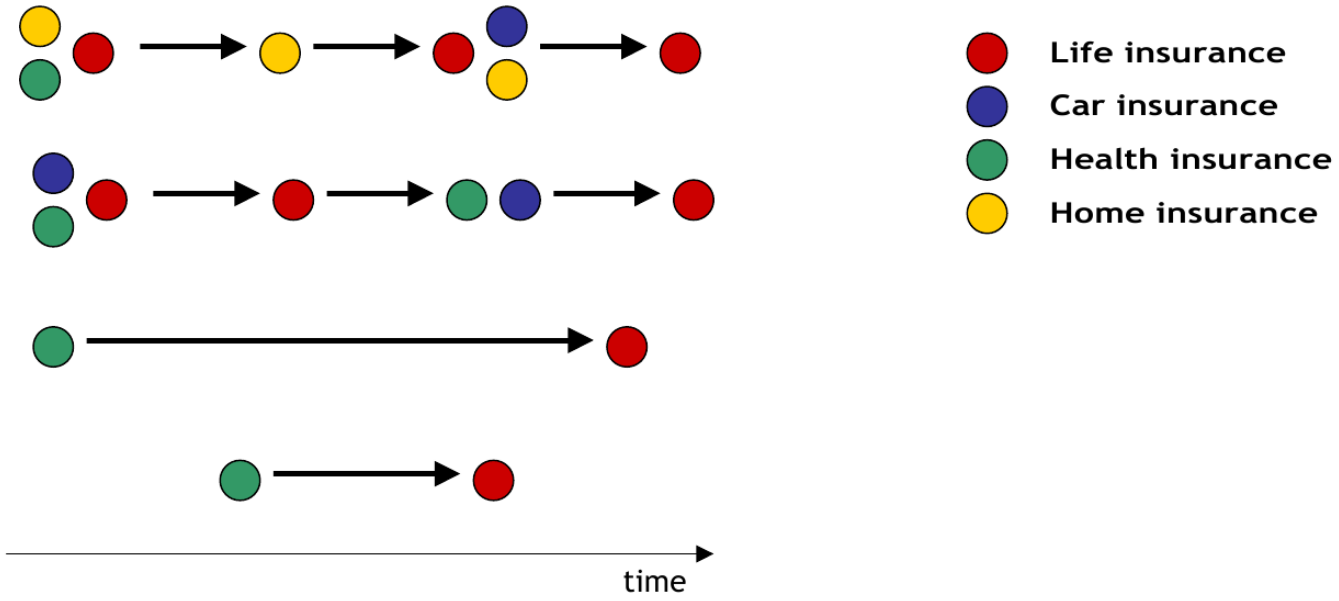
- **motif discovery**
  - hledá krátké sekvenční vzory v souboru nezarovnaných DNA nebo proteinových sekvencí,
  - hledá **diskriminativní** vzory (typické pro jednu třídu sekvencí, neobvyklé ve třídě druhé),
  - tato vlastnost koreluje s biologickým (regulačním) účelem sekvence,
  - transkripční faktor interaguje s DNA přes konkrétní motif,
  - vyhledávání častých podsekvencí je podúlohou,
- událost = nukleotid, řetězec (žádný čas), nejasná orientace DNA.

```
>Z3066
atgacgtgtctgtatattaataagctaacccgcattgagttaaccaataacggattccatacacaatacg
gccaacagagaaagatacctgtgctcacgccattgcttatattggctggttacaatgtgcactatcaatt
ttttaaataa
>Z3067
ttgactgtgaaaattacaggagctacaaaaatgaaccgattctcaaaaactcaaatttatttacattgga
taacgctgcttttcgttgcaataacctatgccgcgatggaactccgtggctggtttcctaaaggtagtag
tacttatctgctgatgcgagaaacacattacaatgcgggtatattcgtttgggtgttaatgttttcacgc
ctgattataaaacaccgttatagtgatccttctattgtgccaccgccacctgcctggcaaatgaaagcgg
cttcgctaatgcacatcatgctttatataaccttccttgcgcttcctctgctggggattgctttgatggc
ttacagtggaaaatcgtggagtttccttggtttcaatgtgtctccctttgttaccccaaacagcgaaatt
aaagcactgataaaaaatattcacgaaacctgggcaaatataggctacttttaatcgtagctcacgctg
gcgcagcactctttcatcactacattcagaaagataatactctgttacgaatgatgcctcgccgcaaata
a
>Z3069
gtggcggagagaggggatttgaacccccggtagagttgcccctactccggttttcgagacctatgctat
gggttaataaaatcaatatattatgtgttttatttggaataaatattctatatttaggattgaaaaatca
gatggttagcatcaaacaacctcagaatattccaagcaaacaggttaaaaataaaactgcacccgaacaa
ttgataacgacagaaaacgccttttccgagccaccaaaattacttcgcatcctatttattgctcacgtta
acgccttgtataactcgagctctccacggtatttaacctctcttctgtttaactataattccaataaatc
tcgtcactga
```



Časté podsekvence – ilustrace 2: pojištění

- událost je uzavření pojistné smlouvy jistého typu,
- v jednom čase může nastat několik událostí,
- sekvence je chronologický sled událostí,
- analogie: orientovaný acyklický graf, délka hrany je dána časem mezi událostmi,
- hledané vzory: obvyklé smluvní posloupnosti uzavřené v rozumném časovém období.



Meyer: Sequence Mining in Marketing.

Časté podsekvence – podobnost s častými množinami položek

- nejprve podobnost na úrovni **reprezentace** úlohy,
- může jít o stejný proces, ale nyní klademe jiné otázky
  - položky: které typy pojištění si lidé objednávají najednou,
  - sekvence: jak se lidé pojišťují v průběhu života,
- transakční reprezentace stále možná a platná (velmi univerzální)
  - musíme zaznamenávat/brát v úvahu více údajů.

Transakce	Položky (typ pojištění)
t_1	domácnost, životní
t_2	auto, domácnost
t_3	penzijní, životní
t_4	cestovní
t_5	penzijní, životní
...	...

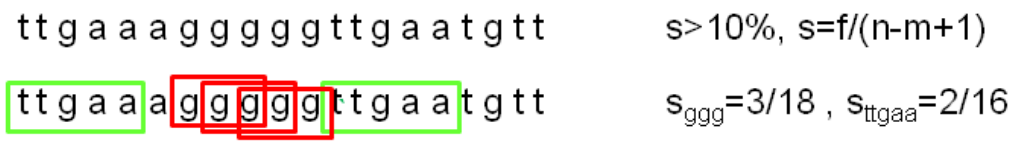
Zákazník	Datum (čas)	Položky (typ pojištění)
z_1	5.10.2003	domácnost, životní
z_1	8.1.2005	cestovní
z_1	3.8.2010	auto, penzijní
z_2	10.10.2003	auto, domácnost
z_2	20.11.2006	penzijní
...	...	...

Časté podsekvence – podobnost s častými množinami položek

- dále podobnost na úrovni **řešení** úlohy,
- APRIORI vlastnost lze zobecnit i pro sekvence:
  Každá podsekvence časté sekvence je častá.
- antimonotónní vlastnost lze podobně převést na monotónní vlastnost:
  Pokud sekvence není častá, žádné z jejích prodloužení není časté.
- modelový algoritmus typu APRIORI pro sekvenční data
  - přímou analogií APRIORI algoritmu pro množiny položek,
  - založený na postupu (neformálně – opakování):
    1. vyhledá triviální časté sekvence (typicky délky 0 nebo 1),
    2. generuje kandidátské sekvence délky o 1 větší,
    3. ověří jejich podporu v transakční databázi,
    4. z množiny kandidátských sekvencí vybere podmnožinu častých sekvencí,
    5. dokud nachází další časté sekvence jde na krok 2.

Časté podřetězce – triviální aplikace APRIORI

- řetězec (string)
  - orientovaná sekvence, ekvidistantní krok, právě jedna položka na transakci,
  - události dány abecedou symbolů, vzor je uspořádaný seznam **sousedních** událostí,
  - $\langle a_1 \dots a_m \rangle$ je podsekvencí $\langle b_1 \dots b_n \rangle$ jestliže $\exists i \; a_1 = b_i \wedge \dots \wedge a_m = b_{i+m}$.
- Příklad: DNA sekvence ($n = 20$, $A = \{a, g, t\}$)



i	C_i	L_i
1	{a}, {g}, {t}	{a}, {g}, {t}
2	(9 vzorů)	{aa}, {ga}, {gg}, {gt}, {tg}, {tt}
3	{aaa}, {gaa}, {gga}, . . . (12 vzorů)	{gaa}, {ggg}, {gtt}, {tga}, {ttg}
4	{gggg}, {gttg}, {tgaa}, {ttga}	{gggg}, {tgaa}, {ttga}
5	{ttgaa}	{ttgaa}

- jak rychle ověřit podporu, tj. nalézt výskyty podsekvence v sekvenci?
  - mj. algoritmy Knuth-Morris-Pratt nebo Boyer-Moore.

Kanonická forma sekvencí

- **kanonické** (standardní) **kódové slovo**
  - jednoznačný způsob zápisu sekvence, vychází z uspořádání abecedy symbolů,
  - obvyklá (ale ne nutná) volba:
    - lexikografické uspořádání abecedy symbolů $a < b < c < \ldots$,
    - lexikograficky (nej)menší kódové slovo kanonické ($bac < cab$),
- orientovaná sekvence
  - jediný možný výklad (způsob čtení), každá (pod)sekvence kanonickým kódovým slovem,
- neorientovaná sekvence
  - nutnost volit mezi dvěma způsoby čtení = kódovými slovy,
  - rutinní aplikace lexikografického uspořádání není možná,
  - v prostoru kanonických slov přestává platit **prefixová vlastnost**:
    - prefix kanonického slova je také kanonickým slovem,

sekvence	kanonický zápis	prefix	kanonický zápis
bab	bab	ba	ab
$cabd$	$cabd$	cab	bac

  - musíme najít jiný způsob zápisu kódových slov.

Kanonická forma neorientované sekvence

- Zápis kanonických kódových slov s prefixovou vlastností
  - odlišně nakládáme se sekvencemi sudé a liché délky,
  - kódové slovo tvoříme od středu sekvence,

	sudá délka	lichá délka
sekvence	$a_m\ a_{m-1}\ \ldots\ a_2\ a_1\ b_1\ b_2\ \ldots\ b_{m-1}\ b_m$	$a_m\ a_{m-1}\ \ldots\ a_2\ a_1\ a_0\ b_1\ b_2\ \ldots\ b_{m-1}\ b_m$
kódové slovo	$a_1\ b_1\ a_2\ b_2\ \ldots\ a_{m-1}\ b_{m-1}\ a_m\ b_m$	$a_0\ a_1\ b_1\ a_2\ b_2\ \ldots\ a_{m-1}\ b_{m-1}\ a_m\ b_m$
kódové slovo	$b_1\ a_1\ b_2\ a_2\ \ldots\ b_{m-1}\ a_{m-1}\ b_m\ a_m$	$a_0\ b_1\ a_1\ b_2\ a_2\ \ldots\ b_{m-1}\ a_{m-1}\ b_m\ a_m$

- kanonické je to lexikograficky menší z kódových slov v tabulce,
- sekvenci **prodloužíme** přidáním
  - dvojice $a_{m+1}\ b_{m+1}$ nebo $b_{m+1}\ a_{m+1}$,
  - jeden symbol je přidán na začátek, druhý na konec.
- příklad

sudá délka			lichá délka		
sekvence	kódová slova		sekvence	kódová slova	
at	at	ta	ule	lue	leu
$data$	$atda$	$taad$	$rules$	$luers$	$leusr$

Kanonická forma neorientované sekvence – prefixovost

- **Důkaz prefixovosti** nové reprezentace **sporem**

  1. předpokládejme, že prefixová vlastnost neplatí,
  2. pak existuje kanonické slovo $w_m = a_1\ b_1\ a_2\ b_2\ \ldots\ a_{m-1}\ b_{m-1}\ a_m\ b_m$,
  3. jehož prefix $w_{m-1} = a_1\ b_1\ a_2\ b_2\ \ldots\ a_{m-1}\ b_{m-1}$ není kanonickým slovem,
  4. důsledkem je $w_m < v_m$, kde $v_m = b_1\ a_1\ b_2\ a_2\ \ldots\ b_{m-1}\ a_{m-1}\ b_m\ a_m$,
  5. a $v_{m-1} < w_{m-1}$, kde $v_{m-1} = b_1\ a_1\ b_2\ a_2\ \ldots\ b_{m-1}\ a_{m-1}$,
  6. avšak $v_{m-1} < w_{m-1} \Rightarrow v_m < w_m$
     - protože v_{m-1} je prefixem v_m a w_{m-1} je prefixem w_m,
  7. $v_m < w_m$ z kroku 6 je ve sporu s $w_m < v_m$ z kroku 4 $\square$.

Kanonická forma neorientované sekvence – efektivita

- dvě možná kódová slova lze tvořit a srovnávat v $\mathcal{O}(m)$,
- zavedením **příznaku symetrie** sekvence lze obě operace provádět v $\mathcal{O}(1)$

$$s_m = \bigwedge_{i=1}^{m} (a_i = b_i)$$

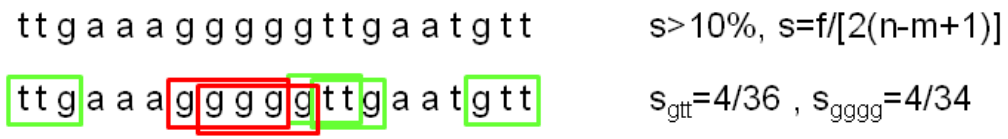
- příznak symetrie udržujeme v konstantním čase operací

$$s_{m+1} = s_m \ \wedge \ (a_{m+1} = b_{m+1})$$

- prodloužení sekvence je **přípustné** v závislosti na hodnotě příznaku:
  - jestliže $s_m =$ true, musí platit $a_{m+1} \leq b_{m+1}$,
  - jestliže $s_m =$ false, jakýkoli vztah mezi a_{m+1} and b_{m+1} je přijatelný.
- počátek vytváření
  - sudá délka: prázdná sekvence, $s_0 = 1$,
  - lichá délka: všechny časté symboly abecedy, $s_1 = 1$,
- postup garantuje výhradně kanonická prodloužení sekvence.

Časté podsekvence – aplikace APRIORI na neorientované sekvence

- uvažujme neorientované sekvence, jinak stejná formalizace jako dosud
  - $\langle a_1 \dots a_m \rangle$ je podsekvencí $\langle b_1 \dots b_n \rangle$ jestliže:
    $\exists i \; a_1 = b_i \wedge \dots \wedge a_m = b_{i+m}$,
    $\exists i \; a_1 = b_{i+m} \wedge \dots \wedge a_m = b_i$.
- Příklad: DNA sekvence ($n = 20$, $A = \{a, g, t\}$)



s>10%, s=f/[2(n-m+1)]

s_{gtt}=4/36 , s_{gggg}=4/34

i	C_i	L_i
0	{}	{}
1	{a}, {g}, {t}	{a}, {g}, {t}
2	**{aa}, {ag}, {at}, {gg}, {gt}, {tt}**	**{aa}, {gg}, {gt}, {tt}**
3	{aaa}, ~~{aag}~~, ~~{aat}~~, ~~{gag}~~, ~~{gat}~~, ~~{tat}~~, ~~{aga}~~, ~~{agg}~~, ~~{agt}~~, {ggg}, {ggt}, ~~{tgt}~~, ~~{ata}~~, ~~{atg}~~, ~~{att}~~, ~~{gtg}~~, {gtt}, {ttt}	{ggg}, {gtt}
4	**~~{aaaa}~~, ~~{aaag}~~, {aaat}, {gaag}, {gaat}, {taat}, ~~{agta}~~, ~~{agtg}~~, ~~{ggta}~~, {agtt}, ~~{tgta}~~, ~~{ggtg}~~, ~~{ggtt}~~, ~~{tgtg}~~, . . . celkem ~~27~~ (1) vzorů**	**{gggg}**

Zobecněná definice podsekvence pro transakční reprezentaci

- Položky: $I = \{i_1, i_2, \ldots, i_m\}$,
- množiny položek: $(x_1, x_2, \ldots, x_k) \subseteq I$, $k \geq 1$, $x_i \in I$,
- sekvence: $\langle s_1, \ldots, s_n \rangle$, $s_i = (x_1, x_2, \ldots, x_k) \subseteq I$, $s_i \neq \emptyset$, $x_1 < x_2 < \ldots < x_k$,
  - uspořádaný seznam elementů, elementy jsou množiny položek,
  - kanonická reprezentace: lexikografické uspořádání položek v každé z množin,
  - př.: $\langle a(abc)(ac)d(cf) \rangle$, zjednodušení zápisu: $(x_i) \sim x_i$,
- délka sekvence l
  - dána počtem instancí položek v sekvenci, l-sekvence obsahuje právě l instancí,
  - př.: $\langle a(abc)(ac)d(cf) \rangle$ je 9-sekvencí,
- α je podsekvencí β, β je nadsekvencí α: $\alpha \sqsubseteq \beta$
  - $\alpha = \langle a_1, \ldots, a_n \rangle$, $\beta = \langle b_1, \ldots, b_m \rangle$, $\exists 1 \leq j_1 \leq \ldots \leq j_n \leq m, \forall i = 1 \ldots n : a_i \subseteq b_{j_i}$,
  - př.: $\langle a(bc)df \rangle \sqsubseteq \langle a(abc)(ac)d(cf) \rangle$, $\langle d(ab) \rangle \not\sqsubseteq \langle a(abc)(ac)d(cf) \rangle$
- databáze sekvencí: $S = \{\langle sid_1, s_1 \rangle \ \ldots, \langle sid_k, s_k \rangle\}$
  - jde o množinu uspořádaných dvojic identifikátor sekvence a sekvence.

Problém vyhledávání podsekvencí v transakční reprezentaci

- Podpora α v databázi S
  - počet sekvencí s z S pro něž platí: $\alpha \sqsubseteq s$,
- Problém vyhledání častých podsekvencí v transakční databázi
  - vstup: S a s_{min} – minimální podpora,
  - výstup: úplná množina častých sekvenčních vzorů (podsekvencí s alespoň prahovou četností).

Id	Sekvence
10	$\langle a(abc)(ac)d(cf)\rangle$
20	$\langle (ad)c(bc)(ae)\rangle$
30	$\langle (ef)(ab)(df)cb\rangle$
40	$\langle eg(af)cbc\rangle$

Id	Čas	Položky
10	t_1	a
10	t_2	a, b, c
10	t_3	a, c
10	t_4	d
10	t_5	c, f

l	sekvenční vzor (s_{min}=2)
3	$\langle a(bc)\rangle$, $\langle aba\rangle$, $\langle abc\rangle$, $\langle (ab)c\rangle$, $\langle (ab)d\rangle$, $\langle (ab)f\rangle$, $\langle aca\rangle$, $\langle acb\rangle$, $\langle acc\rangle$, $\langle adc\rangle$, ...
4	$\langle a(bc)a\rangle$, $\langle (ab)dc\rangle$, ...

Pei, Han et al.: PrefixSpan: Mining Sequential Patterns by Prefix-Projected Growth.

GSP: Generalized Sequential Patterns [Agrawal, Srikant, 1996]

- aplikuje myšlenku APRIORI na sekvenční data,
- klíčová je otázka generování kandidátských sekvenčních vzorů
  - lze jej rozdělit na dva kroky
    1. **spojování** (join)
       * l-sekvenci vytvoříme spojením dvou (l-1)-sekvencí,
       * lze spojit (l-1)-sekvence shodné po vypuštění první položky z jedné a poslední z druhé,
    2. **prořezávání** (prune)
       * vynecháme každou l-sekvenci, která obsahuje (l-1)-podsekvenci, která není častá,

L_3	C_4	
	po spojení	po prořezání
$\langle(ab)c\rangle$, $\langle(ab)d\rangle$, $\langle a(cd)\rangle$, $\langle(ac)e\rangle$, $\langle b(cd)\rangle$, $\langle bce\rangle$	$\langle(ab)(cd)\rangle$ $\langle(ab)ce\rangle$	$\langle(ab)(cd)\rangle$

Agrawal, Srikant: Mining Sequential Patterns: Generalizations and Performance.

Příklad: GSP, s_{min}=2

Id	Sekvence
10	$\langle (bd)cb(ac) \rangle$
20	$\langle (bf)(ce)b(fg) \rangle$
30	$\langle (ah)(bf)abf \rangle$
40	$\langle (be)(ce)d \rangle$
50	$\langle a(bd)bcb(ade) \rangle$

- $s(\langle g \rangle) = s(\langle h \rangle) = 1 < s_{min}$

  (vynechá téměř polovinu z 92 možných 2-kandidátů),
- $\langle (bd)cba \rangle \sqsubseteq s_{10} \wedge \langle (bd)cba \rangle \sqsubseteq s_{50}$

  (vytvořen z $\langle (bd)cb \rangle$ a $\langle dcba \rangle$),

  (musí být časté i vzory $\langle (bd)ba \rangle$, $\langle (bd)ca \rangle$ a $\langle bcba \rangle$).

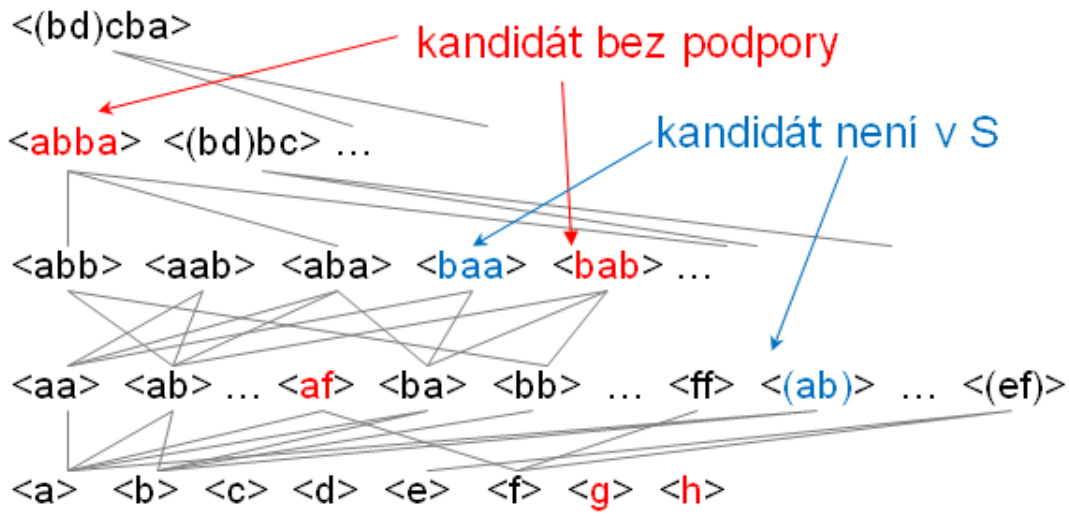
s5: 1 kand. 1 5-vzor

s4: 8 kand. 6 4-vzorů

s3: 46 kand. 19 3-vzorů

s2: 51 kand., 19 2-vzorů

s1: 8 kand., 6 1-vzorů



Nevýhody APRIORI přístupu

- jde o přístup generuj (join krok) a testuj (prune krok),
- problémy zmiňované v souvislosti s častými množinami položek přetrvávají a zesilují
  1. generuje velké množství kandidátských vzorů
     - evidentní už pro 2-sekvence: $m \times m + \frac{m(m-1)}{2} \rightarrow \mathcal{O}(m^2)$
       (u množin položek to byl pouze druhý člen, tedy zhruba třetina),
  2. vyžaduje velký počet průchodů databází
     - pro každou délku vzoru jeden průchod,
     - počet přístupů dán max. délkou vzoru $\leq max(|s|, s \in S)$ (obvykle $>> m$),
       (maximální délka množiny položek je m a tedy nejvýše m přístupů),
  3. vyhledání dlouhých sekvenčních vzorů je problematické
     - celkový počet možných kandidátských vzorů je exponenciální vzhledem k délce vzoru,
       (roste stejně rychle jako u množin položek, opět ale problém $\leq max(|s|, s \in S) >> m$).
- omezeny odlišným přístupem v algoritmex FreeSpan a PrefixSpan.

FreeSpan [Han, Pei, Yin, 2000], $s_{min} = 2$

- využívá **rekurzivního** přístupu rozděl a panuj
  - rozhoduje na základě sestupně seřazeného seznamu častých položek,
    - $f\text{-}list = \langle (a:4), (b:4), (c:4), (d:3), (e:3), (f:3) \rangle$,
    - $(g:1)$ není časté,
  - sekvenční vzory dělí do disjunktních skupin
    - vzory obsahující pouze nejčastější položku,
    - vzory obsahující druhou nejčastější položku a žádnou méně častou, atd.
  - tvoří projekční databáze sekvencí (jednu pro každou skupinu)
    - ze sekvencí odstraní všechny položky, které nejsou skupinou postiženy,
    - odstraní sekvence, v nichž není položka, která ve vzoru být musí.
- podproblémy mají méně položek (zpočátku), obsahují méně sekvencí (ke konci).

Id	Sekvence	a-projekce	b-projekce	...	f-projekce
10	$\langle a(abc)(ac)d(cf) \rangle$	$\langle aaa \rangle$	$\langle a(ab)a \rangle$	...	$\langle a(abc)(ac)d(cf) \rangle$
20	$\langle (ad)c(bc)(ae) \rangle$	$\langle aa \rangle$	$\langle aba \rangle$	...	
30	$\langle (ef)(ab)(df)cb \rangle$	$\langle a \rangle$	$\langle (ab)b \rangle$	...	$\langle (ef)(ab)(df)cb \rangle$
40	$\langle eg(af)cbc \rangle$	$\langle a \rangle$	$\langle ab \rangle$	...	$\langle e(af)cbc \rangle$

Pei, Han et al.: PrefixSpan: Mining Sequential Patterns by Prefix-Projected Growth.

PrefixSpan [Pei, Han et al., 2001]

- založen na podobné myšlence, efektivnější než předchůdce FreeSpan,
- projekce na základě **prefixového** výskytu podsekvence (u FreeSpan libovolného výskytu)
  - umožňuje účinnější rozklad databáze,
- $\beta = \langle s'_1, \ldots, s'_m \rangle$ je prefixem $\alpha = \langle s_1, \ldots, s_n \rangle$ pokud:

  (1) $m \leq n$, (2) $\forall i \leq m-1\ s'_i = s_i$, (3) $s'_m \subseteq s_m$,
  (4) $\forall$ položky z $(s_m - s'_m) >$ $\forall$ položky z s'_m,
  - př.: $\langle a \rangle$, $\langle aa \rangle$, $\langle a(ab) \rangle$ a $\langle a(abc) \rangle$ jsou prefixy $\langle a(abc)(ac)d(cf) \rangle$,
  - př.: $\langle ab \rangle$, $\langle a(bc) \rangle$ nejsou prefixy $\langle a(abc)(ac)d(cf) \rangle$,
- neformálně: postfix je doplňkem prefixu
  - př.: prefix $\langle a \rangle$ má vzhledem k $\langle a(abc)(ac)d(cf) \rangle$ postfix $\langle (abc)(ac)d(cf) \rangle$,
  - př.: prefix $\langle aa \rangle$ má vzhledem k $\langle a(abc)(ac)d(cf) \rangle$ postfix $\langle (\_bc)(ac)d(cf) \rangle$,
- $\alpha' \sqsubseteq \alpha$ je projekcí α vzhledem k prefixu $\beta \sqsubseteq \alpha$, pokud:

  (1) α' má prefix β,
  (2) neexistuje α'', která je nadsekvencí α' (tj. $\alpha' \sqsubset \alpha''$), podsekvencí α a má prefix β,
  - př.: projekcí $\langle a(abc)(ac)d(cf) \rangle$ vzhledem k prefixu $\langle (ac)d \rangle$ je $\langle (ac)d(cf) \rangle$.

PrefixSpan – algoritmus, příklad ($s_{min} = 2$)

- PrefixSpan: vstup S a s_{min}
  1. $i = 1$, init projekční prefixové databáze $S|_{\alpha_0} = S|_{\emptyset} = S$,
  2. opakuj pro všechny projekční prefixové databáze $S|_{\alpha_{i-1}}$
     (a) najdi časté i-vzory (nadprahová podpora v $\alpha_{i-1} \cdot S|_{\alpha_{i-1}}$),
     (b) pokud je množina i-vzorů neprázdná
        i. rozděl stavový prostor dle i-vzorů (α_i) jako prefixů
           vznikne množina projekčních databází $S|_{\alpha_i} = (\alpha_{i-1} \cdot S|_{\alpha_{i-1}})|_{\alpha_i}$,
        ii. i=i+1 a jdi na krok (2).

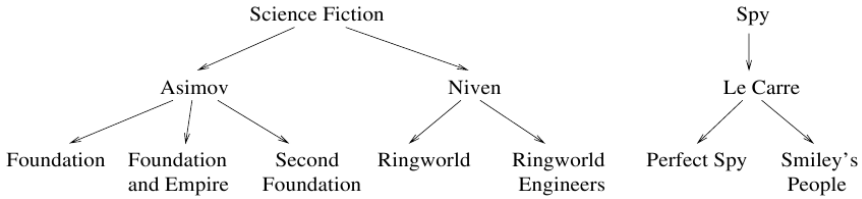
Id	Sekvence
10	$\langle a(abc)(ac)d(cf)\rangle$
20	$\langle (ad)c(bc)(ae)\rangle$
30	$\langle (ef)(ab)(df)cb\rangle$
40	$\langle eg(af)cbc\rangle$

Prefix	Projekční databáze (postfixy) resp vzory
$\langle a\rangle$	$\langle (abc)(ac)d(cf)\rangle$, $\langle (\_d)c(bc)(ae)\rangle$, $\langle (\_b)(df)cb\rangle$, $\langle (\_f)cbc\rangle$
	2-vzory: $\langle aa\rangle$:2, $\langle ab\rangle$:4, $\langle ac\rangle$:4, $\langle ad\rangle$:2, $\langle af\rangle$:2, $\langle (ab)\rangle$:2
$\langle b\rangle$	$\langle (\_c)(ac)d(cf)\rangle$, $\langle (\_c)(ae)\rangle$, $\langle (df)cb\rangle$, $\langle c\rangle$
	2-vzory: $\langle ba\rangle$:2, $\langle bc\rangle$:3, $\langle (bc)\rangle$:2, $\langle bd\rangle$:2, $\langle bf\rangle$:2
$\langle aa\rangle$	$\langle (\_bc)(ac)d(cf)\rangle$, $\langle (\_e)\rangle$
	STOP (žádné 3-vzory)
$\langle (ab)\rangle$	$\langle (\_c)(ac)d(cf)\rangle$, $\langle (df)cb\rangle$
	3-vzory: $\langle (ab)c\rangle$:2, $\langle (ab)d\rangle$:2, $\langle (ab)f\rangle$:2

Zobecnění problému vyhledávání častých sekvencí

- definice podsekvence (slajd 13) stále není dostatečně obecná pro řadu praktických problémů,
- př.: nákupy knih

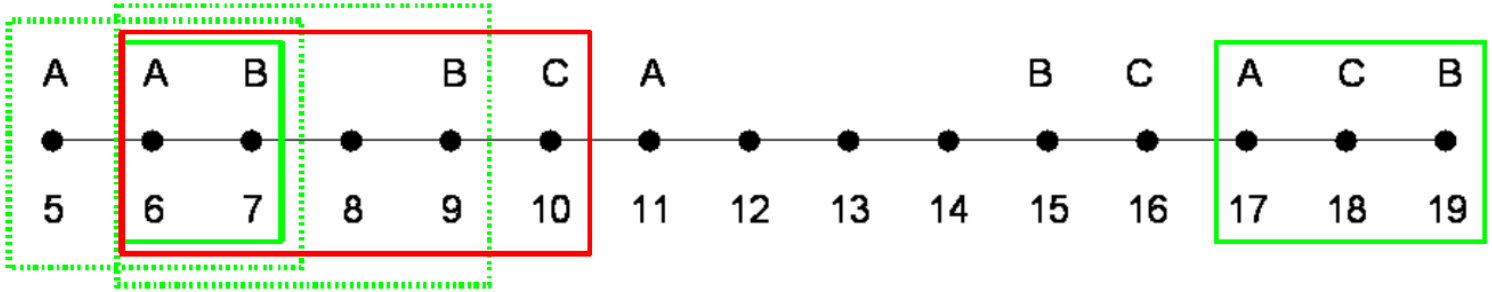
ID	Čas	Položky
C1	1	Ringworld
C1	2	Foundation
C1	15	Ringworld Engineers, Second Foundation
C2	1	Foundation, Ringworld
C2	20	Foundation and Empire
C2	50	Ringworld Engineers



- přínos GSP je i v zavedení několika rozšíření
  1. zavedení časových omezení
     - sousední elementy sekvence nesmí být příliš vzdálené (MaxGap) nebo blízké (MinGap),
  2. rozšířená definice transkace
     - položky zařadí do jediné transakce pokud mají blízké časové značky,
     - klouzavé okno, parameterm je jeho šířka WinSize,
  3. taxonomie položek
     - orientovaný acyklický graf definuje hierarchii pojmů vystavěnou nad položkami.

Epizodální pravidla

- analogie asociačních pravidel,
- umožňují předvídat další vývoj posloupnosti na základě vzorů,
- př.: jediná sekvence, jediná položka na pozici, MaxGap=3



- S je sekvence, $\alpha = \langle AB \rangle$ a $\beta = \langle ABC \rangle$ jsou její podsekvence,
- α je prefixem β,
- **epizodální pravidlo** je pravděpodobnostní implikace
  - $\alpha \Rightarrow postfix(\beta, \alpha)$, tj. $\langle AB \rangle \Rightarrow \langle C \rangle$
- podobně jako u asociačních je vedle podpory parametrem zadání spolehlivost
  - $conf(\alpha \Rightarrow postfix(\beta, \alpha)) = \frac{s(\beta,S,3)}{s(\alpha,S,3)} = \frac{1}{2}$.

Časté podsekvence – shrnutí, kategorizace

- Rozlišujeme různé typy problémů v závislosti na typu sekvencí
  - jedna vs více sekvencí v databázi
    - ovlivní definici podpory,
  - orientované vs neorientované sekvence
    - ovlivní kanonickou reprezentaci,
  - jediná položka vs více položek na jediné pozici v sekvenci
    - ovlivní složitost řešení,
  - existence omezení
    - mj. šířka okna pro definici transakce,
    - MinGap a MaxGap pro definici sekvence,
    - rozšiřují praktickou použitelnost, mírně zesložiťují řešení,
  - existence taxonomií položek
    - rozšiřují praktickou použitelnost, mírně zesložiťují řešení,
  - položky nebo (označené) intervaly
    - interval: $I = (start, end, label)$.
- příště: od sekvenčních ke strukturálním vzorům (stromy/grafy).

Doporučené doplňky – zdroje přednášky

:: Četba

- Agrawal, Srikant: **Mining Sequential Patterns.**
- Agrawal, Srikant: **MSPs: Generalizations and Performance.**
  - od APRIORI k jeho sekvenčním verzím AprioriAll a GSP,
  - http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.95.2818&rep=rep1&type=pdf,
- Pei, Han et al.: **PrefixSpan: Mining Sequential Patterns by Prefix-Projected Growth.**
  - FreeSpan (idea) a PrefixSpan algoritmy, srovnání efektivity s GSP,
  - http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.12.7211,
- Mannila et al.: **Discovery of Frequent Episodes in Event Sequences.**
  - epizodální pravidla,
  - http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.49.3594,
- Borgelt: **Frequent Pattern Mining.**
  - neorientované sekvence,
  - http://www.borgelt.net/teach/fpm/slides.html.