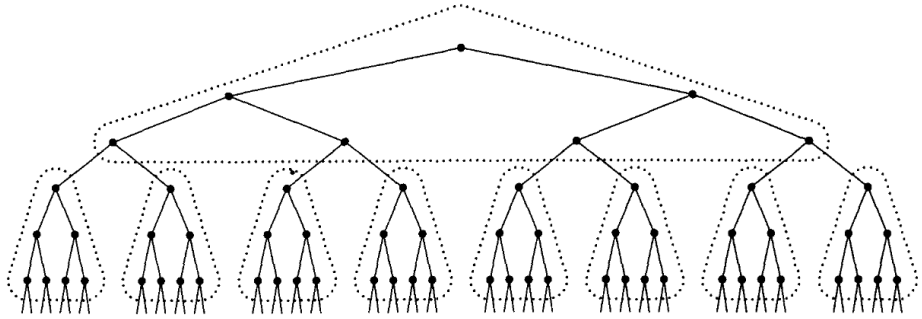
Mutliway trees Kony neví jak se to řekne česky.

Vyhledávání:

- interní
- externí – uvažujeme, že získáváme informaci z velkých dat, které se vyskytují na uložištích s přímým přístupem.

Stromy se hodí k externímu vyhledávání, pokud zvolíme vhodnou formu jejich reprezentace.
Při použití naivního přístupu potřebujeme asi $\lg n$ přístupů.
(přístup = načtení uzlu).
Takže pro 1 000 000 uzlů 20 přístupů na disk.

ukradený obrázek hned na prvním slajdu.



Pokud ovšem budeme načítat 7 uzlů současně, budeme potřebovat jen asi 1/3 přístupů.
Seskupení uzlů do *stránek* (pages) o sedmi uzlech z toho vlastně udělá oktonární strom.

- pokud budeme používát 128-ární větvení, stačí při milionu prvku asi 3 přístupy.
- kořenovou stránku můžeme udržovat stále v interní paměti.
- Jen dvě reference v interní paměti jsou potřeba (kořen a aktuální uzel), takže celkově 254 klíčů.

Nechceme mít libovolnou velikost stránky, protože

- velikost interní paměti je omezená,
- načíst velkou stránku trvá dlouho.

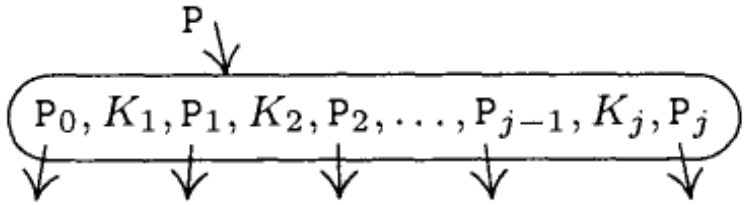
B-stromy

1970 – R. Bayer & E. McGreight, M. Kaufman

Binární strom řádu m je strom, který splňuje následující vlastnosti

1. Každý uzel má nejvýše m uzlů.
2. Každý uzel, mimo kořene a listů, má alespoň $m/2$ potomků.
3. Kořen má alespoň 2 potomky (pokud není listem)
4. Všechny listy jsou na stéjné úrovni a nenesou řádné informace.
5. Všechny nelistové uzly s k-uzly obsahují $k - 1$ klíčů.

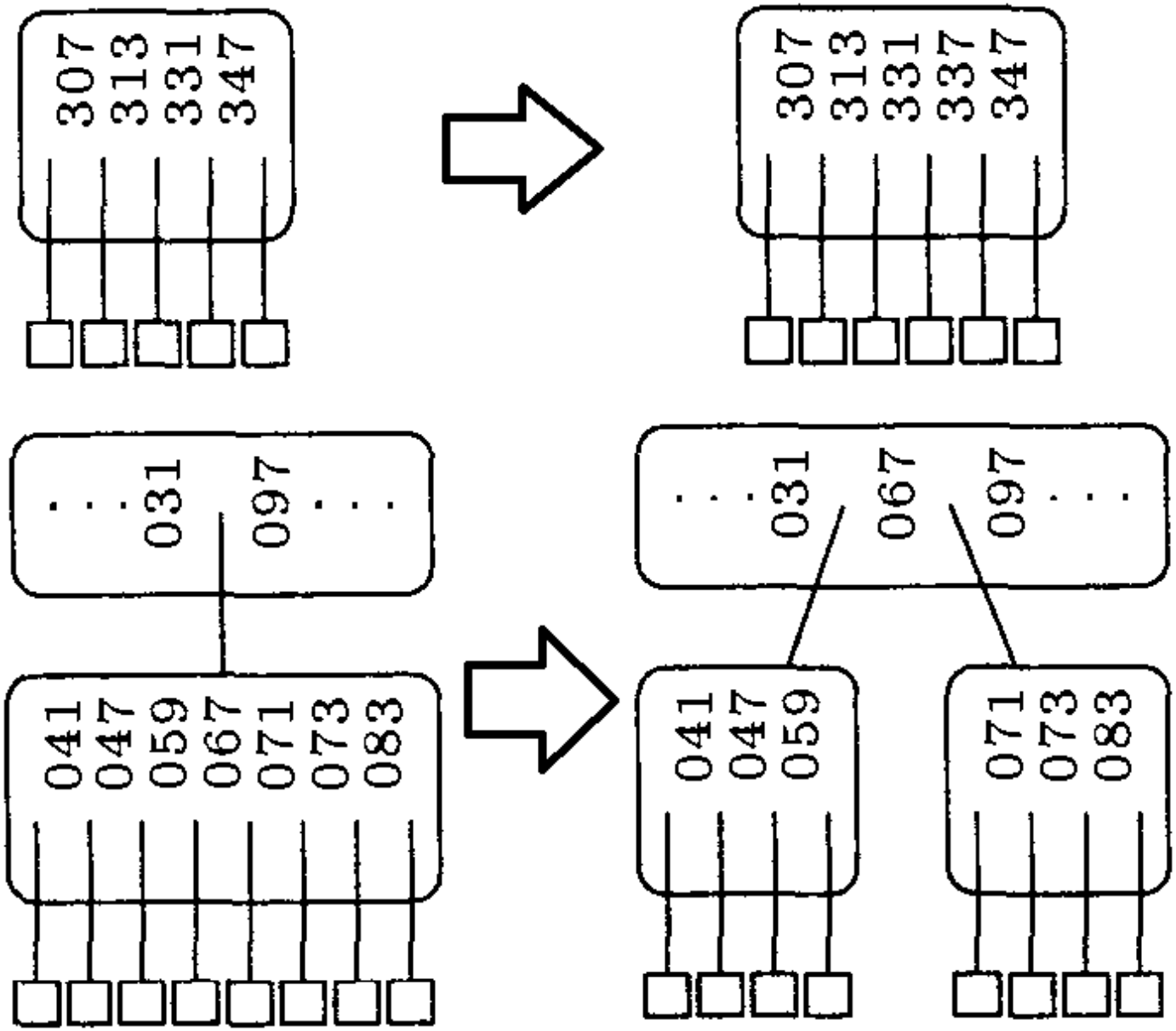
Uzel, který obsahuje j klíčů, $j + 1$ ukazatelů může být reprezentován jako



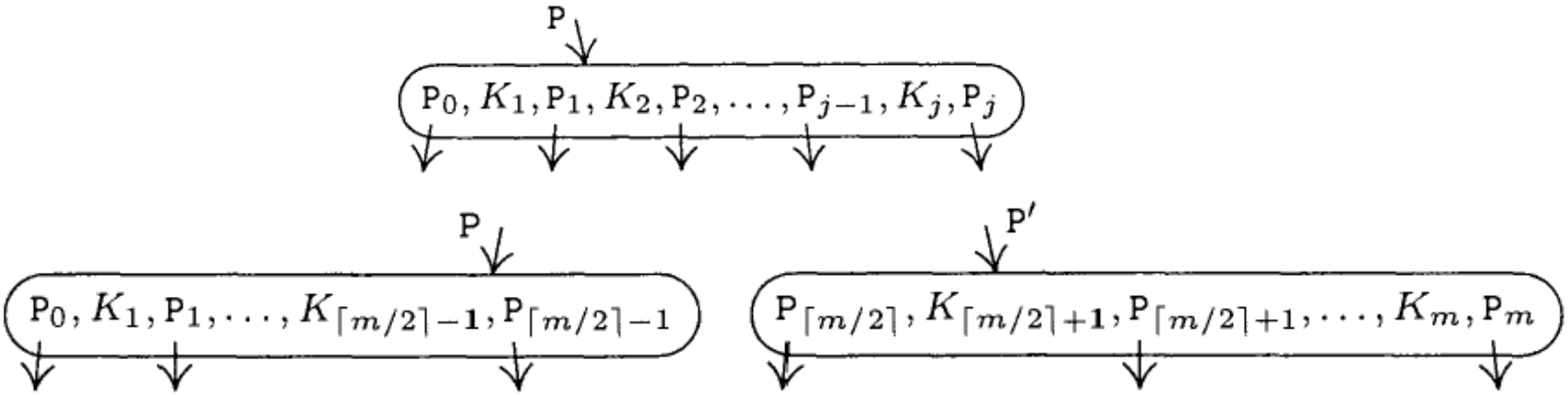
kde $K_1 < K_2 < \cdots < K_j$ a ukazatel P_i ukazuje na podstrom s klíči mezi K_i a K_{i+1}.

Vyhledávání je přímočaré

Vkladání je snadné (a studentům známé)



Rozlomení (split) uzlu (obecně),



- vkládání zachovává vlastnosti B-stromu

Čas vyhledávání

Kolik uzlů musíme načíst v nejhořím případě, když hledáme v B-stromu řádu m.

Předpokládejme, že je v něm n klíčů, a tedy n+1 listů je na úrovni l.

Pak počet uzlů na úrovni 1,2,3. . . je nejméně

$2, 2\lceil m/2 \rceil, 2\lceil m/2 \rceil^2, \ldots$

takže

$$n + 1 \geq 2\lceil m/2 \rceil^{l-1}$$

A tedy

$$l \leq 1 + \log_{m/2} \left(\frac{n+1}{2} \right)$$

Potřebujeme přistoupit k nejvýše l uzlům.

Když je nový klíč vkládán, můžeme rozlomit až l uzlů.

Pokud máme p interních uzlů, existuje nejméně $1 + (\lceil m/2 \rceil - 1)(p - 1)$ klíčů, takže

$$p \leq 1 + \frac{n - 1}{\lceil m/2 \rceil - 1}$$

během vložení.

Z toho plyne, že průměrný počet splitů při budování stromu o n klíčích je méně, než

$$\frac{1}{\lceil m/2 \rceil - 1}$$

pro každé vkládání.

Varianty

všechny ukazatele v hloubce $l - 1$ jsou `NIL`, žádné jine nejsou `NIL`.
Můžeme odstranit všechny tyto ukazatele a použít jinou hodnotu m pro nejnižší patro.

Vlastně bychom mohli mít různé m pro každé patro a zobecnit na B-strom řádu $m_1, m_2, \ldots$. Algoritmus vkládání zůstane v podstatě stejný.

Totéž můžem jeste rozvinout a mít úplně jiný formát uzlu v každém patře.
Někdy klíče tvoří jenom malou část, pak by byla chyba ukládat celé záznamy v uzlech blízko kořene – snižuje se tím m

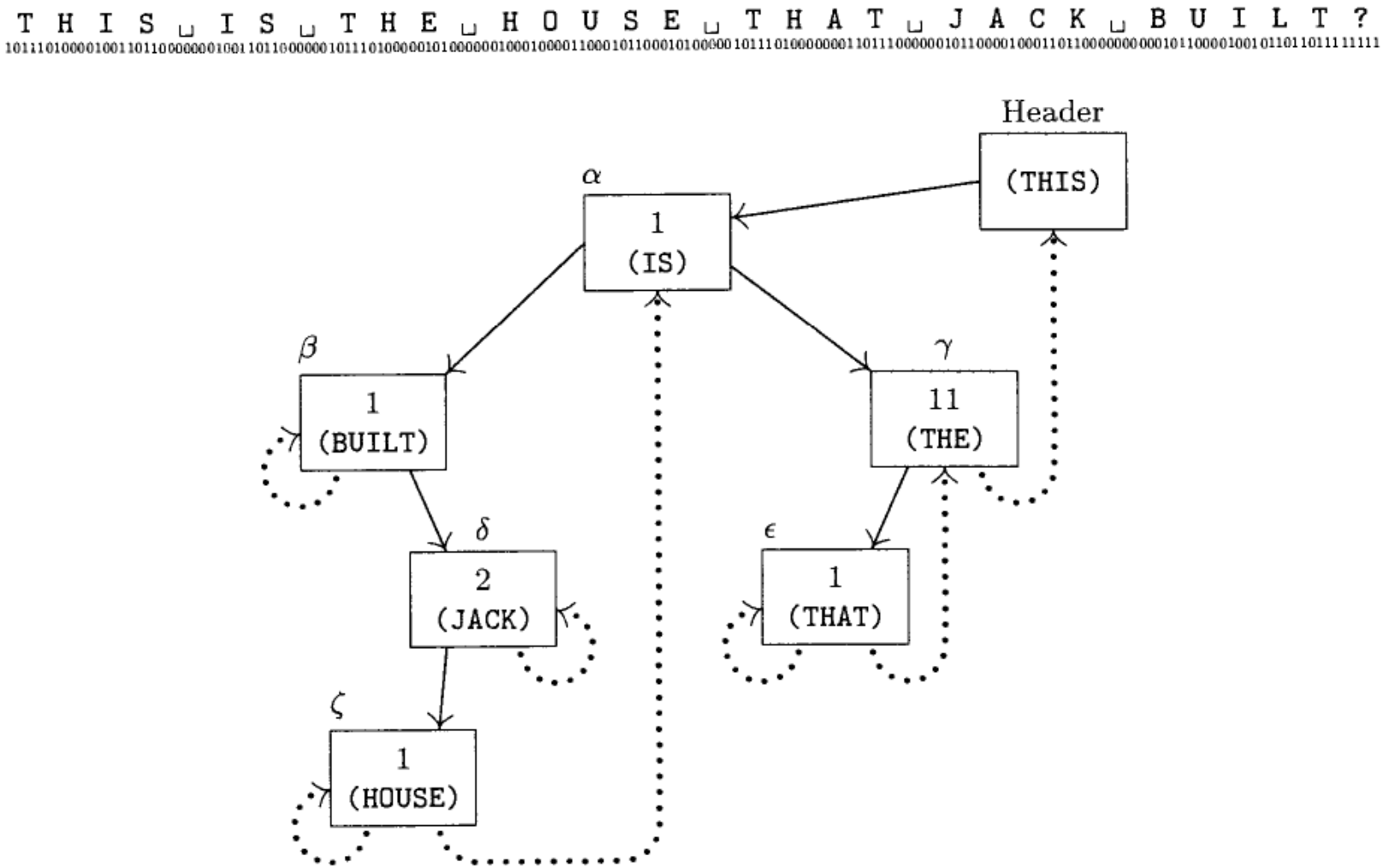
Můžeme prvky duplikovat a data záznamů přesunout až do listů.

Pokud ještě spojíme všechny listy do seznamu, dostáváme $B+$-strom.

B-stromy s klíči proměnlivé délky – stačí udržovat uzel polonaplněný.

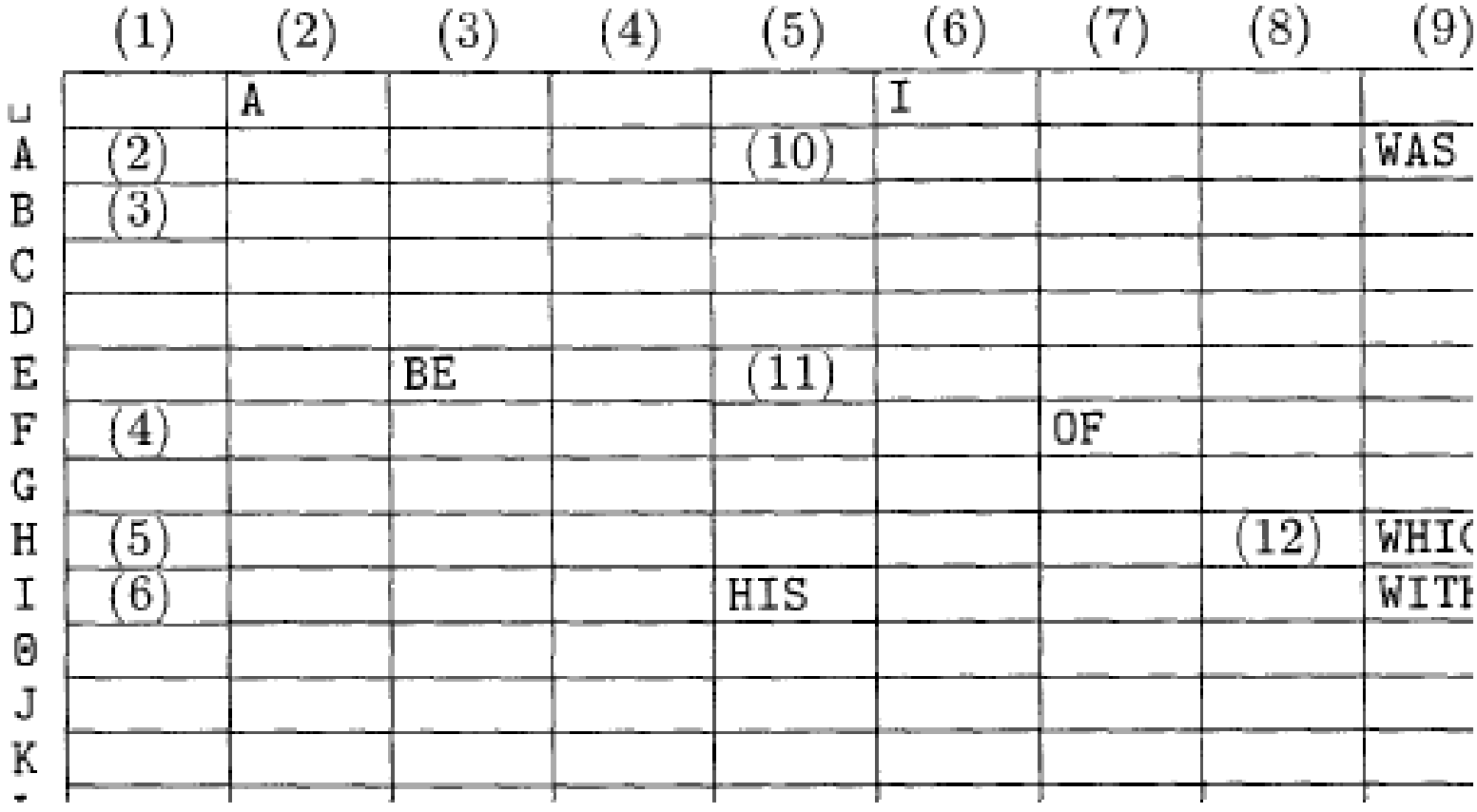
B-stromy s přetečením (overflow) – přeplněné uzly přetékají do sourozenců. Pokud jsou naplněny uzel i sourozenec, splitneme jejich sjednocení do potomků. A navíc musí být větší kořen.

T H I S ␣ I S ␣ T H E ␣ H O U S E ␣ T H A T ␣ J A C K ␣ B U I L T ?



Digitální vyhledávání (digital searching)

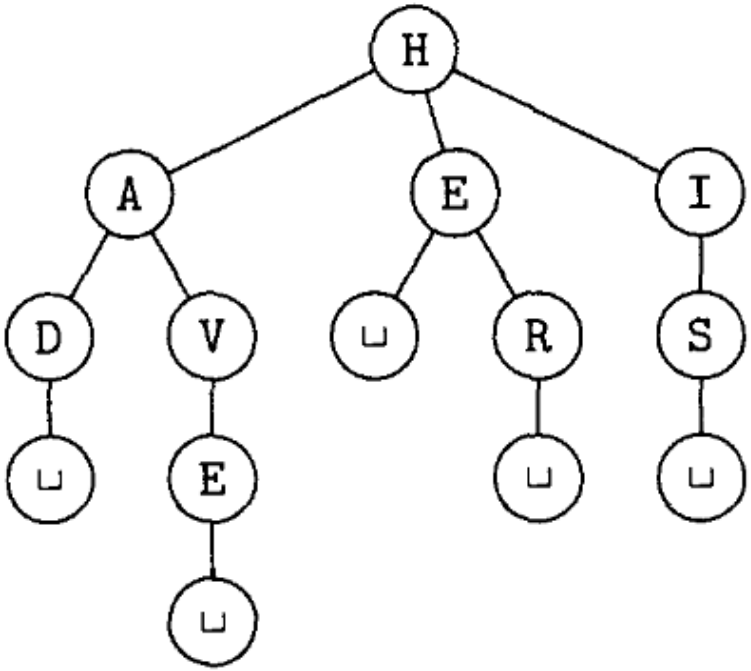
Namísto porovánávání klíčů můžeme využít jejich reprezentace jako posloupnosti čísel nebo znaků abecedy.
Trie je vlasntě M-ární strom, jehož prvky jsou M-místné vektory s komponentami odpovídajícími
Algoritmus vyhledávání v trie – známe z KMI/FJAA

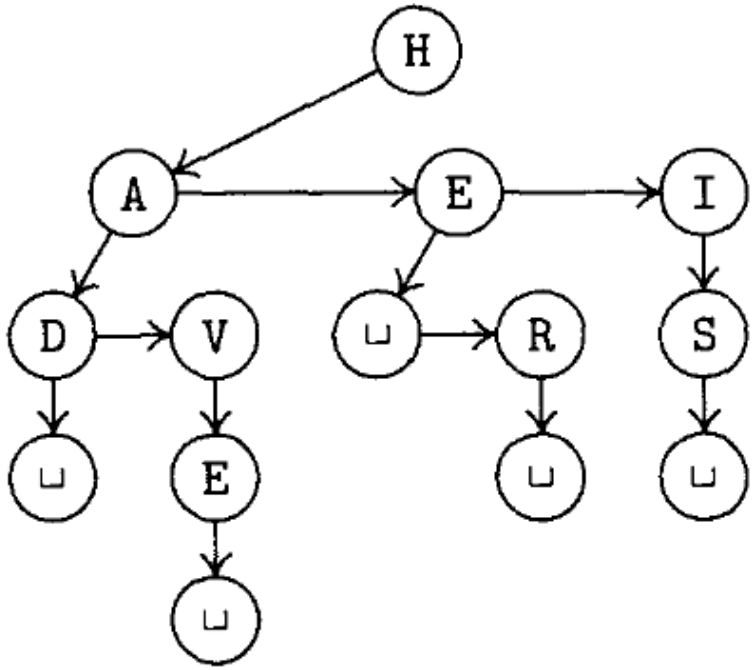

(1) (2) (3) (4) (5) (6) (7) (8) (9)
A
I
A
(2)
(10)
WAS
B
(3)
C
D
E
BE
(11)
F
(4)
OF
G
H
(5)
(12)
WHI
I
(6)
HIS
WIT
J
K

Nějaká pozorovnání ve srovnání s optimálním stromem

- Zabírá to moc paměti
  (tabulka o 360 políčkách reprezentuje 31 klíčů;
  binární strom by zabral 62 „políček“).
- Úspěšné hledání zabere stejně, neúspěšné je v trie kratší.
- V některých případech to tuto výhodu ztrácí
  (rozdíl v posledním písmeni).

Co s tím teda...


H
A
E
I
D
V
␣
R
S
␣
E
␣
␣
␣

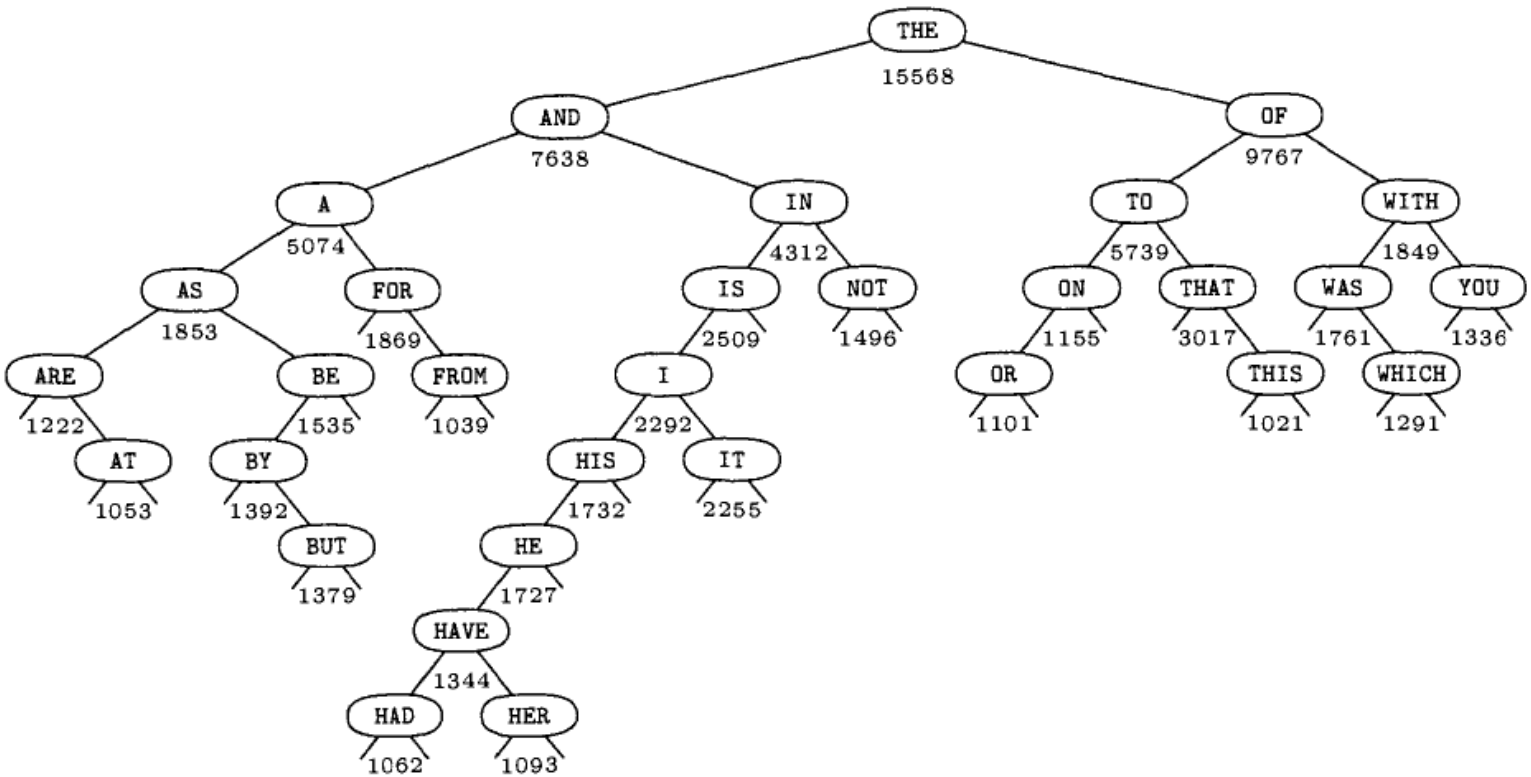

H
A
E
I
D
V
␣
R
S
␣
E
␣
␣
␣

Binární případ

Uvažujme $M = 2$

- *vyhledávání v digitálním stromu*
- *PATRICIA*

Vyhledávání v digitálním stromu



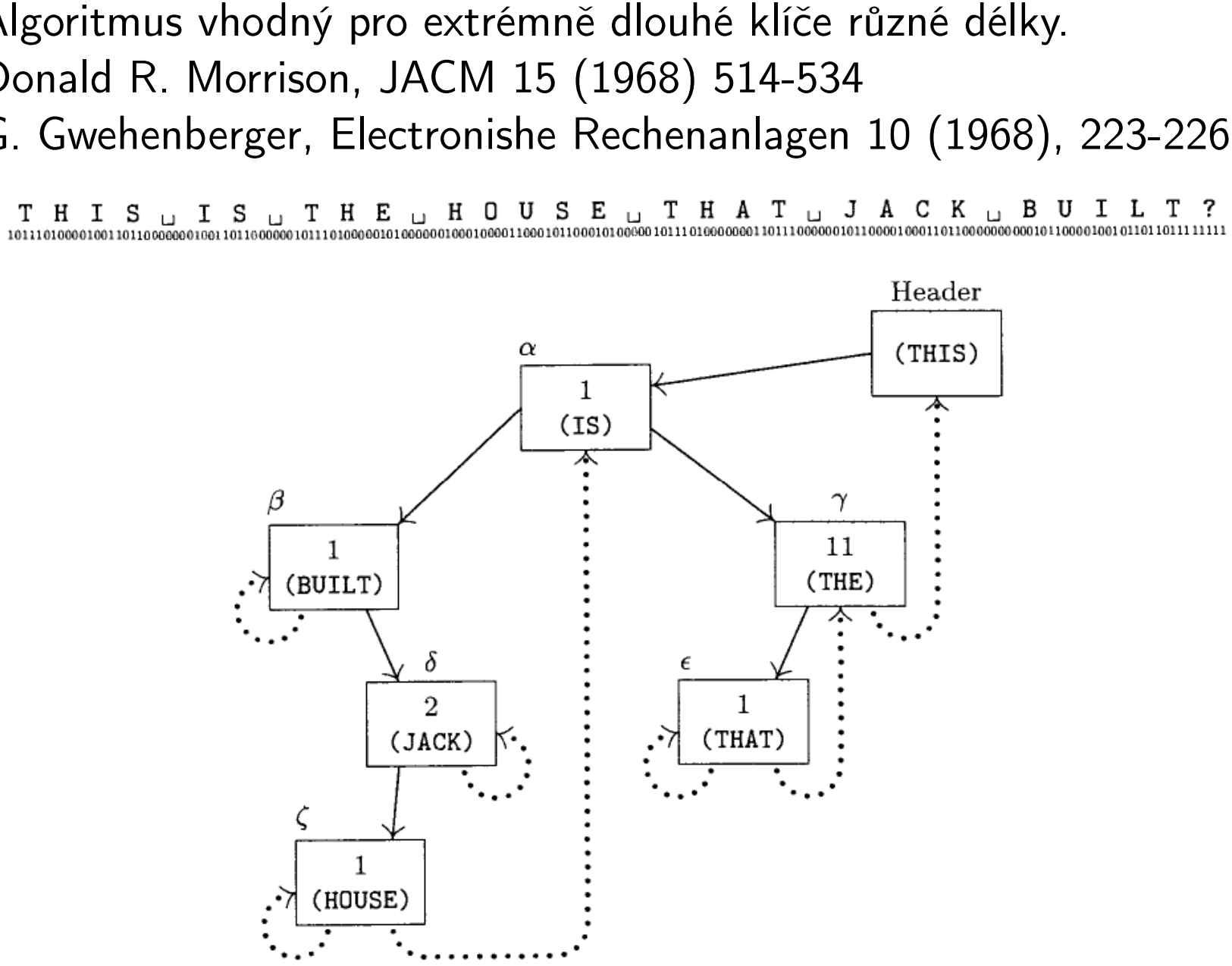
Patricia (Practical Algorithm To Retrieve Information Coded In Alphanumeric)

Algoritmus vhodný pro extrémně dlouhé klíče různé délky.

Donald R. Morrison, JACM 15 (1968) 514-534

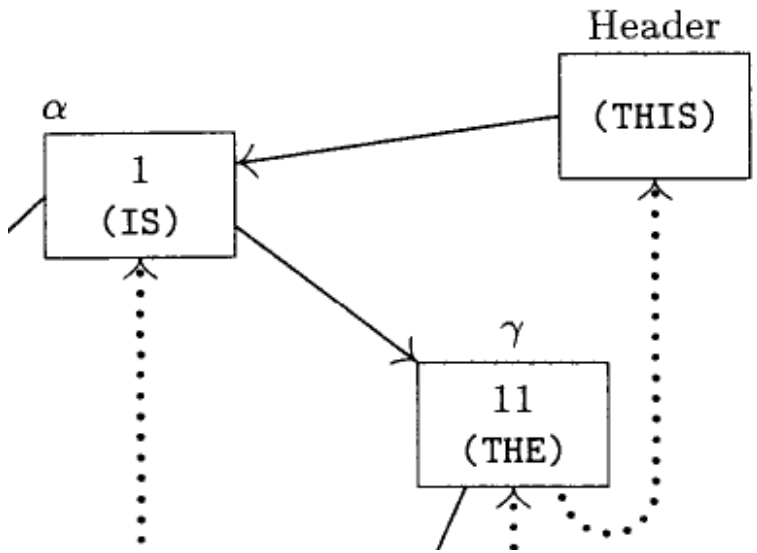
G. Gwehenberger, Electronishe Rechenanlagen 10 (1968), 223-226

T H I S ␣ I S ␣ T H E ␣ H O U S E ␣ T H A T ␣ J A C K ␣ B U I L T ?



Strom, který Patricia používá pro hledání, se skládá z hlavičky a $n - 1$ uzlů, a uzly obsahují tato pole:

- `KEY` – ukazatel do textu.
- `LLINK`, `RLINK` – ukazatele na podstromy
- `LTAG`, `RTAG` – bity, které říkají, jestli `LLINK` a `RLINK` jsou ukazatele na potomky, nebo na předky uzlů.
- `SKIP` – číslo, které říká, kolik bitů se má při hledání přeskočit.



Algoritmus vyhledávání PATRICIA

Vstup: text, strom ve tvaru, jako na předchozím slajdu, klíč K
Výstup: místo v textu, které začíná na K

P1 [Inicializace] $\texttt{P} \leftarrow \texttt{HEAD}$, $j \leftarrow 0$

P2 [Vlevo] $\texttt{Q} \leftarrow \texttt{P}$
$\texttt{P} \leftarrow \texttt{LLINK(Q)}$
Pokud $\texttt{LTAG(Q)}=1$, pokračuj P6

P3 [Přeskoč bity] $j \leftarrow j + \texttt{SKIP(P)}$, pokud $j > n$, pokračuj P6

P4 [Testuj bit] Pokud j-tý bit K je 0, pokračuj P2, jinak pokračuj P5

P5 [Vpravo] $\texttt{Q} \leftarrow \texttt{P}$
$\texttt{P} \leftarrow \texttt{RLINK(Q)}$
Pokud $\texttt{RTAG(Q)}=0$, pokračuj P3

P6 [Porovnání] Porovnej K s klíčem, který začíná na pozici $\texttt{KEY(P)}$.
Pokud jsou stejné, skonči úspěchem, jinak skonči neúspěchem.

Algoritmus vkládání PATRICIA

Vstup: PATRICIA a vkládaný klíč K
Výstup: updatovaná PATRICIA

Pokud je PATRICA prázdná:
$\texttt{LLINK}(HEAD) \leftarrow \texttt{HEAD}, \texttt{LTAG}(HEAD) \leftarrow 1$.
jinak:
Proveď neúspěšné vyhledávání — musí skončit neúspěchem v kroku P6.

Předpokládejme, že K se shoduje s nalezeným klíčem v prvních l bitech a liší se v $l+1$ bitu. K tam má bit b, nalezený klíč tam má bit $1-b$
Zopakuj vyhledávání s tím te se hledá jen prvních l bitů z K (nemusíme provádět P6).

Pak nastavíme

$R \Leftarrow$ alokuj nový uzel

$\texttt{KEY}(R) \leftarrow$ pozice K v textu

Pokud $\texttt{LLINK}(Q) = P$ pak

  $\texttt{LLINK}(Q) \leftarrow R$

  $t \leftarrow \texttt{LTAG}(Q)$

  $\texttt{LTAG}(Q) \leftarrow 0$

jinak

  $\texttt{RLINK}(Q) \leftarrow R$

  $t \leftarrow \texttt{RTAG}(Q)$

  $\texttt{RTAG}(Q) \leftarrow 0.$

Pokud $b = 0$ pak
  $\texttt{LTAG}(R) \leftarrow 1$
  $\texttt{LLINK}(R) \leftarrow R$
  $\texttt{RTAG}(R) \leftarrow t$
  $\texttt{RLINK}(R) \leftarrow P$
jinak
  $\texttt{RTAG}(R) \leftarrow 1$
  $\texttt{RLINK}(R) \leftarrow R$
  $\texttt{LTAG}(R) \leftarrow t$
  $\texttt{LLINK}(R) \leftarrow P$.
Pokud $t = 1$ pak
  $\texttt{SKIP}(R) \leftarrow 1 + l - j$
jinak
  $\texttt{SKIP}(R) \leftarrow 1 + l - j + \texttt{SKIP}(P)$
  $\texttt{SKIP}(P) \leftarrow j - l - 1$.