Aplikovaná informatika

Podklady předmětu
Aplikovaná informatika
pro akademický rok 2013/2014
Radim Farana

9

Obsah

- Dynamické datové struktury,
- Optimální stromy,
- B-stromy.
- Realizace indexových souborů.

Optimální stromy

- Pokud se prvky vyskytují s různou pravděpodobností.
- Pravděpodobnost chápeme jako váhu prvku.
- Konstruujeme stromy s minimální váženou délkou cesty: $P_l = \sum_{i=1}^{n} p_i h_i \longrightarrow \min$

kde je h_i - úroveň i-tého prvku stromu,
p_i - pravděpodobnost přístupu k *i*-tému prvku,
P_l - vážená délka cesty stromu.

Optimální stromy

- Teorie pravděpodobnosti
- Každému jevu A ⊂ E (množina všech přípustných jevů, jistý jev) je přiřazeno jako pravděpodobnost číslo $P(A)$, přičemž platí následující axiomy:
  - pravděpodobnost je nezáporná, tj. $P(A) \geq 0$;
  - pravděpodobnost sjednocení konečně mnoha nebo spočetně mnoha vzájemně neslučitelných jevů $A_1 \subset E$, $A_2 \subset E$, ... je rovna součtu pravděpodobností těchto jevů, tj. $P(A_1 \cup A_2 \cup ...) = P(A_1) + P(A_2) + ...$;
  - pravděpodobnost jistého jevu E je rovna 1, tj. $P(E) = 1$.

Optimální stromy

- Pravděpodobnost výskytu prvku určujeme často jako poměr počtu výskytů tohoto prvku ku počtu všech prvků v souboru prvků: $p_i = \frac{m_i}{\sum_{i=1}^{n} m_i}$

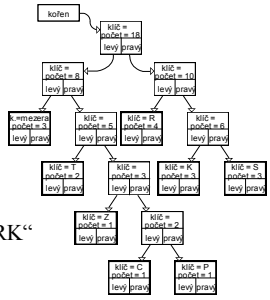
  kde je m_i - počet výskytů i-tého prvku,
  n - počet různých prvků v souboru.
- Beze ztráty obecnosti můžeme pracovat přímo s počtem výskytů prvku, pokud platí: $m_i \; 0$ pro $i = 1, 2, ... n$
- Vážená délka cesty stromu je pak: $P_I = \sum_{i=1}^{n} m_i h_i$

Optimální stromy

- Strom, u kterého dosahuje vážená délka cesty minimální hodnoty nazýváme optimální strom.
- Složitost algoritmu je řádově $O(n^2)$ a velikost potřebné paměti rovněž $O(n^2)$.
- Pokud se spokojíme s kvazioptimálním stromem, můžeme využít velmi efektivní algoritmus, který má složitost $O(n.\log n)$ a paměťové nároky úměrné $O(n)$.

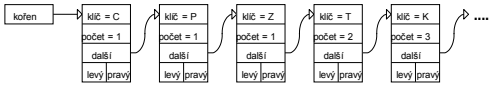
Optimální stromy

- Příklad optimálního stromu – strom Huffmanova kódu.
  - informaci ale nesou jen koncové prvky (prefixový kód).
  - Příklad pro text: „STRC PRST SKR KRK“



Strom Huffmanova kódu

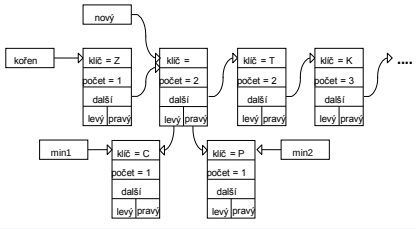
- Určíme, které znaky se v souboru vyskytují a kolikrát, data uložíme do lineárního seznamu.
- Prvky seřadíme podle počtu výskytů sestupně.



  - Ukazatele levý a pravý je nutno nastavit na *nil*.

Strom Huffmanova kódu

- Mezi první dva prvky vložíme nový prvek se sumou jejich výskytů a zařadíme ho do seznamu na správné místo.



Strom Huffmanova kódu

- Postup opakujeme až do okamžiku, kdy bude výsledný seznam obsahovat jediný prvek – kořen stromu Huffmanova kódu.

```
if kořen <> nil then
  while kořen↑.další <> nil
    min1 = kořen
    kořen = kořen↑.další
    min2 = kořen
    kořen = kořen↑.další
    new (nový)
    nový↑.levý = min1
    nový↑.pravý = min2
    nový↑.počet = min1↑.počet + min2↑.počet
    zařazení prvku nový do seznamu
  end while
end if
```


B-stromy

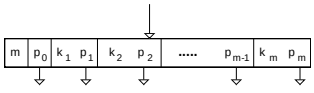
- Jsou speciálním případem stromů vyššího stupně (každý prvek může mít více následníků).
- Navrhl je R. Bayer v roce 1970
- Strom je složen ze stránek.
- Každá stránka musí obsahovat n až $2n$ prvků pro danou konstantu n (kromě kořenové).
- Strom s N prvky a maximální velikostí stránky $2n$ vyžadovat maximálně $\log_n N$ přístupů ke stránce pro vyhledání prvku.
- Paměť je využita nejméně na 50 %.

B-strom

- Parametr n označuje řád stromu.
- Pravidla pro vytvoření B-stromu:
  - Každá stránka obsahuje nejvýše $2n$ prvků (klíčů).
  - Každá stránka kromě kořenové obsahuje alespoň n prvků.
  - Každá stránka je buď koncová (nemá následníky), nebo má $m + 1$ následníků, kde m je počet prvků stránky.
  - Všechny koncové stránky jsou na stejné úrovni.

B-strom

- Datová struktura stránky



- Vyhledání prvku pomocí sekvenčního nebo binárního prohledávání.

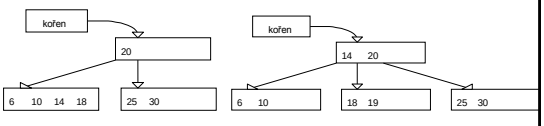
B-strom

Objevitelé: Rudolf Bayer & Ed McCreight:
B jako *„ballanced“*, ale možná jako *„Bayer“* nebo *„Boeing“*, neboť pracovali pro *„Boeing Scientific Research Labs“*

- Pokud prvek neleží ve stránce, mohou nastat případy:
  - *klíč* < k_1 – vyhledávání pokračuje ve stránce $p0\uparrow$,
  - k_i < *klíč* < k_{i+1} – vyhledávání pokračuje ve stránce $p_i\uparrow$,
  - k_m < *klíč* – vyhledávání pokračuje ve stránce $p_m\uparrow$.
- Pokud má ukazatel hodnotu nil, je nutno prvek přidat, mohou nastat případy:
  - $m < 2n$ – datová struktura není plná, prvek se přidá do stránky.
  - $m = 2n$ – stránka je plná, rozdělí se na dvě stránky po n prvcích (jedna obsahuje prvních n prvků, druhá posledních n prvků). Zbývající prvek (ze středu posloupnosti prvků) je přesunut do vyšší stránky. Pokud dojde k přeplnění vyšší stránky, opět dojde k jejímu rozdělení.

B-strom

- B-strom roste od koncových prvků ke kořeni.
- Nová úroveň vznikne přeplněním a rozdělením kořenové stránky.



Vložení prvku s klíčem 19 do B-stromu

B-strom - realizace

- Datová struktura
  - stránka : structure
    m - počet prvků ve stránce
    p0 - ukazatel na prvního následníka
    e [1 až 2n] - prvky stránky
    end structure
  - prvek : structure
    klíč - položka klíče
    data - ostatní data spojená s prvkem
    p - ukazatel na další stránku
    end structure

B-strom - realizace

```
hledej (klíč, a, ByRef h,ByRef u)
if a = nil then
  REM klíč není v B-stromu, prvku u přiřadíme nový klíč
  u.klíč = klíč nového prvku
  u.data = data nového prvku
  h = true
  REM prvek u bude putovat nahoru v B-stromu
else
  REM hledáme prvek s udaným klíčem ve stránce a pomocí binárního prohledávání
  if prvek nalezen then
    REM manipulace s daty nalezeného prvku a↑.e[i]
  else
    hledej (klíč, následník, h, u)
    if h then
      REM prvek postupuje nahoru
      if a↑.m < 2n then
        REM prvek u přidáme do stránky a↑
        h = false
      else
        REM stránku a↑ rozdělíme na dvě stránky
        REM střední prvek dosadíme do u
        h = true
      end if
    end if
  end if
end if
end hledej
```

Parametry:
klíč – hledaný klíč,
a – aktuální stránka,
h – informuje o rozdělení stránky,
u – pro předání prvku,
který je přesouván do vyšší stránky.

B-strom - realizace

- Kořenová stránka musí být obsloužena speciálně:

```
vkládání (klíč)
  hledej (klíč, kořen, h, u)
  if h then
    q = kořen
    new (kořen)
    kořen↑.m = 1
    kořen↑.p0 = q
    kořen↑.e[1] = u
    REM prvek u obsahuje informaci o klíči a ukazatel
    REM na stránku následníků
  end if
end vkládání
```

B-strom, průchod stromem

- Obvykle je požadován průchod v pořadí klíčových hodnot:

```
průchod (p)
  if p <> nil then
    průchod (p↑.p0)
    for i = 1 to p↑.m step 1
      REM zpracování dat p↑.e[i].data
      průchod (p↑.e[i].p)
    end for
  end if
end průchod
```


B-strom, rušení stromu

- Obvykle se strom ruší najednou:

```
zrušení (p)
  if p <> nil then
    zrušení (p↑.p0)
    for i = 1 to p↑.m step 1
      zrušení (p↑.e[i].p)
      REM zrušení dat p↑.e[i].data
    end for
    dispose (p)
  end if
end zrušení
```