VYHLEDÁVÁNÍ V TABULKÁCH S ROZPTÝLENÝMI POLOŽKAMI

Doc. Ing. Jan Honzík, CSc. - katedra počítačů FE VUT v Brně

Příspěvek o vyhledávání v tabulkách s rozptýlenými položkami (Hash-tables, Scattered tables) navazuje na serii příspěvků o vyhledávání, uvedených ve sbornících předcházejících seminářů PROGRAMOVÁNÍ. Téma referátu vychází z části učebních osnov předmětu Programovací techniky pro 3. ročník studia oboru Elektronické počítače na Elektrotechnických fakultách.

Vyhledávání v tabulkách s rozptýlenými položkami představuje jednu z nejrychlejších metod vyhledávání. Tato metoda má velmi dobré dynamické vlastnosti v souvislosti s vkládáním nových položek do tabulky; méně příznivé dynamické vlastnosti má v souvislosti s rušením položky v tabulce. Tyto tabulky se často používají v překladačích i v jiných systémových programech. Příklady v příspěvku jsou zapsány v programovacím jazyce Pascal.

1. Tabulky s přímým přístupem

Nechť je dána množina všech klíčů $K = K_1, \ldots, K_n$, které se budou vkládat do tabulky a nechť je nad typem klíče definována relace rovnosti. Je-li možné nalézt jedno-jednoznačnou mapovací funkci $f(K_i) = i$ $(i=1,2,\ldots,n)$ pro všechny prvky množiny K, pak lze vytvořit tzv. tabulku s přímým přístupem. Tuto tabulku tvoří pole, v němž položka s klíčem K_j bude uložena na indexu j daného pole. U každé položky lze zjistit, zda je obsazená či volná (např. pomocnou Booleovskou složkou). Mechanismus operací INITTAB, INSERT, DELETE a SEARCH je triviální, a nemá smysl ho zde uvádět.

Obtíž s využitím této jinak vysoce účinné tabulky spočívá v obtíži nalézt vhodnou mapovací funkci. Přesto se v praxi podobné tabulky často používají a za tím účelem se používají numerické klíče, které známe pod názvy "pořadové" nebo "evidenční číslo".

Tam, kde však musíme pracovat s jiným typem klíče - např. s textovým klíčem - není možné tento mechanismus použít beze zbytku.

2. Vhodná mapovací funkce

Podle Knutha /1/ existuje pro 31 různých prvků, které se mají zobrazit do 41 prvkové množiny 41^{31} (10^{50}) možných funkcí. Přitom pouze (41!/31!) z nich je jedno-jednoznačných. To znamená, že poměr vhodných funkcí ku všem možným je v tomto případě $1:10^7$. Jedno-jednoznačné funkce jsou tedy velmi řídkým jevem a s jejich nalezením pro obecně zadanou množinu klíčů nelze počítat.

V dalších úvahách budeme hledat takovou mapovací tabulku, která klíče z dané množiny "rozptýlí" v dané tabulce, aniž budeme trvat na jedno-jednoznačnosti. Klíče, které mají stejnou hodnotu mapovací funkce nazveme <u>synonyma</u>. Pokusu o umístění nové položky na místo již obsazené budeme říkat <u>kolize</u>. Mapovací funkci použitou k rozptýlení nazveme <u>rozptylovací funkce</u>.

Předpokládejem existenci celočíselné funkce NUM(K), která transformuje klíč na hodnotu celého kladného čísla. Tato funkce významně ovlivní množství synonym vzniklých nad danou množinou klíčů. Nelze proto udělat obecné závěry o její volbě bez znalosti vlastností množiny klíčů.

Mějme rozptylovací funkci R(K)=H(NUM(K), kde funkce H zajistí transformaci přirozeného čísla s intervalem (0..MAXINT) do intervalu pole, jímž implementujeme tabulky (nejčastěji Ø.. MAX resp. 1..MAX). Pro funkci H se nejčastěji využívá vlastností operace modulo.

H(i) = i <u>mod</u> (MAX+1) pro interval Ø..MAX

H(i) = i <u>mod</u> MAX+1 pro interval 1..MAX

na dobrou rozptylovací funkci se kladou dva základní požadavky:

a) výpočet rozptylovací funkce musí být dostatečně rychlý
b) rozptylovací funkce vytváří co nejméně kolizí

Je zřejmé, že v řadě případů jsou tyto požadavky protichůdné.

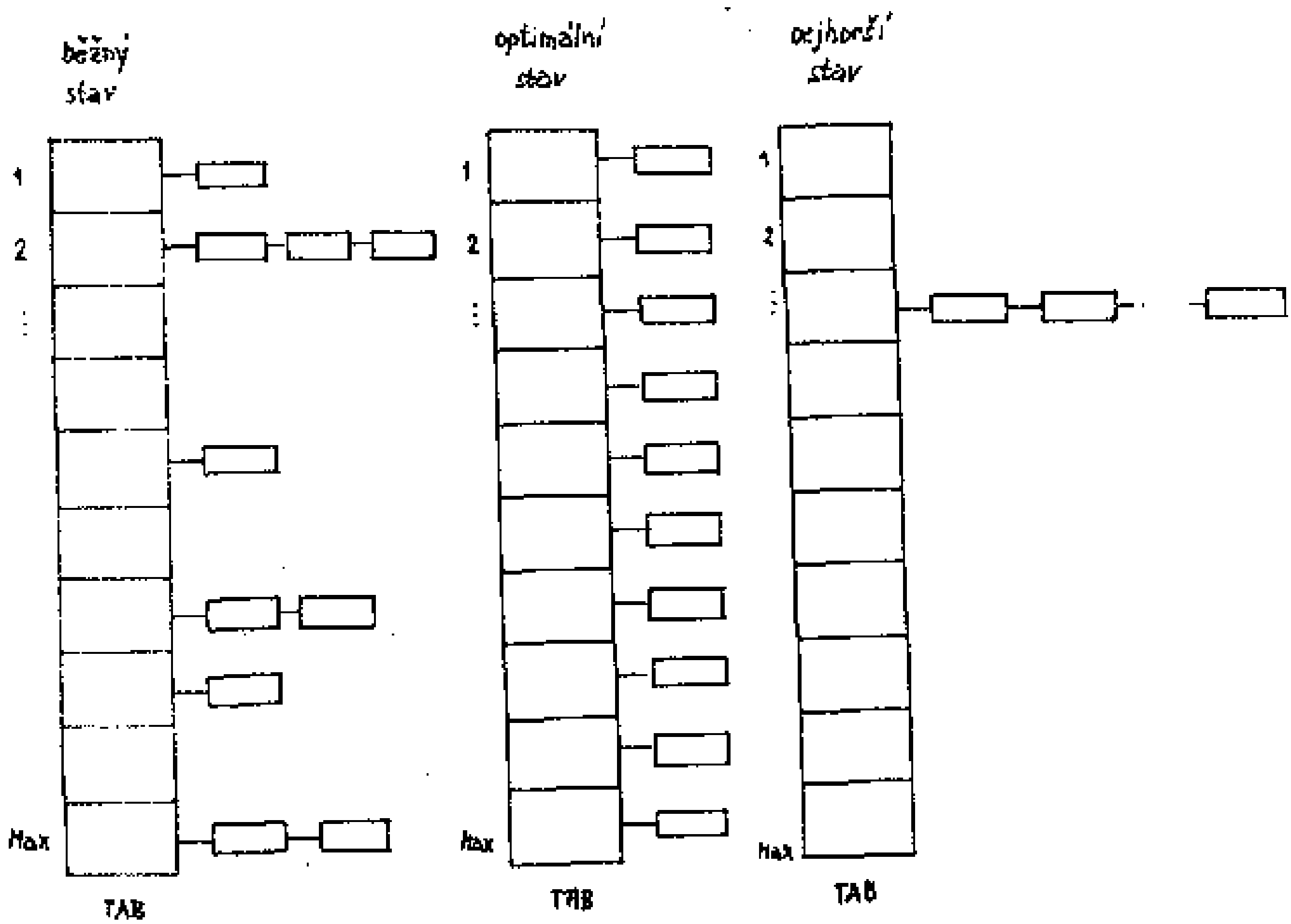
3. Princip tabulek s rozptýlenými položkami

Princip tabulek s rozptýlenými položkami je podobný práci s indexsekvenčním souborem. Rozptylovací funkce určí index rozptylovacího pole, v němž jsou uloženy začátky seznamů synonymních položek tabulky. Vyhledání bude tedy sestávat ze dvou fází:

a) určení začátku seznamu synonym

b) sekvenční vyhledávání průchodem seznamu

Z toho vyplývá, že délka vyhledávání záleží na délce seznamu synonym, kterým se prochází. Nejhorší případ je dán nejdelším seznamem synonym. Princip tabulek s rozptýlenými položkami je znázorněn na obr. 3.1.



Obr. 3.1 Princip tabulek s rozptýlenými položkami

Na obr. 3.1 znázorněn "běžný stav", v němž je nejhorší případ pro hledání prvku, který je v seznamu o třech prvcích. Nejhorší případ, v němž nevhodná rozptylovací funkce vytvořila ze všech klíčů synonyma, degraduje tabulku s rozptýlenými položkami na sekvenční vyhledávání v seznamu

Podle způsobu, jakým se realizuje seznam synonym, lze tabulku s rozptýlenými položkami rozděli do dvou skupin.

a) tybulky s explicitně zřetězenými synonymy

b) tabulky s implicitně zřetězenými synonymy

Explicitním zřetězením se rozumí zřetězení pomocí ukazatele. Implicitním zřetězením se rozumí, že z adresy (indexu) každého prvku seznamu lze určit adresu (index) následníka v seznamu

4. Tabulky s rozptýlenými položkami s explicitně zřetězenými synonymy

Základem tabulky je pole. Tabulky z této skupiny můžeme rozdělit podle toho, jakým způsobem je organizována paměť pro položky seznamu.

a) Pole obsahuje na každém indexu místo pro 1. položku seznamu a pro ukazatel na další synonymum v seznamu synonym, který je tvořen prvky "oblasti přeplnění".

b) Pole obsahuje pouze ukazatele. Všechny prvky jsou uloženy vždy v seznamu, jak to znázorňuje obr. 3.1. Tento přístup je zejména z hlediska vkládání a rušení prvku podstatně jednodušší.

Také oblast přeplnění může být organizována různě:

A) Paměťový prostor je získáván mechanismem dynamického přidělování paměti (v Pascalu "new", obdobně v uživatelově systému dyn. přidělování paměti).

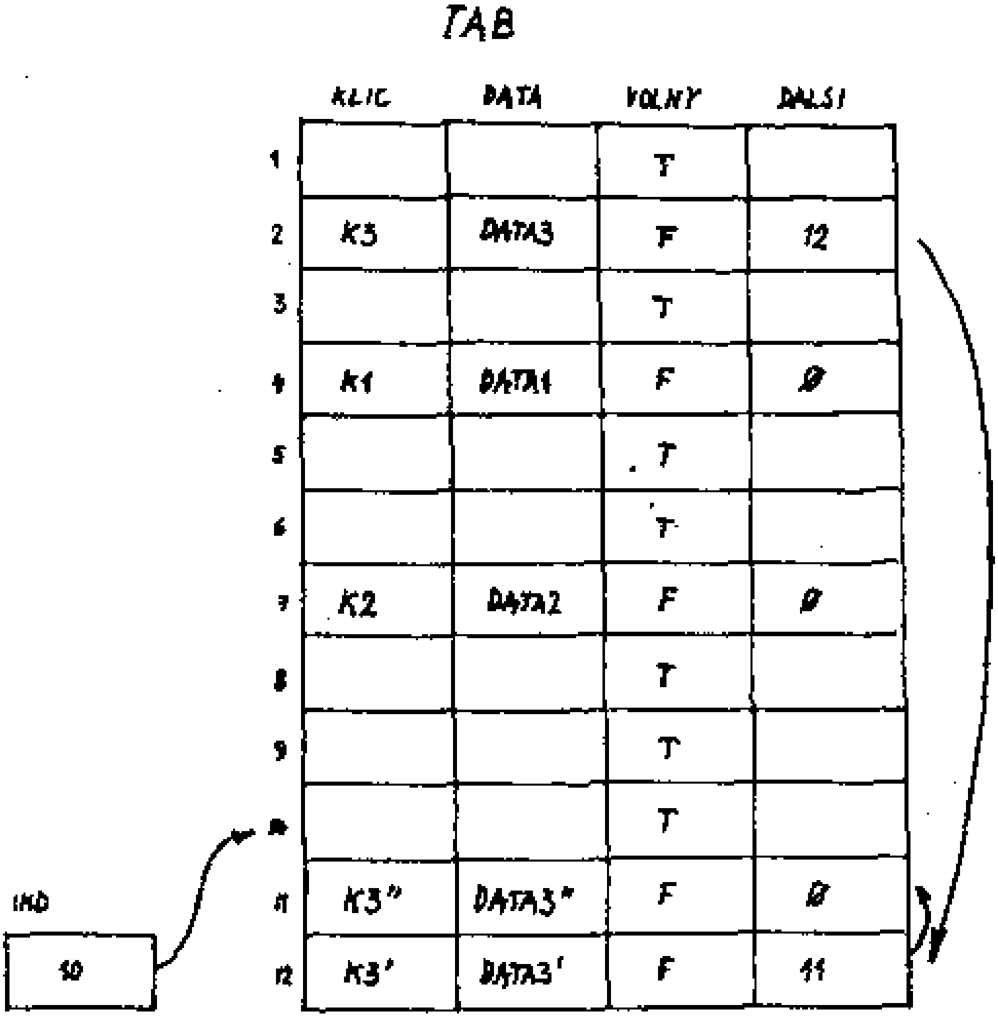
B) Oblast přeplnění tvoří zvláštní pole (může to být např. "horní" část pole, jehož "dolní" část je použita jako rozptylovací pole).

C) Oblast přeplnění a rozptylovací pole se vzájemně překrývají.

Ze všech případů se zdá být nejúčinnější a také nejjednodušší případ (b,A). Po vyhledání začátku seznamu synonym (I:=H(K)) se dále vyhledává, vkládá i ruší prvek stejně jako v lineárním zřetězeném seznamu. Je-li nad typem klíče definována relace uspořádání, je možné uspořádat prvky v seznamu podle velikosti a zkrátit tak délku neúspěšného vyhledávání.

Protože je tento příklad jednoduchý, nebudeme jeho algoritmus podrobněji rozvádět. Je uveden v /2/ na str. 194.

V /1/ je uveden algoritmus, v němž se oblast přeplnění překrývá s rozptylovacím polem. Princip je znázorněn na obr.4.1.



Obr. 4.1. Princip "Knuthovy" metody

Klíče K3, K3' a K3'' tvoří seznam synonym. Volné místo pro nový prvek seznamu synonym se hledá pomocným systémem "dynamického přidělování paměti", využívajícího pomocnou proměnnou IND.

Nechť jsou definovány typy:

```
TYPPOLOZKY  = record
                  KLIC : TYPKLIC;
                  DATA : TYPDATA;
                  VOLNY: Boolean;
                  DALSI: Ø..MAX
              end;
TYPTAB = array [1..MAX] of TYPPOLOZKY
```

Operace inicializace pak ustaví pole volných prvků a nastaví pomocný ukazatel IND.

```
procedure INIT(var TAB:TYPTAB;var IND:POSINT);   {POSINT=Ø..MAX}
var I:POSINT;
begin for I:=1to MAX do TAB [I] .VOLNY:=true;
      IND:=MAX
end;  {procedury INIT}
```

Operace INSERT v sobě obsahuje i mechanismus vyhledávání. Podle uvedeného Knuthova algoritmu má tvar:

```
procedure INSERT (var TAB:TYPTAB;K:TYPKLIC;UDAJ:TYPDATA;
                  var IND:POSINT;var ERROR:Boolean);
var I:POSINT;
    JESTE,NASEL:Boolean;
    begin I:=R(K);  {Rozptylovaci funkce dává hodnoty 1..MAX}
        ERROR:=false;NASEL:=false;  {Inicializace Bool.proměnných}
        if not TAB [I] .VOLNY
           then {prvek v poli není volný, hledej v seznamu}
              begin JESTE:=TAB [I] .KLIC≠K {Nastavení proměnné cyklu}
                    while JESTE do
                     begin I:=TAB [I] .DALSI; {index následníka}
                           if I=Ø then JESTE:=false {konec seznamu}
                                  else JESTE:=TAB [I] .KLIC≠K
                     end;
                 if I≠Ø then begin
                               NASEL:=true;
                               TAB [I] .DATA:=UDAJ {přepis při nale-
                                                              zení}
                             end;
                 if not NASEL
                  then {Najde místo pro novou položku}
                  begin while (IND≠Ø)and(not TAB [IND] .VOLNY)do
                              IND:=IND-1;
```

```
                    ERROR:=IND=∅;
                    if not ERROR
                       then begin TAB [I] .DALSI:=IND;
                                                  {připojení}
                                       I:=IND {příprava pro vlože-
                                                              ní}
                            end
               end
          end; {pro then za if not TAB [I] .VOLNY}
      if (not NASEL)and(not ERROR)then {vložení nové složky}
        begin TAB [I] .KLIC:=K;
              TAB [I] .DALSI:=∅;
              TAB [I] .DATA :=UDAJ
              TAB [I] .VOLNY:=false
        end
end; {procedury INSERT}
```

Nevýhodou této metody je skutečnost, že v tabulce nelze zrušit prvek stejným způsobem, jako ve zřetězeném seznamu. Důvodem je překrývání rozptylovacího pole a oblasti přeplnění.

5. Tabulky s rozptýlenými položkami a implicitně zřetězenými synonymy

Tyto tabulky jsou implementovány jedním polem, v němž se překrývá rozptylovací pole s oblastí přeplnění. Index následníka v seznamu synonym je dán součtem indexu předchůdce a přírůstku INC. Podle druhu přírůstku můžeme TRP s implicitně zřetězenými synonymy rozdělit na:

a) lineární vyhledávání (INC je konstantní; např. INC:=1)
b) kvadratické vyhledávání (INC lineárně roste, např.INC:=INC+1)
c) metoda dvou rozptylovacích funkcí (INC je konstantní, ale získává se druhou rozptylovací funkcí, INC:=R_2(K))

Ve všech příkladech se s polem pracuje jako s kruhovým seznamem. Je-li index následníka větší než MAX, redukuje se o tuto hodnotu.Významným je způsob, kterým se určuje poslední prvek seznamu synonym.

Pro účinnost TRP s implicitně zřetězenými synonymy je významný tzv. koeficient zaplnění tabulky N/MAX. Účinnost některých metod se výrazně snižuje, přibližuje-li se koeficient zaplnění hodnotě 1.

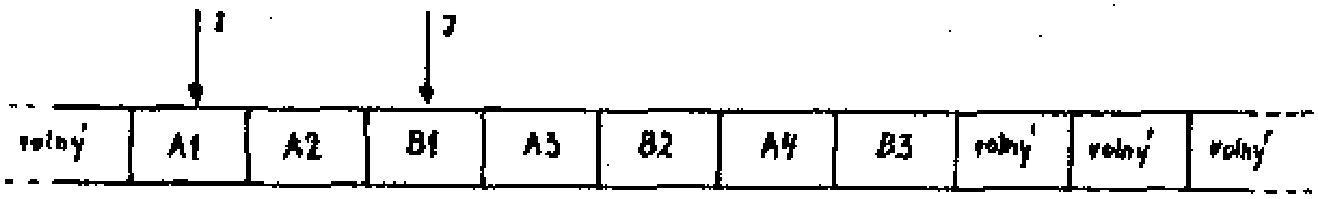
5.1. Tabulky s rozptýlenými položkami s lineárním vyhledáváním

Vyhledávání v této tabulce postupuje podle těchto pravidel:

a) Rozptylovací funkce určí index prvního přístupu do pole tabulky

b) Od této pozice se zahájí vyhledávání v sekvenčně umístěném seznamu v poli. Vyhledávání končí úspěšně při nalezení položky s vyhledávaným klíčem nebo neúspěšně, dojde-li se k prvnímu neobsazenému prvku pole. S polem se pracuje jako s kruhovým seznamem.

Pro zajištění konečnosti neúspěšného vyhledávání se musí zachovat alespoň jeden neobsazený prvek pole, který slouží jako zarážka. Nejčastěji se to zajišťuje pomocnou proměnnou, udávající počet prvků v tabulce. Na obr. 5.1. je uveden příklad, který ukazuje, že prohledávaný seznam nemusí být složen vždy jen ze synonym. Nechť klíče A1, A2, A3 a A4 jsou synonyma se společným přístupovým indexem I=R(A1)=R(A2)=R(A3)=R(A4) a klíče B1,B2 a B3 jsou také synonyma s přístupovým indexem J=R(B1)=R(B2)=R(B3). Budou-li se do tabulky vkládat klíče v pořadí A1,B1,A2,A3,B2,A4,B3, bude mít tabulka tvar podle obr. 5.1.



Obr. 5.1. TRP s lineárním vyhledáváním pro INC=1

Příklad na obr.5.1 znázorňuje situaci, která je v TRP s lineárním vyhledáváním častá. Projevuje se vytvářením shluků obsazených prvků. Situaci lze zlepšit zvýšením hodnoty přírůstku INC. Tato hodnota však nesmí být dělitelem délky základního pole, aby se zajistil průchod všemi prvky pole.

5.2. Tabulky s rozptýlenými položkami s kvadratickým vyhledáváním

Jiný významný způsob zabránění shluků spočívá v kvadratickém vyhledávání, t.zn., že hodnoty indexu po sobě jdoucích prvků vytvářejí kvadratickou funkci. V [3] je popsána a odvozena metoda, která postupně prochází indexy

$$I_0 = R(K)$$

$$I_j = (I_0 + 0{,}5j + 0{,}5j^2) \ \underline{mod}\ MAX$$

přičemž platí

$$I_{j+1} = I_j + INC_j$$

$$INC_{j+1} = INC_j + 1$$

Je-li $I_0 = 1$ a $INC_0 = 1$, pak se postupně projde indexy 1,2,4,7,11, 16,22,.. Pro tuto metodu musí mít MAX hodnotu prvočísla ve tvaru 4n+3 (např. 991)

Nechť jsou dány typy

```
TYPPOLOZKY = record
                 KLIC : TYPKLIC;
                 DATA : TYPDATA;
            OBSAZENY : Boolean;
             end;
TYPTAB = array [1..PRVOCISLO] of TYPPOLOZKY;
```

Pak operace INITTAB bude mít tvar:

```
procedure INITTAB (var TAB:TAPTAB);
var I: integer;
begin for I:=1 to PRVOCISLO do TAB [I] .OBSAZENY:= false
end;
```

Operace vkládání má tvar:

```
procedure INSERT (var:TYPTAB;K:TYPKLIC;UDAJ:TYPDATA;var ERROR:
                                                      Boolean);
var I, INC: integer;
    KONEC,VLOZIL : Boolean;
begin ERROR:=false;
```

```
INC:=1;    {* INC:=-PRVOCISLO}
I:=R(K); {I je z intervalu 1..PRVOCISLO}
VLOZIL := false;

repeat KONEC:=false;
        if not TAB [I] .OBSAZENY
           then          {Našel volný prvek, vkládá a končí}
                 with TAB [I] do
                 begin KLIC:=K;
                       DATA:=UDAJ;
                       OBSAZENY:=true;
                       KONEC:=true;VLOZIL:=true
                 end
           else {prvek není volný}
              begin if TAB [I] .KLIC=K
                       then {Našel shodu, přepisuje}
                          begin TAB [I] .DATA:=UDAJ;
                                KONEC:=true
                          end
                       else {připrav index dalšího prvku}
                          begin
                            I:=I+INC; {*I:=I+abs(INC)}
                           if I > PRVOCISLO then I:=I-PRVOCI
                                                       SLO;
                           INC:=INC+1;
{* if INC ≥ PRVOCISLO}     if INC ≥ (PRVOCISLO div 2)
                               then begin KONEC:=true
                                          ERROR:=true
                                    end
                           end
              end
  until KONEC
end ; {procedury INSERT}
```

Tato metoda připouští, aby seznam synonymních klíčů nebyl větší, než je polovina tabulky, což v praktických aplikacích nepříznivě snižuje účinnost. Tento nedostatek odstraňuje metoda napsaná v [3] "Full Table Quadratic Searching" jednoduchými úpravami uvedenými v komentářových závorkách a označených * .

5.3. Tabulky s rozptýlenými položkami se dvěma rozptylovacími funkcemi

Shluky v tabulkách lze odstranit také pomocí přírůstku INC, jehož konstantní hodnota pro daný klíč se určí druhou rozptylovací funkcí. První rozptylovací funkce R(K) dává hodnotu z intervalu Ø..MAX, druhá Q(K) z intervalu 1..MAX takovou, která není dělitelem čísla MAX+1. Bude-li MAX+1 prvočíslo, pak je to každé číslo z intervalu 1..MAX.

Konec neúspěšného prohledávání seznamu synonym nastane tehdy dojde-li se k prázdné položce. Protože algoritmus je principiálně podobný předcházejícím algoritmům, nemá smysl uvádět jeho podobný zápis.

5.4. Brentova varianta

Brentova varianta je variantou metody se dvěma rozptylovacími funkcemi. Má význam za předpokladu, že úspěšné vyhledávání je častější než neúspěšné vyhledávání s následným vložením. Při vkládání položky se snaží najít vhodnější místo, než je první volná položka. Cena za náročnější vkládání tohoto místa se vrátí při rychlejším vyhledávání.

Metoda pracuje na tomto principu:

a) Po průchodu S prvky seznamu synonym (s přírůstkem Q(K)) jsme nalezli volné místo. Je to místo, na které původní metoda po neúspěšném vyhledání vloží novou položku.

b) Od každého prvku seznamu P_i (i=0,1,...,S-1) seznamu synonym začnu vyhledávat nejbližší volné místo, při průchodu s přírůstkem odvozeným funkcí Q z klíče uloženého ve zkoumaném prvku P_i. Najde-li takové místo po počtu kroků K, pro nějž platí K+i < S, provedu následující přesun:

   Na právě nalezené místo přesunu prvek P_i

   Na místo prvku P_i vložím nově vkládaný prvek.

c) Nenalezne-li se pro žádný prvek P_i volné místo po počtu kroků j vyhovujícímu nerovnosti j+i<S, nelze Brentovu strategii uplatnit a nový prvek vložíme stejně jako původní metoda dvojí rozptylovací funkce.

Příklad:

a) Neúspěšné vyhledávání skončilo nalezením volného místa pro S=3(vertikálně). Hledání vhodného volného místa – I [i,j](horizontálně).

INC=Q(TAB[I_0].KLIC)   INC=Q(TAB[I_0].KLIC)

TAB [$I_Ø$]=obsazený ⟶ TAB[$I_{Ø,1}$]=obsaz ⟶ TAB[$I_{Ø,2}$]=obsaz

Q(K)   Q(TAB [I_1].KLIC)

TAB [I_1]=obsazený ⟶ TAB[$I_{1,1}$]=obsaz

TAB [I_2]=obsazený

TAB [I_3]=volný ;

"Horizontálně" se nejde dále, než je hranice $j < S-i$. V daném příkladě nebylo nalezeno vhodné volné místo a proto se provede TAB [I_3] := NOVÝ PRVEK, jako u původní metody.

b) Neúspěšné vyhledávání skončilo nalezením volného místa pro S=3.

INC=Q(TAB [I_0] .KLIC)   INC=Q(TAB[I_0].KLIC)

Q(K)   TAB [I_0] =obsaz. ⟶ TAB[$I_{Ø,1}$]=obsaz ⟶ TAB[$I_{Ø,2}$]=volný

Q(K)   TAB [I_1] =obsaz.

Q(K)   TAB [I_2] =obsaz.

TAB [I_3] =volný

Protože TAB [$I_{Ø,2}$] = volný a $(i = Ø)(j = 2)$ a tudíž $i+j < S$ provede se Brentův přesun:

TAB [$I_{Ø,2}$] : = TAB [$I_Ø$] ;
TAB [$I_Ø$] : = NOVÝ PRVEK ;

Důkaz Brentovy varianty:

Nechť $c = \sum_{i=1}^{w} p(K_i)$ je celkový počet porovnání pro vyhledání všech klíčů $K_1, \ldots, K_w$ v tabulce. Jestliže má každá položka <u>stejnou pravděpodobnost</u> být vyhledávána, pak c/w je průměrný počet porovnání pro vyhledání jednoho klíče. Cílem je tedy co nejmenší C. Novou položku musíme vkládat tak, aby přírůstek D pro určení nového c byl co nejmenší.

V případě a) bude p(K) = S+1 a tudíž D = S+1

V případě b) bude p(K) = i+1 ale z přesunu I_i té položky o j vzroste dále o tuto hodnotu, a tudíž D=i+j+1.

V případě b bude přírůstek menší, je-li splněno i+j S

6. Hodnocení uvedených metod

S výjimkou metody uvedené v odst. 2., nelze v uvedených tabulkách provést operaci DELETE jednoduchým zrušením. Jednou z možností je rozlišení stavu položky třístavovým indikátorem (VOLNÝ, OBSAZENÝ, ZRUŠENÝ) a "zaslepení" rušené položky.

Metody s explicitním zřetězením jsou ekonomické s ohledem na počet potřebných porovnání ve srovnání s metodami s implicitním zřetězením, ale spotřebují více paměti pro ukazatele. Při krátkých položkách se může ukázat, že je výhodnější větší tabulka s implicitním zřetězením, než menší tabulka s explicitním zřetězením.

Nejvýhodnější se jeví metoda s explicitním zřetězením s oddělenou oblastí přetečení. Potřebuje však mechanismus dynamického přidělování paměti.

Velmi účinná je i Brentova varianta, jejíž účinnost pro úspěšné vyhledávání neklesá se vzrůstajícím koeficientem zaplnění tabulky tak, jako u jiných metod s implicitním zřetězením.

Tabulky s rozptýlenými metodami mají také své nevýhody:

a) z neúspěšného vyhledání nelze získat žádnou dodatečnou informaci (např. nejbližší vyšší či nižší klíč)
b) dimensování polí pro tabulku není vždy snadné
c) údaje o době vyhledávání mají statistický charakter. Nejhorší případy se mohou lišit od průměrných hodnot. S tím je nutno počítat při použití těchto tabulek pro aplikace v reálném čase.

7. LITERATURA

1/1 Knuth,D.: The art of Computer programing
Vol.3.Sorting and searching
Addison - Wesley, 1973

/2/ Honzík,J. a kol.: Programovací techniky
VUT Brno, 1985

/3/ Colin Day,A.: Fortran techniques with special reference to
non-numerical application
Cambridge University Press 1972