9. přednáška

Vyhledávání III.
Řazení I.

Obsah přednášky:

- Tabulky s přímým přístupem, princip indexsekvenčního vyhledávání.
- Tabulky s rozptýlenými položkami (TRP).
- Vlastnosti a konstrukce rozptylovací (hashovací) funkce.
- TRP s explicitním a implicitním zřetězením synonym.
- Metoda dvojí rozptylové funkce. Hodnocení metod vyhledávání.

- Řazení I. Základní pojmy: stabilita, přirozenost, časová a prostorová složitost algoritmu řazení.
- Řazení podle více klíčů
- Řazení bez přesunu položek. MacLarenova metoda.

Přednáška IAL – 9.
Tabulky s přímým přístupem

Nechť existuje množina klíčů **K** dané tabulky a množina sousedních míst (adres) v paměti **H** které realizují vyhledávací tabulku. Existuje-li jedno-jednoznačná funkce mapující každý prvek první množiny do druhé množiny a naopak, pak hovoříme o tabulce s přímým přístupem.

Příklad: Kdyby množina klíčů byla dána intervalem 1..100 a tabulku reprezentovalo pole T[1..100], pak mapovací funkce pro klíč K je T[K]. Každý prvek pole musí mít Booleovskou složku „obsazeno“ nebo „prázdný“, která určuje, zda daný prvek na své adrese je, či není. Při inicializaci se nastaví všechny prvky tabulky na „prázdný“.

Pak vyhledání spočívá v přímém zjištění, zda na pozici klíče (indexu) dané tabulky je obsazeno a pak tam prvek je nebo je tomu naopak.

To je důvodem, proč se v řadě případů používá jako klíč hodnota integer. Časová složitost přístupu v přímé tabulce je 1.

Obtíž: Nalezení vhodné mapovací funkce.

Mapovací funkce

Hledání vhodné mapovací funkce je obtížné. Uvádí se, že pro 31 prvků, které se mají zobrazit do 41 prvkové množiny existuje 41^{31} tj cca 10^{50} různých možných mapovacích funkcí. Přitom jen (41!/31!) z nich dává odlišné hodnoty pro různé klíče (jsou jedno-jednoznačné). Poměr „vhodných“ funkcí ku všem možným je tedy asi 1:10 000 000.

Jevu, kdy se dva různé klíče namapují do stejného místa říkáme kolize.
Dvěma nebo více klíčům, které se namapují do téhož místa říkáme synonyma.

Paradox společných narozenin“

Je dobrá naděje, že mezi 23 osobami, které se sejdou ve společnosti, se najdou dvě osoby, které mají narozeniny ve stejný den“.

Jinými slovy: Najdeme-li náhodně funkci, která mapuje 23 klíčů do tabulky o 365 prvcích, je pravděpodobnost, že se žádné dva klíče nenamapují do stejného místa rovna 0.4927.

Dva požadavky na mapovací funkci:

- rychlost
- co nejméně kolizí

Mapovací funkce:

Je dáno mapovací pole intervalem nejčastěji [0..n] nebo [1..N].

Mapovací funkce transformuje klíč na index, jehož hodnota musí být v daném intervalu. Nejčastěji se mapovací funkce dělí do dvou etap:

- převod klíče na přirozené číslo (N>0)
- převod přirozeného čísla na hodnotu spadající do intervalu (nejčastěji s použitím operace modulo).

Nechť K je celé číslo větší než nula. Pak funkce

$$h(K) = K \mod (Max+1)$$

získá hodnoty z intervalu 0..Max

funkce

$$h(K) = K \mod Max + 1$$

získá hodnoty z intervalu 1..Max

Explicitní a implicitní zřetězení

- Při explicitním zřetězení obsahuje prvek adresu následníka.
- Při implicitním zřetězení je adresa následníka funkcí adresy předchůdce

Tabulka s rozptýlenými položkami sestává z mapovacího prostoru (pole) a ze seznamů synonym. Každý seznam začíná na jednom prvku mapovacího pole.

Vyhledávání v tabulce s rozptylovací funkcí (hashing table)

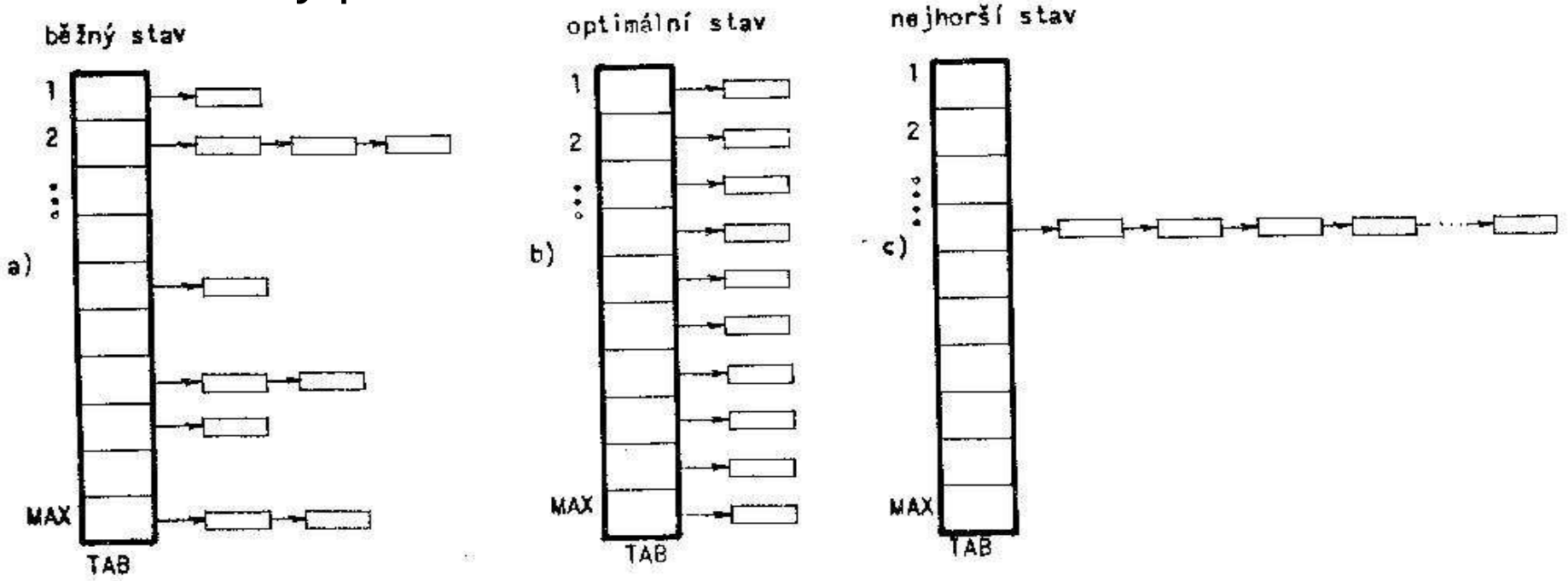
Princip vyhledávání v TRP spočívá ve dvou krocích:

- Mapovací funkce k danému klíčí nalezne index prvku v poli, který je začátkem seznamu synonym, které se namapovaly do téhož místa.
- Při sekvenčním průchodu tímto seznamem synonym vyhledáváme položku s daným klíčem.

Proto má vyhledávání v TRP indexsekvenční charakter.

TRP s explicitním zřetězením synonym

Maximální doba vyhledávání je dána délkou nejdelšího seznamu synonym. Rušení prvků je stejné jako ve zřetězeném seznamu. Seznamy prvků mohou být seřazeny podle klíče.

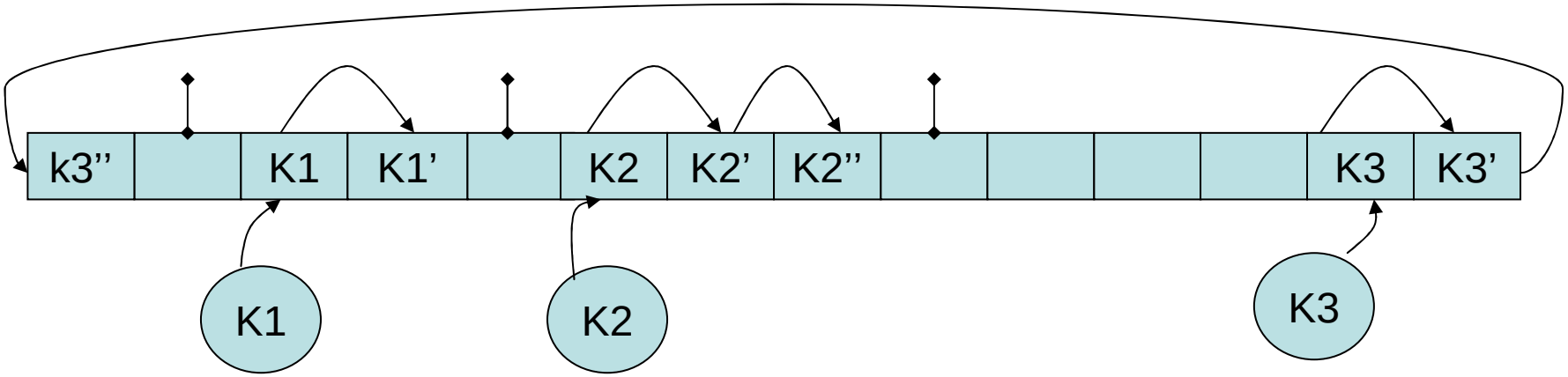


Klasifikace TRP na základě seznamu synonym

1. Explicitně zřetězený seznam
2. Implicitně zřetězený seznam (funkce kroku)
   A) krok je konstantní
      a) krok určuje programátor
      b) krok určuje program (druhá rozptylovací funkce)
   B) krok je proměnný (kvadratická metoda)

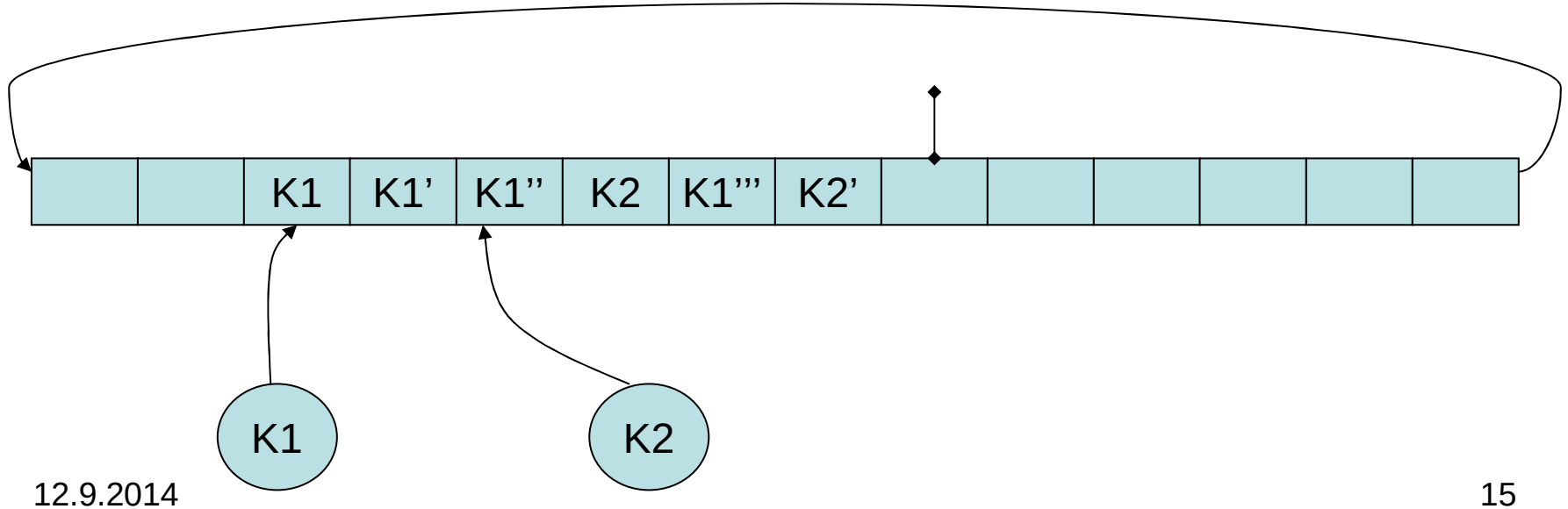
TRP s implicitně zřetězeným seznamem synonym

Explicitní zřetězení s funkcí pevného kroku = 1 a(i+1) = a(i) + 1



Konec seznamu synonym je dán prvním volným prvkem, který se najde se zadaným krokem. Nové synonymum se uloží na první volné místo (na konec seznamu). Tabulka (pole) musí obsahovat alespoň jeden volný prvek. Efektivní kapacita je o 1 menší než počet položek. Tabulka je implementovaná kruhovým polem.

Nechť jsou již do tabulky vloženy klíče K1, K1’ a K1’’. Následně se klíč K2 namapoval do položky, která je obsazena (je tam klíč K1’’). Klíč byl uložen na první volné místo. Další klíč K1’’’ se namapoval do položky K1. První volné místo pro klíč K1’’’ bylo nalezeno za K2. Klíč K2’ se namapoval do položky K2. První volné místo se nalezlo za klíčem K1’’’.
V této tabulce se dva seznamy synonym překrývají. Prvek K1’’ je vstupním bodem seznamu synonym K2. Nelze je zrušit ani zaslepením.



Velikost rozptylovacího pole

- Krok s hodnotou jedna má tendenci vytvářet “shluky“ (angl. cluster).
- Výhodnější je krok větší než 1. Takový krok by ale měl mít možnost „navštívit“ všechny položky pole. Kdyby pole mělo sudý počet prvků, pak sudý krok (např krok = 2) by z východiska lichého indexu mapovaného prvku mohl navštívit pouze liché indexy. Např. pole 1..10, krok 2, začátek na indexu 5 projde indexy: 5,7,9,1,3. Krok 4 a začátek na indexy 7 projde indexy: 7,1,5,9,3.

Kdyby měl krok hodnotu prvočísla, které je nesoudělné s jakoukoli velikostí pole, pak by mohl postupně projít všemi prvky pole.

Výhodnější ale je, aby hodnotu prvočísla měla velikost mapovacího pole. Pak jakýkoli krok dovolí projít všemi prvky mapovacího pole.

Je vhodné dimenzovat velikost mapovacího pole TRP tak, aby bylo rovno prvočíslu.

Kvadratická metoda

Mezi metody s automatickou změnou kroku patří tzv. „kvadratická metoda“. Hodnota kroku se v ní zvětšuje s každým krokem o 1. Při rozvinutí kruhového pole vytvářejí „navštívené“ adresy kvadratickou funkci.

Ve skriptech Vybrané kapitoly... je podrobnější popis dvou variant kvadratické metody.

TRP s dvojí rozptylovací funkcí

- Metoda dvojí rozptylovací (hashovací) funkcí patří mezi metody, v níž je krok v rozptylovacím poli určen programem – jeho druhou rozptylovací funkcí.
- Nechť má rozptylovací pole rozsah 0..Max, (kde hodnota Max+1 je prvočíslo).
- Nechť KInt je klíč transformovaný na celou hodnotu KInt>0.
- První rozptylovací funkce vytváří hodnotu z intervalu 0..Max ind= Kint mod (Max+1).
- Druhá rozptylovací funkce vytváří krok s hodnotou z intervalu 1..Max ind = Kint mod Max+1.

Vyhledání - Search

- Vyhledávací cyklus začíná na indexu získaném první rozptylovací funkcí. Končí úspěšně nalezením prvku s hledaným klíčem, neúspěšně při dosažení konce seznamu synonym (prázdným prvkem). Index následujícího prvku je dán přičtením kroku k aktuálnímu prvku při respektování kruhovosti pole:

$$ind_{i+1} = (ind_i + krok) \bmod Max$$

- Operace Insert vloží nový prvek na místo prvního prázdného prvku.

• Maximální kapacita TRP pro rozsah pole 0..Max je Max (o 1 menší než počet prvků) – alespoň jeden prvek musí zůstat jako „zarážka“ vyhledávání.

• TRP s implicitním zřetězením se používají v aplikacích, v nichž se nepoužívá operace Delete.

• Pro efektivní použití TRP s implicitním zřetězením se pole tabulky dimenzuje tak, aby její maximální zaplnění, dané poměrem $N_{akt}/(Max+1)$, nebylo větší než cca 0.6-0.7.

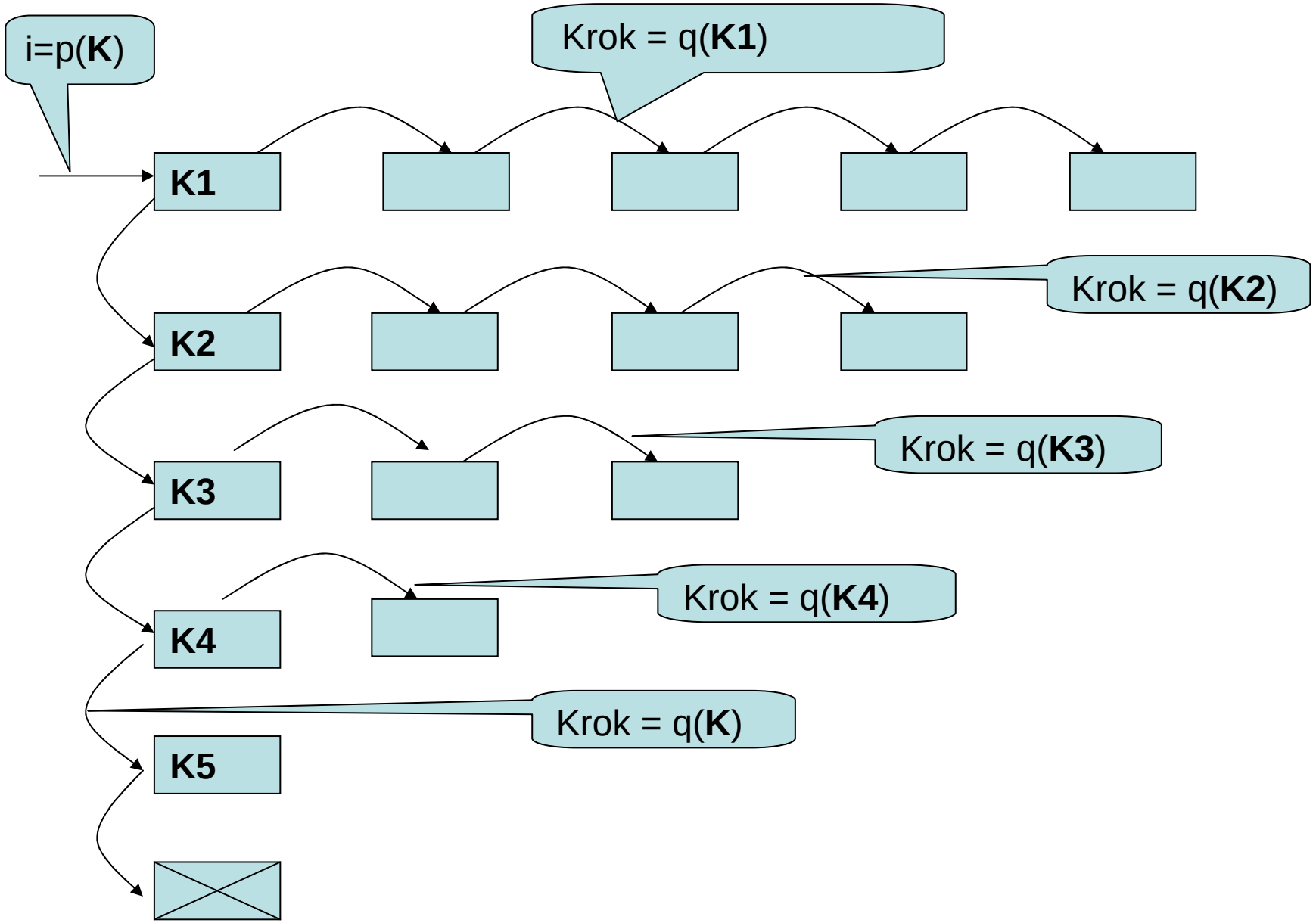
K domácímu zamyšlení

- Podívejte se na Knuthovu metodu TRP s explicitním zřetězením v poli. Proč nelze zrušit v této tabulce položku, i když je zřetězena explicitně?
- Proč nelze rušit položky v TRP s implicitním zřetězením jinak než zaslepením?

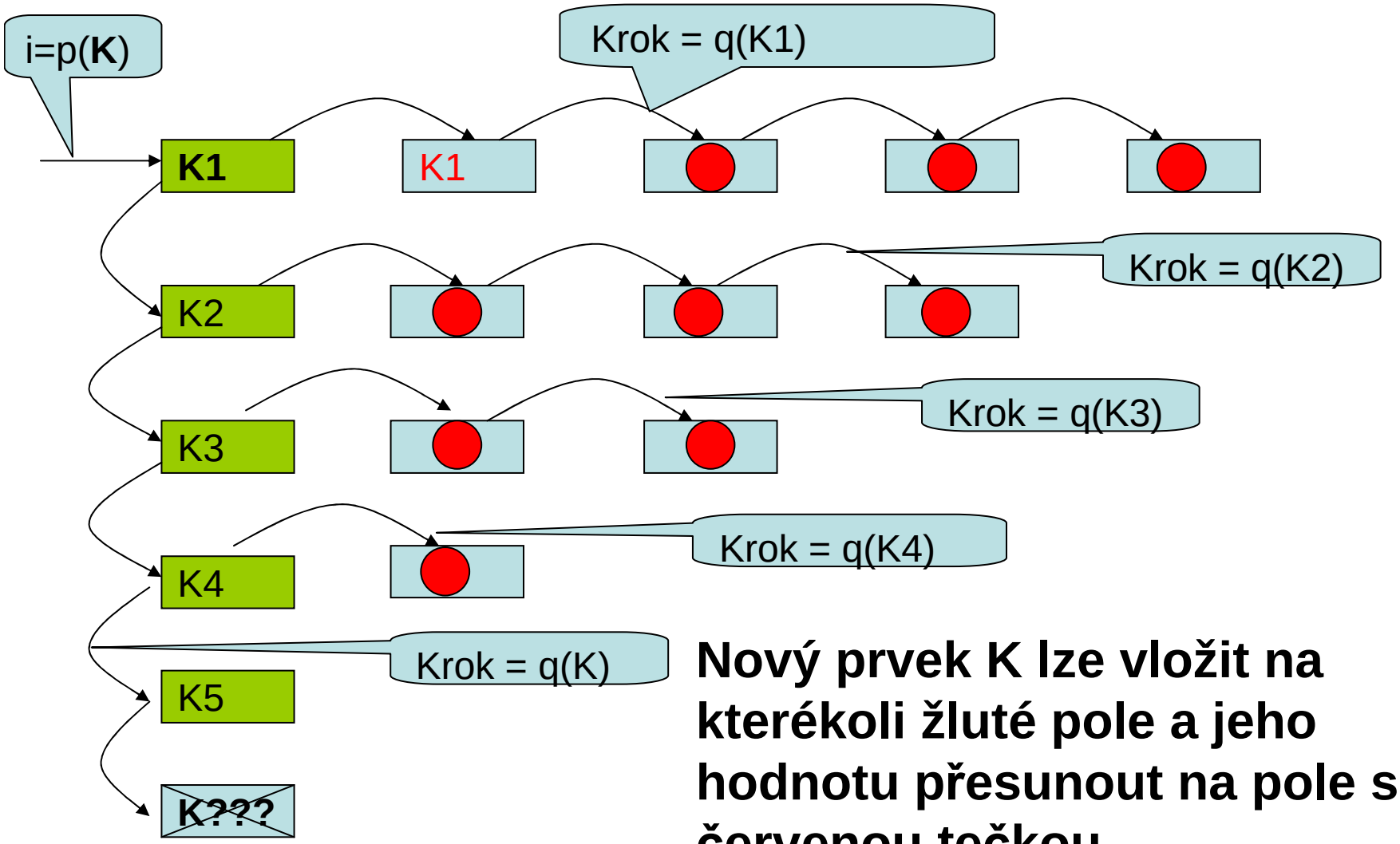
Brentova varianta

Brentova varianta je varianta metody TRP se dvěma rozptylovacími funkcemi. Princip vyhledávání (Search) je shodný s TRP se dvěma rozpt. funkcemi.

Brentova varianta je vhodná za podmínky, že počet případů úspěšného vyhledávání častější, než počet neúspěšného vyhledání s následným vkládáním. Brentova varianta provádí při vkládání rekonfiguraci prvků pole s cílem investovat do vkládání a získat lepší průměrnou dobu vyhledání.


i=p(**K**)
Krok = q(**K1**)
K1
Krok = q(**K2**)
K2
Krok = q(**K3**)
K3
Krok = q(**K4**)
K4
Krok = q(**K**)
K5

Prvek K1’ se přesune na první volné místo s krokem q(K1’)
a na jeho místo se vloží K.



Nový prvek K lze vložit na kterékoli žluté pole a jeho hodnotu přesunout na pole s červenou tečkou

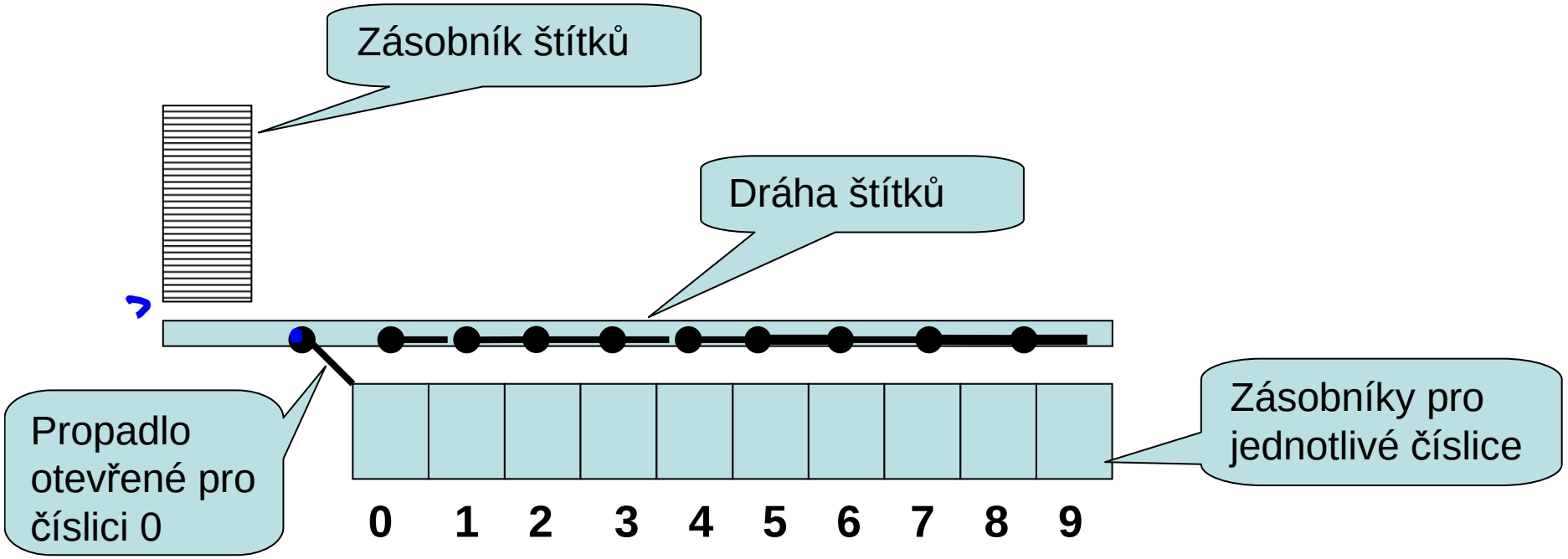
- Normální metoda by nový klíč vložila na první volné místo s krokem q(k) – zde po 6 krocích.
- Brentova varianta hledá první volné místo mezi červeně zakroužkovanými poli s krokem q(K1’), resp. q(K1’’) atd. Na toto místo vloží prvek K1’, resp. K1’’ atd. a na uvolněné místo vloží prvek K.
- Protože posun čelního prvku je menší než zde 5, je celková průměrná hodnota délky vyhledávání menší, než kdyby byl prvek K vložen na 6 pozici shora.

Hodnocení TRP

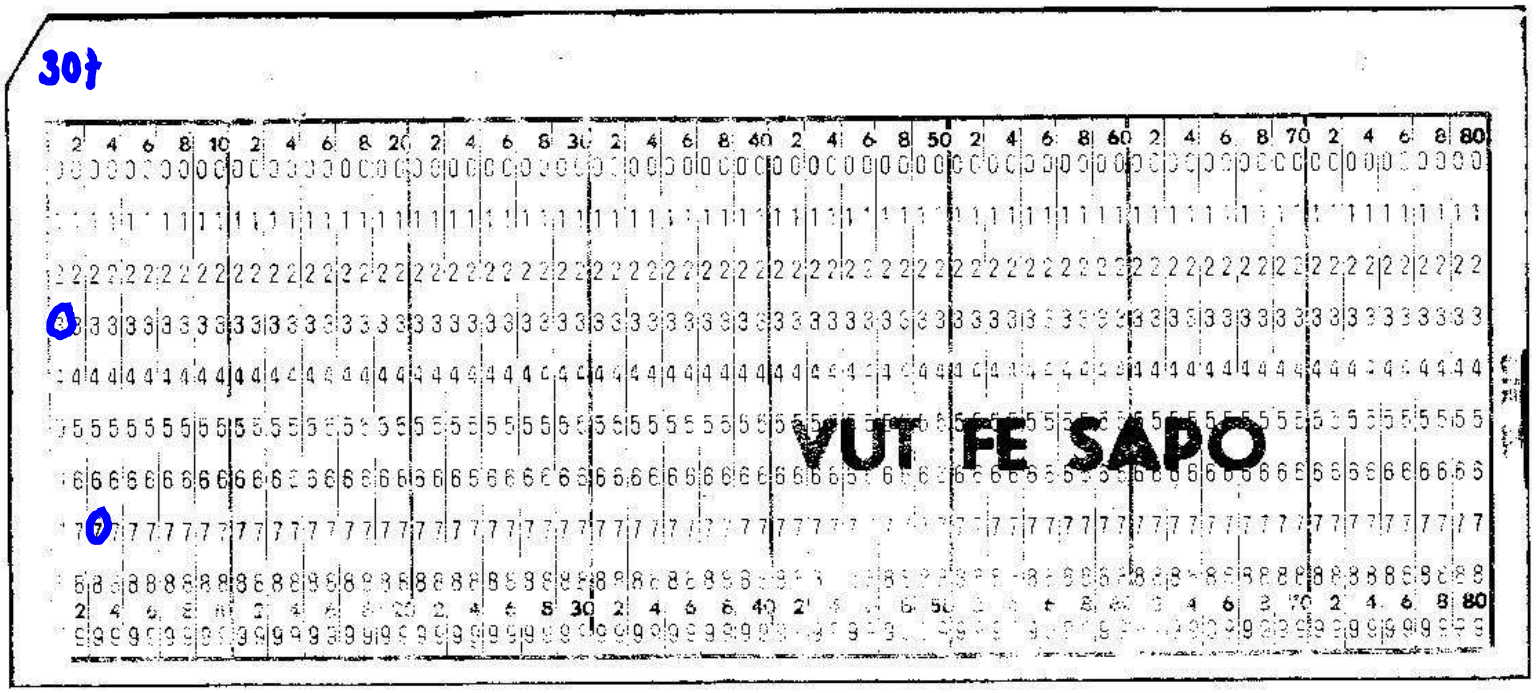
- Neexistuje obecné pravidlo jak nalézt nejvhodnější rozptylovací funkci. Dobrá rozptylovací funkce se může stanovit na základě znalosti vlastností množiny klíčů.
- Operaci Delete lze řešit pomocí „zaslepení“ – vložením klíče, který nebude nikdy vyhledáván.

Řazení

- Herman Hollerith použil „třídicí“ stroj pro sčítání obyvatelstva U.S.A. v r. 1890.



Třídicí stroj by použit pro seřazení děrných štítků podle hodnoty čísla, zapsaného pomocí dekadických číslic representovaných dírou na dané pozici v daném sloupci. Štítek Hollerith měl 90 sloupců a obdélníkové dírky. Štítek Aritma viz. obr. měl 80 sloupců a kulaté dírky. Číslice 0 se neděrovala.



Příklad: Seřazení štítků podle velikosti klíče, který byl reprezentován číslem vyděrovaným ve sloupcích 10,11,a 12 proběhlo ve třech etapách (dáno počtem sloupců čísla). V první etapě se štítky třídicím strojem roztřídily do 10 skupin od 0 do 9 podle sloupce s nejnižší prioritou – tedy podle sloupce 12. Z 10 balíčků štítkú se vytvořil jeden tak, že „nulový“ balíček byl vespod, „jedničkový“ byl nad ním atd. a „devítkový“ byl nahoře. Tento balík štítků se vložil do zásobníku, a začala druhá etapa – třídění podle sloupce 11 se stejným postupem. Na konci poslední etapy byl získán balík seřazených štítků. Řazení bylo provedeno tříděním

Terminologie

- Třídění (angl. sorting) položek neuspořádané množiny je uspořádání do tříd podle hodnoty daného atributu – klíče položky. Mezi třídami nemusí být definovaná relace uspořádání! (Mohu třídit směs jablek, hrušek a švestek do tří tříd).
- Pozn: Protože Hollerith dosahoval řazení pomocí třídění na třídicím stroji, používá se v praxi pro řazení v češtině i v angličtině nepřesné terminologie „třídění“ („sorting“). My budeme v předmětu IAL systematicky používat správnou terminologii. (Ani v tělocviku se neříká „setřiďte se podle velikosti...“).

- Uspořádání podle klíčů (collating) je seřazení položek podle uspořádané množiny klíčů. (Příklad: uspořádaná množina 4 barev je: [červená, modrá, fialová, růžová], nebo uspořádaná množina dekadických číslic je [2,4,5,3,1,7,9,0,8,6]. Tato uspořádaná množina definuje výsledné uspořádání.
- Řazení (ordering, sequencing) je uspořádání položek podle relace lineárního uspořádání nad klíči.

- Slučování (coalescing) je vytváření souboru položek sjednocením několika souborů položek téhož typu.
- Setřídění (v terminologii děrnoštítkové éry také „zakládání“ – angl. merging) je vytváření souboru seřazených položek sjednocením několika souborů položek téhož typu, které jsou již seřazeny.

- Sekvenčnost řazení vyjadřuje, že řadicí algoritmus přistupuje ke zpracovávaným prvkům sekvenčně (bez ohledu na to, že jsou např.uloženy v poli, které umožňuje náhodný přístup. Opakem sekvenčnosti je využití přímého (náhodného) přístupu a pak algoritmům říkáme „nesekvenční“.
- Přirozenost řazení je vlastnost algoritmu, při níž je doba potřebná k seřazení náhodně uspořádaného pole větší, než pole již uspořádaného a doba potřebná k seřazení opačně seřazeného pole je vetší, než doba seřazení náhodně uspořádaného. V opačném případě říkáme, že se algoritmus „nechová přirozeně“.

- Dohoda: Nebude-li stanoveno jinak, budeme předpokládat seřazení od nejmenšího k největšímu.
- Stabilita je vlastnost řadicího algoritmu, která vyjadřuje, že mechanizmus algoritmu zachovává relativní pořadí klíčů se stejnou hodnotou. Příklad: nestabilní algoritmus může uspořádat sekvenci: 7,5’,3,1,5’’,9,2,5’’’,8,4,6 s výsledkem: 1,2,3,4,5’’,5’,5’’’,6,7,8,9. Stabilní algoritmus vytvoří: 1,2,3,4,5’,5’’,5’’’,6,7,8,9.

Řazení podle více klíčů

- V praxi je řazení podle více klíčů velmi časté. Jako příklad mohou sloužit:
  - Řazení podle data narození, kde datum sestává ze tří číselných klíčů: rok, měsíc a den.
  - Řazení studentů podle čtyř klíčů: obor, ročník, studijní průměr a jméno. Úkolem je např. vytvořit seznam po oborech, v oboru po ročnících, v ročníku podle studijního průměru a studenty se stejným průměrem seřadit abecedně podle jména.

Problém lze řešit třemi způsoby:

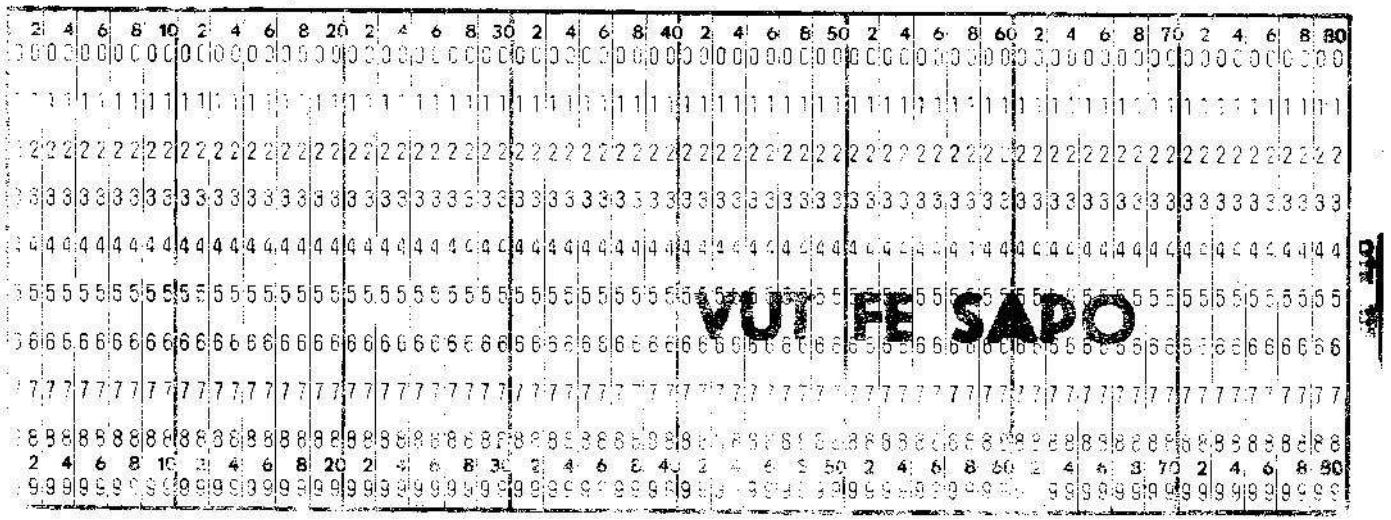
1. Vytvoření složené relace uspořádání:

```
function PrvniStarsi(Prv, Druh:TDatNar):Boolean;
(* false znamená, že druhý je starší, nebo jsou oba stejně staří *)
begin
     if Prv.Rok<>Druh.Rok
     then
        PrvniStarsi:= Prv.Rok<Druh.Rok
     else
        if Prv.Mes<> Druh.Mes (* rok je shodný *)
    then PrvniStarsi := Prv.Mes<Druh.Mes
        else  PrvniStarsi:= Prv.Den<Druh.Den
                (* měsíc je shodný *)
end;
```

2. Neuspořádanou množinu položek lze řadit postupně podle vzrůstající priority jednotlivých klíčů. Podmínkou je použití stabilní řadicí metody! Příklad:
   Skupinu osob lze seřadit podle stáří tak, ze se:
   1. Napřed seřadí podle dne data narozeni
   2. Pak se seřadí podle měsíce data narození
   3. Nakonec se seřadí podle roku data narození

   Tento způsob se podobá řazení děrných štítků v Hollerithově metodě.

Děrný štítek Aritma



K domácímu procvičení

a) Napište proceduru, která ze zadaného pole osob vytvoří seřazený seznam podle narozenin v roce. Při shodném datu narozenin má starší přednost.

```
type
   TDatNar = record
       Rok, Mes, Den:integer
   end;
   TOsoba = record
        Jmeno:string;
        DatNar:TDatNar
    end;
```

b) Napište proceduru libovolného algoritmu řazení pole, který znáte z prvního ročníku tak, aby se při volání procedury jedním vhodným parametrem ovládala složka, která bude klíčem řazení. Nechť pole je pole prvků typu TOsoba. Pak procedura:

```
procedure Razeni(var Pole:TPole, XX:TXX);
```

bude řadit jednou podle složky Rok, jindy podle složky Mes a jindy podle složky Den, v závislosti na parametru XX. Nalezněte pro tento účel vhodný typ a deklarujte ho. Trojí volání této procedury pokaždé podle jiné složky může vytvořit seznam podle stáří nebo seznam podle narozenin.

3. V praxi se často používá metoda „aglomerovaného klíče“. Uspořádaná N-tice klíčů se konvertuje na vhodný typ, nad nímž je definována relace uspořádání. V Pascalu je takovým typem typ string. Takovým aglomerovaným klíčem je např. rodné číslo. Rodné číslo lze pro řazení použít bez úpravy jako řetězec jen pro stejné pohlaví. Má tvar: RRMMDDXXXX, ale ženy mají MM zvýšené o 50 (žena narozená na apríla má r.č. např: 8454015471).

K domácímu procvičení

1. Napište funkci

function PrvniStarsi (RC1,RC2:string):Boolean;

2. Napište funkci

function MaDrivNarozeniny
(RC1,RC2:string):Boolean;

kde RC1 a RC2 jsou rodná čísla. V případě stejně starých osob nebo stejných narozenin má přednost žena před mužem. V případě rovnosti u stejného pohlaví rozhoduje pořadové číslo rodného čísla XXXX.

3. Je dán typ

```
type
   TObor= (infsys,intsys,pocsys,grasys);
   TStudent=record
      jmeno:string;
      obor:TObor;
      rocnik:integer;
      prumer:real
   end;
```

Vytvořte aglomerovaný (integrovaný) klíč pro vytvoření seznamů:

a) Podle oboru, v oboru podle ročníku, v ročníku podle průměru v průměru podle jména

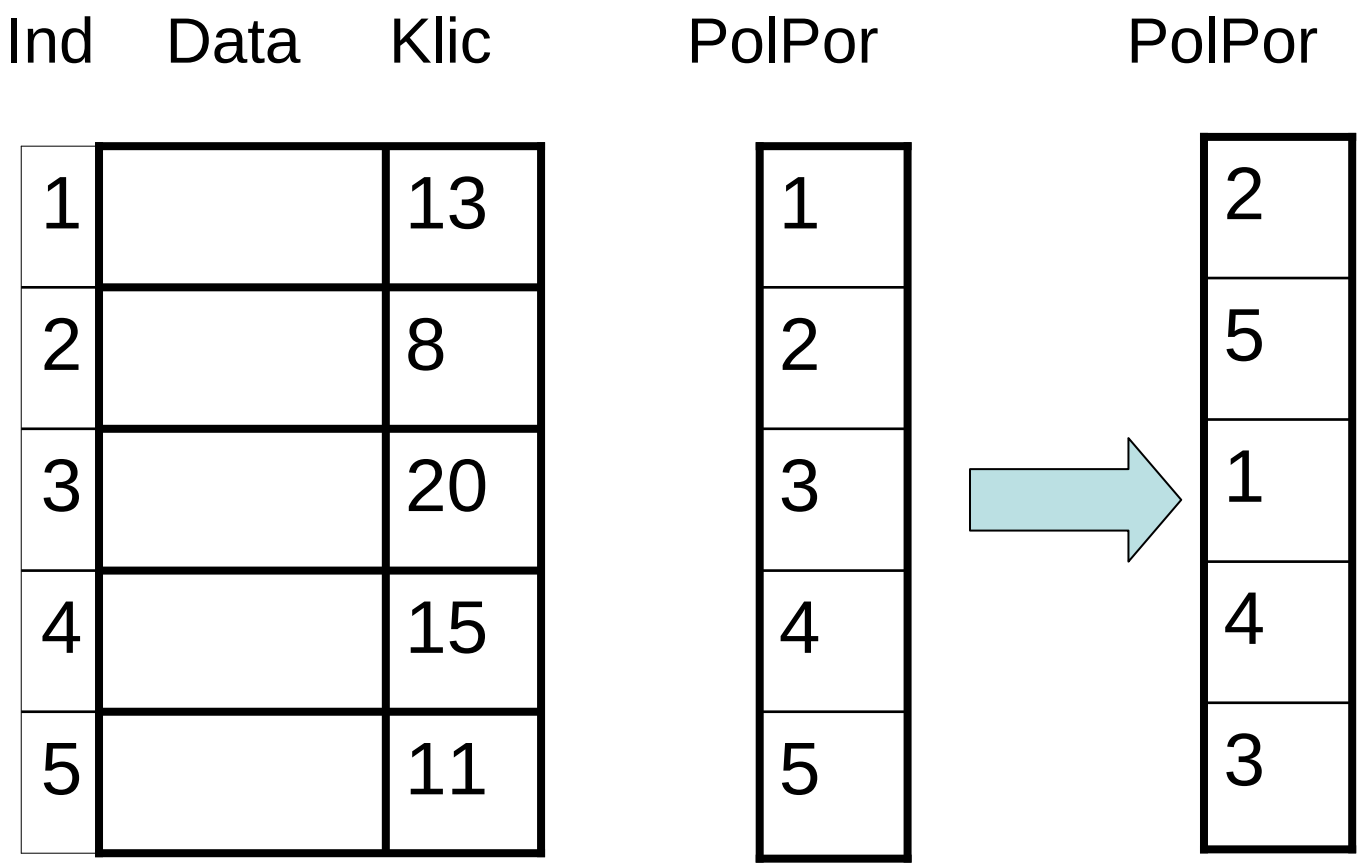
b) Podle průměru, v průměry oboru, v oboru podle ročníku, v ročníku podle jména.

Nápověda: Aglomerovaný klíč bude typu string. Průměr můžete převést na integer např: 2.75 ⇨ 275. String omezte na 20 znaků.

Řazení polí bez přesunu položek

- Nejčastěji prováděnými operacemi v algoritmech řazení jsou přesuny položek v poli a porovnávací operace. V případě „dlouhých“ položek jsou přesuny časově velmi náročné. Především tuto situaci, ale i některé jiné situace řeší řazení polí bez přesunu položek.

K danému poli se vytvoří tzv. pole pořadí (pořadník) PolPor. Inicializuje se hodnotami shodnými s indexem. Výsledkem řazení je pořadník, v němž jsou uspořádány (seřazeny) indexy prvků řazeného pole.



Nechť jsou dány typy:

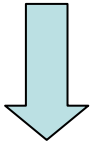
```
type
   TPolozka=record
       Data:TData;
       Klic:TKlic
    end;
    TPole=array[1..Max] of TPolozka;
     TPorad=array[1..Max] of integer;
var
   Pole:TPole;
   Porad:TPorad;
```

Pak inicializace má tvar:

```
for i:=1 to Max do Porad[i]:= i;
```

- Každá relace mezi dvěma prvky pole v normálním algoritmu řazení se v odpovídajícím algoritmu pro řazení bez přesunu transformuje tímto způsobem:

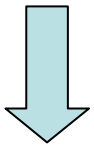
Pole[i].Klic > Pole[j]. klic



Pole[Porad[i]].Klic > Pole[Porad[j]].Klic

- Každá výměna dvou prvků i a j pole v algoritmu řazení s přesunem se v zápisu algoritmu řazení bez přesunu transformuje takto:

Pole[i] :=: Pole[j]



Porad[i]:=:Porad[j]

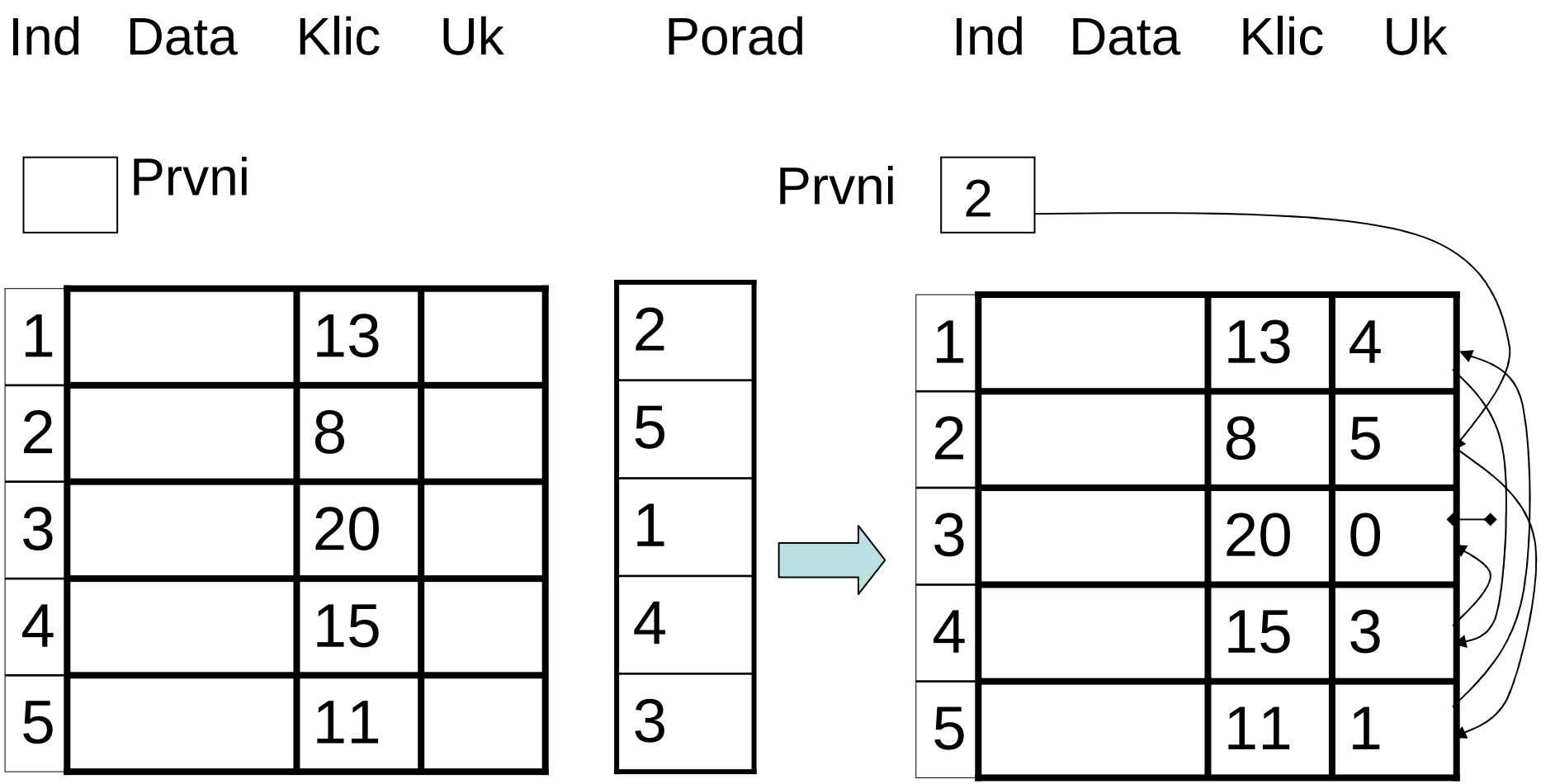
kde operace „:=:“ reprezentuje výměnu

Pole seřazené bez výměny položek lze průchodem vložit do výstupního seřazeného pole VystPole cyklem:

```
for i:=1 to Max do VystPole[i]:=
Pole[Porad[i]]
```

Pole seřazené pomocí „pořadníku“ lze také zřetězit a vytvořit seřazený seznam.

Zřetězení prvků pole seřazeného bez přesunu položek



Zřetězení provede úsek programu:

```
Prvni:=Porad[1];
for i:=1 to Max-1 do
   Pole[Porad[i]].Uk:=Porad[i+1];
Pole[Porad[MAX]].Uk:=0; (* 0 ve funkci nilu*)
```

Zřetězenou seřazenou posloupnost lze převést ze zdrojového pole do cílového pomocí jednoduchého cyklu, který je vhodným příkladem pro domácí procvičení.

MacLarenův algoritmus

MacLarenův algoritmus uspořádá pole seřazené bez přesunu na místě samém – neboli „in situ“, tedy bez pomocného pole.

```
i:=1;  Pom:=Prvni;
while i<Max do begin
(* Hledání následníka přesunutého na pozici větší než i*)
  while Pom<i do Pom:=Pole[Pom].Uk;
    (* výměna akt. prvního s akt. minimálním *)
     Pole[i] :=:Pole[Pom];
          Pole[i].Uk:=:  Pom; (* stejná výměna
  ukazatelů*)
     i:=i+1 (* prvních i-1 prvků je již na svém místě *)
end;
```

Komentář k MacLarenovu algoritmu.

První prvek seznamu (na který ukazuje proměnná Prvni) se vymění s prvkem pole na indexu 1. Tím se nejmenší položka dostane na své místo. Na prvek, který byl z prvního indexu pole odsunut jinam však některý prvek ukazoval. Je třeba ho najít a změnit jeho ukazatel tak, aby místo na první index ukazoval na místo, kam byl první odsunut.

Tím je první prvek ošetřen. Dalším „prvním“ se stane index o jednu větším a cyklus pokračuje tak dlouho, až se vymění předposlední (Max-1) prvek, kdy končí.

S ohledem na velkou délku položky je součet času řadicího algoritmu bez přesunu položek (minimálně linearitmický) s časem McLarenova algoritmu (lineární) kratší, než čas samotného řadicího algoritmu s přesunem položek.