Aplikovaná informatika

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

6

Obsah

- Algoritmy externího třídění,
  - jednocestné slučování,
  - dvoucestné slučování,
  - přirozené slučování,
  - Polyfázové slučování.
- Aplikace algoritmů externího třídění pro třídění interní:
  - Dvoucestné slučování (Merge Sort).
- Kombinované algoritmy

Definice datové struktury

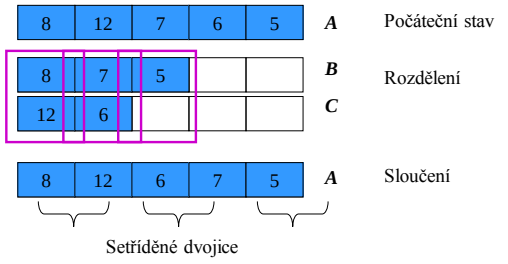
- Data tvoří záznamy (množiny prvků).
- Jeden z prvků je klíč třídění.
- Datový typ klíče je vždy jednoduchý.
- Klíč je neoddělitelný od zbytku záznamu.
- Záznamy jsou do sekvenčního souboru, který je možno číst pouze jednosměrně, po přečtení je prvek ze souboru automaticky odebrán.
- V algoritmu zjednodušeně pracujeme s prvkem pole jako by byl sám klíčem.

Dvoucestné slučování (2-way Merging)

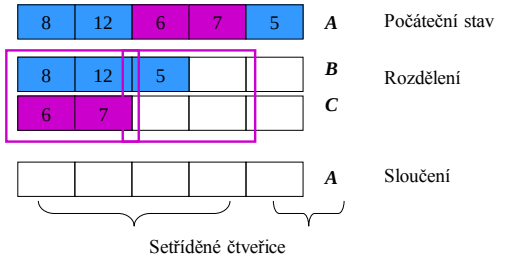
Myšlenka: Každý průchod programem lze rozdělit do fáze rozdělování a do fáze slučování. Ve fázi rozdělování jde o rozdělení souboru (***A***), do dvou dalších souborů (***B***), (***C***). Následovně je provedeno sloučení souborů ***B*** a ***C*** do souboru ***A*** tak, že vzniknou setříděné dvojice prvků. V dalším průchodu vzniknou setříděné čtveřice, pak osmice, atd. Celkově potřebujeme $\log_2 n$ průchodů pro setřídění celého souboru.

Dvoucestné slučování (2-way Merging)

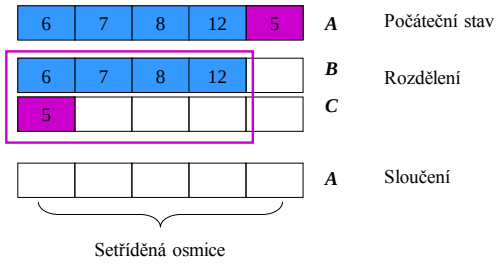
Příklad:



Dvoucestné slučování (2-way Merging)



Dvoucestné slučování (2-way Merging)



Dvoucestné slučování (2-way Merging)

- Analýza algoritmu:
  - Při každém průchodu se velikost setříděných skupin zdvojnásobí. Potřebujeme tedy $\lceil \log_2 n \rceil$ průchodů algoritmem.
  - Ve fázi slučování i rozdělování jsou přečteny a uloženy všechny prvky souboru.
  - Celkově potřebujeme $4n(\lceil \log_2 n \rceil)$ přístupů do souboru (čtení nebo zápis).

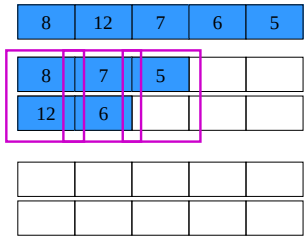
Jednocestné slučování (1-way Merging)

Myšlenka: Fáze rozdělování znamená pouze přepsání všech prvků ze zdrojového souboru do dvou cílových. Přidáním dalšího pomocného souboru by bylo možné obě fáze, rozdělování i slučování, spojit do jedné (kromě počátečního rozdělení).

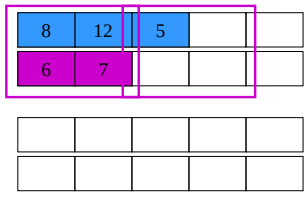
Tím se počet přístupů do souboru zmenší na polovinu.

Jednocestné slučování (1-way Merging)

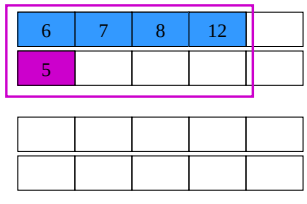
Příklad:



Jednocestné slučování (1-way Merging)



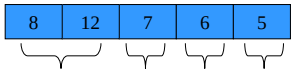
Jednocestné slučování (1-way Merging)



Přirozené slučování

Myšlenka: V náhodném souboru existují již přirozeně setříděné skupiny prvků (*n*-tice). Využijme toho a slučujme vždy tyto skupiny (skupina končí pokud za prvkem následuje prvek menší). Algoritmus se především liší ve fázi počátečního rozdělování, kdy rozdělujeme *n*-tice střídavě do obou souborů (***B*** a ***C***).

Příklad:



Výchozí soubor

Přirozené skupiny prvků

Polyfázové slučování

Myšlenka: Princip spočívá ve slučování z *n-1* souborů do *n*-tého souboru. Jakmile je některý ze vstupních souborů prázdný, přepíná se výstup do tohoto souboru a původní výstupní soubor se stává vstupním.

Základním problémem je počáteční distribuce *n*-tic do souborů.

Polyfázové slučování

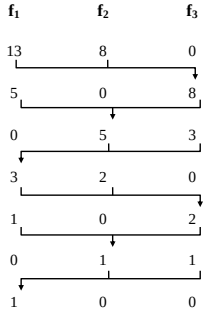
Příklad:

Optimální rozdělení *n*-tic pro polyfázové slučování se třemi páskami.

Počáteční počty *n*-tic jsou vždy dva sousední prvky Fibonacciho posloupnosti (1. řádu).

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55

kdy každé číslo (kromě prvních dvou) je součtem dvou předcházejících



Polyfázové slučování

- Pro více souborů platí, že počáteční počet *n*-tic se má rovnat součtu (*n*-1), (*n*-2), ..., 1 po sobě jdoucích Fibonacciho čísel (*n*-2). řádu. Obecně je Fibonacciho posloupnost čísel definována pro řád ***p*** :

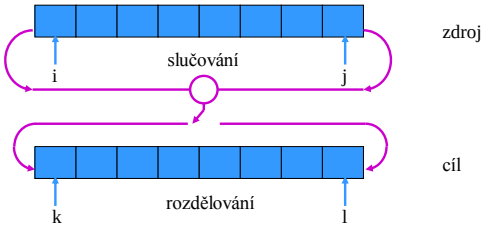
$$f_{i+1}^{(p)} = f_i^{(p)} + f_{i-1}^{(p)} + \ldots + f_{i-p}^{(p)} \qquad \text{pro } i \geq p$$

$$f_p^{(p)} = 1$$

$$f_i^{(p)} = 0 \qquad \text{pro } 0 \leq i < p$$

Dvoucestné slučování (Merge Sort)

Myšlenka: Algoritmy externího třídění je možno použít i pro třídění interní. Na pole pohlížíme jako na dva soubory – jeden čteme od začátku pole, druhý od konce pole. Problém – potřebujeme celkem čtyři soubory neboli dvě pole nebo jedno velikosti $2n$ prvků.



Dvoucestné slučování (Merge Sort)

Algoritmus: p – velikost setříděných skupin

```
nahoru = True; p = 1
while not p>=n
    if nahoru then
        i = 1; j = n; k = n+1, el = 2*n
    else
        k = 1; el = n; i = n+1, j = 2*n
    end if
    {sloučení p-prvkových posloupností z
     polí od i, j do k, el}
    nahoru = not nahoru; p = 2*p
end while
```

Dvoucestné slučování (Merge Sort)

```
{sloučení p-prvkových posloupností z
     polí od i, j do k, l}
h = 1; m = n
while not m=0
    if m>=p then q = p else q = m end if
    m = m - q
    if m>=p then r = p else r = m end if
    m = m - r
    {sloučení q prvků od i a r prvků od j
     cíl určen k, přírůstek h}
    h = -h
    výměna obsahu k a el
end while
```


Dvoucestné slučování (Merge Sort)

```
{sloučení q prvků od i a r prvků od j
     cíl určen k, přírůstek h}
while (q<>0) and (r<>0)
    if a(i)<a(j) then
        {přesuň prvek od i ke k}
        {nastav nové i a k}
        q = q - 1
    else
        {přesuň prvek od j ke k}
        {nastav nové j a k}
        r = r - 1
    end if
end while
{zkopíruj konec posloupnosti od i, totéž od j}
```


Dvoucestné slučování (Merge Sort)

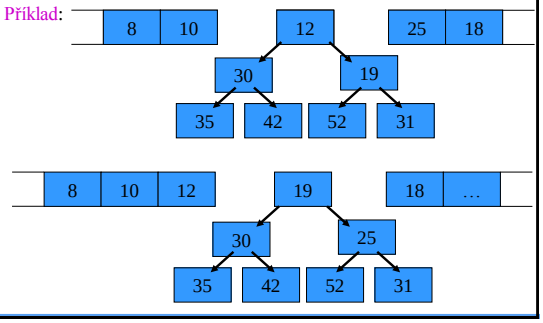
```
nahoru = True; p = 1
while not p>=n
  h = 1; m = n
  if nahoru then
    i = 1; j = n; k = n+1, el = 2*n
  else
    k = 1; el = n; i = n+1, j = 2*n
  end if
  while not m=0
    if m>=p then q = p
            else q = m
    m = m - q
    if m>=p then r = p
            else r = m
    m = m - r
    while (q<>0) and (r<>0)
      if a(i)<a(j) then
        a(k) = a(i); k = k + h
        i = i + 1; q = q - 1
      else
        a(k) = a(j); k = k + h
        j = j - 1; r = r - 1
      end if
    end while  '(q<>0) and (r<>0)
    while r<>0 'zbytek posloupnosti od j
      a(k) = a(j); k = k + h
      j = j - 1; r = r - 1
    end while
    while q<>0 'zbytek posloupnosti od i
      a(k) = a(i); k = k + h
      i = i + 1; q = q - 1
    end while
    h = -h; t = k; k = el; el = t
  end while       'not m=0
  nahoru = not nahoru; p = 2*p
end while         'not p>=n
if not nahoru then 'přesun z horní části dolů
  for i = 1 to n
    a(i) = a(i+n)
  end for
end if
```

Externí třídění s využitím paměti

Myšlenka: Algoritmy externího třídění jsou koncipovány pro minimální využití operační paměti. Ve skutečnosti je ale operační paměť k dispozici a může být využita pro zefektivnění algoritmu. Například prodloužením *n*-tic polyfázového slučování aplikací hromady.

Externí třídění s využitím paměti

Příklad:



Hledání minima, maxima, mediánu

- Hledání minima, maxima
  - aplikace třídění hromadou.
- Hledání mediánu (prvek ležící v setříděné posloupnosti uprostřed)
  - aplikace třídění rozdělováním.