Externé triedenie

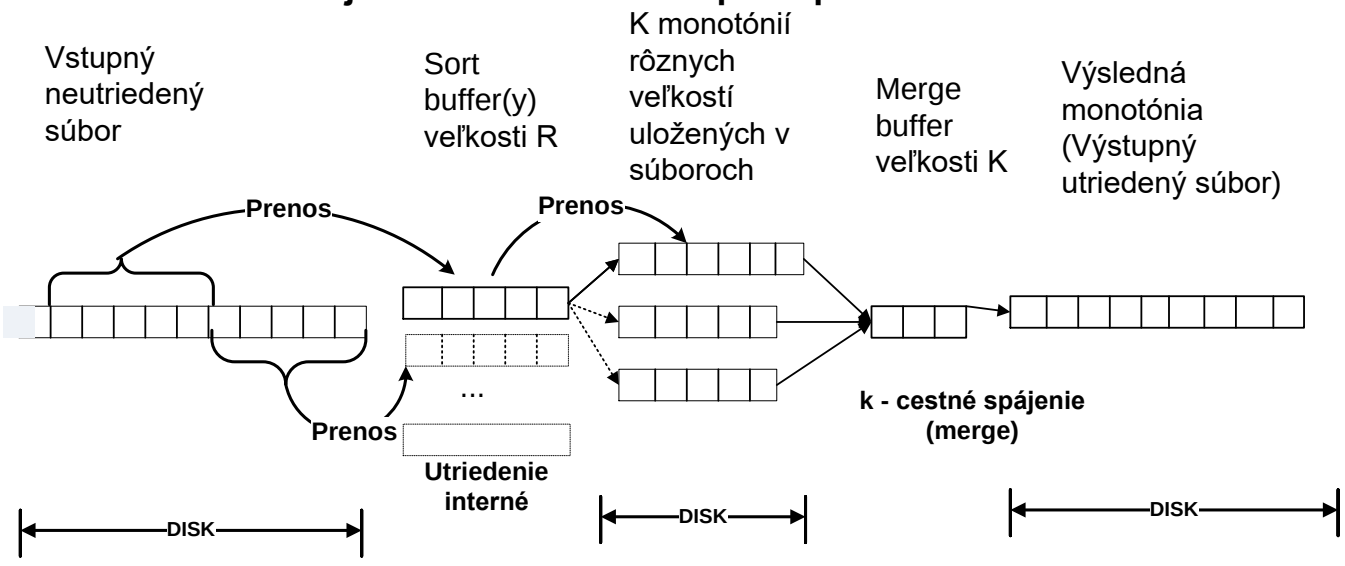
Situácia:

- **máme lineárny neutriedený súbor s možnosťou sekvenčného (sériového) prístupu a potrebujeme ho utriediť**
- **pamäťové možnosti nám nedovolia utriediť ho naraz v operačnej pamäti (napr. Quick-sortom) a triedenie quicksortom na disku by bolo z hľadiska počtu prístupov na disk neefektívne**
- **snahou je minimalizovať počet čítaní a zápisov na disk pri čo najmenšej spotrebe pamäte**

Využitie: Napríklad v procese budovania B$^+$stromu z neutriedeného súboru.

Predpoklad: externé médium = disk

Zjednodušená schéma postupu



Bežný je dvojfázový postup:

- **1. fáza (predtriedenie): transformácia vstupného neutriedeného súboru na *K* utriedených častí – výstupných súborov (tzv. monotónie, runs). Monotónie nemusia byť rovnakej dĺžky a ich počet je závislý na poradí kľučov v neutriedenom súbore. Snažíme sa ich počet minimalizovať, v druhej fáze budeme totiž potrebovať „merge buffer“ s veľkosťou rovnou počtu monotónií. Vždy potrebuje aspoň**

jeden „sort buffer“, ale je možné využiť aj viac „sort bufferov“ súčasne a vytvárať tak niekoľko monotónií súčasne.

- **2. fáza (spájanie, merge): transformácia *K* utriedených čiastkových súborov na jeden utriedený výstupný súbor.**

Je potrebné vyriešiť problém vyhľadania minimálneho prvku z N neusporiadaných prvkov v bufferi.
Na toto interné predtriedenie možno použiť napr. Quicksort, alebo vhodnú implementáciu prioritného frontu, keďže dominantné sú operácie Vlož a VyberMinimálny. V príklade uvedenom nižšie využívame implicitnú haldu.
Kritérium efektivity je počet „prechodov“ záznamu operačnou pamäťou.
V praxi sa odporúča N rozumne ohraničiť, alokácia obrovského bufferu nie je vhodná.

Pre proces k - cestného spájania sa použije K bufferov pre anticipované čítanie (pre každú monotóniu jeden) a 1 buffer pre kumulovaný zápis (pre výsledný utriedený súbor).

Je možné všetky monotónie naukladať aj do jediného súboru (pri použití jediného „sort bufferu“), pričom je potrebné si pamätať diskové adresy ich začiatkov. Nevýhoda tohto prístupu je nemožnosť použitia anticipovaného čítania.

Všetký dáta sa dva razy načítajú a dva razy zapíšu. Výsledkom je jediný utriedený súbor.

<u>Algoritmus k - cestného spájania (2 fáza algortimu):</u>

1. **Prenes jeden prvok z každej monotónie do „merge buffera“.**
2. **Najmenší prvok z buffera prenes do výstupnej monotónie a nahraď ho ďalším prvkom z tej monotónie, z ktorej pochádza prenesený prvok, ak taký existuje, inak vlož ∞.**
3. **Ak v bufferi existuje iba jeden prvok ≠ ∞, prenes do výstupnej monotónie všetky prvky zo vstupnej monotónie odpovedajúcej tomuto prvku – koniec algoritmu, inak choď na krok 2.**

Ukážka tvorby monotónií (1 fáza algoritmu) pri použití jediného „sort buffera“ veľkosti 5:

Vstupný neutriedený súbor:

18 14 19 13 17 16 9 6 1 7 15 3

18 14 19 13 17

Naplň „sort buffer“

13 14 19 18 17

Transformuj buffer na implicitnú haldu

Minimálny prvok (13) dáme do monotónie 1 a náhradíme ho ďalším, nasleduje preosiatie, aby opäť išlo o implicitnú haldu.

14 16 19 18 17

Halda Neutr.

Minimálny prvok (14) zaradíme do monotónie 1 a nahradíme ho ďalším. 9 nemôže patriť do monotónie 1, keďže je menšia ako najväčší prvok z tejto monotónie (14). Vymeníme ho s posledným a preosejeme tak, aby boli haldovo usporiadané prvé 4 záznamy.

16 17 19 18 9

Halda Neutr.

17 18 19 6 9

Ďalšie prípady rovnaké

Halda Neutr.

18 19 1 6 9

Halda Neutr.

19 7 1 6 9

Neutr.

Monotónia bude narastať až pokým nebudú všetky záznamy v buffery menšie ako najväčší záznam z monotónie! Následne začneme vytvárať ďalšiu monotóniu.

15 7 1 6 9

Transformuj buffer na implicitnú haldu

Halda

1 6 15 7 9

Minimálny prvok (1) presunieme do monotómie 2, náhrada ďalším (3), preosiatie

3 6 15 7 9

Monotónia 1

13
14
16
17
18
19

Monotónia 2

1
3

<u>Ukážka k - cestného spájania vytvorených monotónií (2 fáza algoritmu):</u>
Výsledkom 1 fázy vznikli monotónie: 13, 14, 16, 17, 18, 19 a 1, 3, 6, 7, 9, 15.
Použijeme merge buffer (prioritný front) o veľkosti rovný počtu monotónií, teda 2.
Naplníme ho najmenším prvkom z každej z nich.
Buffer = {13, 1}
Vyberieme najmenší prvok z buffera (v tomto prípade 1), zapíšeme ho do výstupného utriedeného súboru a do buffera vložíme najmenší prvok z rovnakej monotónie (teda 3).
Výstupný súbor = {1}
Buffer = {13, 3}

Postupujeme ďalej:
Výstupný súbor = {1, 3}
Buffer = {13, 6}

Výstupný súbor = {1, 3, 6}
Buffer = {13, 7}

Výstupný súbor = {1, 3, 6, 7}
Buffer = {13, 9}

Výstupný súbor = {1, 3, 6, 7, 9}
Buffer = {13, 15}

Výstupný súbor = {1, 3, 6, 7, 9, 13}
Buffer = {14, 15}

Výstupný súbor = {1, 3, 6, 7, 9, 13, 14}
Buffer = {16, 15}

Výstupný súbor = {1, 3, 6, 7, 9, 13, 14, 15}
Buffer = {16, ∞}

Nastáva situácia z bodu 3 a teda môžeme všetky prvky zo zostávajúcej monotónie preniesť do výstupného súboru.
Poznamka: Použitie hodnoty ∞ je v texte použité najmä pre názornosť, nie je nutné ju pri praktickom použití vkladať do bufferu.