



PA152: Efektivní využívání DB
7. Třídění

Vlastislav Dohnal

Poděkování

- Zdrojem materiálů tohoto předmětu jsou:
 - Přednášky CS245, CS345, CS345
 - Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom
 - Stanford University, California
 - Přednášky dřívější verze PA152
 - Pavel Rychlý
 - Fakulta informatiky, Masarykova Univerzita

Použití algoritmů třídění (sorting)

- Výsledek
 - ORDER BY
- Operace spojení
 - JOIN
- Odstranění duplicit
 - DISTINCT

Předpoklady

- Operační paměť
 - Omezená velikost – M bloků
- Data uložena na disku
 - Vstup (relace) je čtený z disku
- Výsledek zůstává v paměti
 - Výstup je obvykle dále zpracováván
- Náklady na třídění (sorting)
 - Počet čtení z disku

Třídění v paměti

■ Mnoho algoritmů

- BubbleSort – $O(n^2)$
- QuickSort – $\Theta(n \log n)$
- MergeSort – $O(n \log n)$
- InsertSort – $O(n^2)$
- HeapSort – $O(n \log n)$
- RadixSort – $O(kn)$
- Counting Sort – $O(k+n)$
- ...

Příklady

■ Counting Sort

- Malý počet různých hodnot
- Potřebujeme seřadit 100 známek (A-F)
 - Vytvoříme pole pro jednotlivé známky
 - Spočítáme počty jednotlivých známek
 - Zapíšeme výsledek

■ Radix Sort

- Rekurzivní řazení po jednotlivých bajtech (bitech)
- Po bajtech aplikuj Counting Sort
 - První průchod zjistí počty
 - Druhý průchod umístí data na správné místo

Vlastnosti třídících algoritmů

- Data v operační paměti
- Většinou třídí na místě (in-place)
- Potřebují málo další paměti ($\log n$)

Malá operační paměť

■ Komprese dat

- Zpracovávat pouze klíče s ukazateli na záznam, nikoli celé záznamy
- Ok, ale při výstupu stejně musím záznamy číst → náhodné čtení

■ Virtualizace paměti

- Většinou pomalé → příliš mnoho V/V

■ Úprava algoritmu

- Spojení více algoritmů
- Většinou se používá MergeSort a QuickSort

MergeSort – v paměti

■ Princip „rozděl a panuj“

□ Rozděluj na poloviny

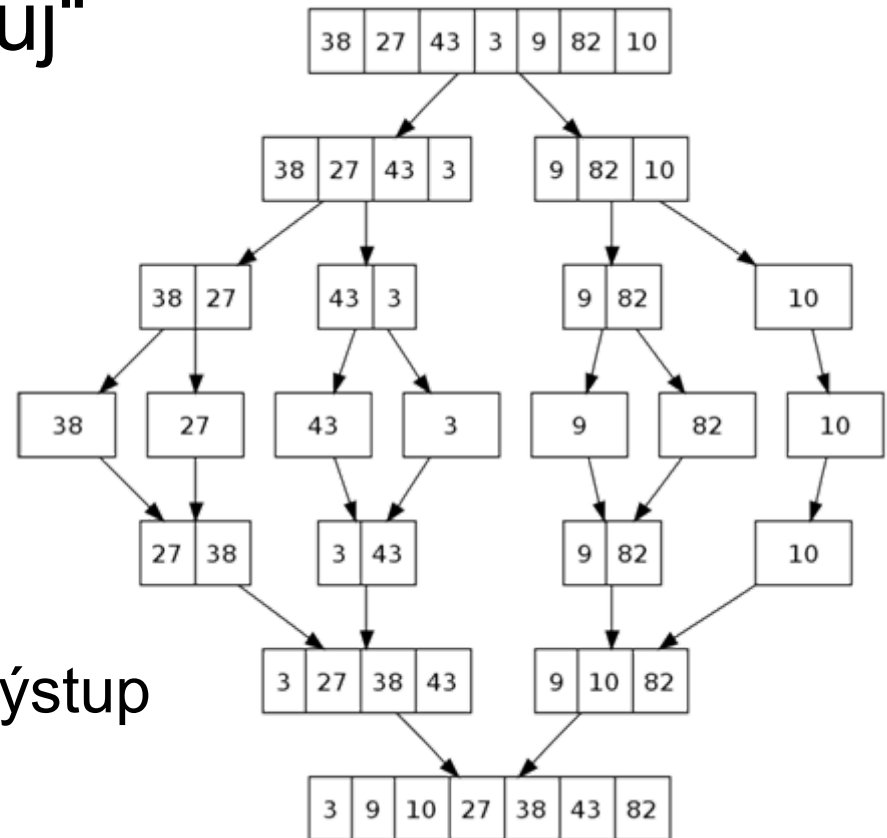
■ Až na jednotlivé klíče

□ Spojuj uspořádané části

■ Lineární průchod oběma částmi

■ Nejmenší dávám na výstup

■ $O(n \log n)$



Převzato z Wikipedia.org, MergeSort Algorithm

MergeSort – varianta

- Dvoufázové třídění, vícecestné slévání
 - Two-Phase Multiway MergeSort
- Princip
 1. Rozděl na dávky o velikosti volné RAM
 2. Postupně dávky v paměti uspořádej
 3. Všechny dávky naráz spoj

Dvoufázový MergeSort

■ Příklad

- Relace 100 mil. záznamů, každý 100 bajtů
- Blok má 4kB, tj. 40 záznamů
 - Relace uložena na 250 000 blocích
- Operační paměť pro třízení
 - 50MB, tj. 12 800 bloků

■ Fáze 1

- $250\,000 / 12\,800$ dávek = 20 dávek
 - Poslední má pouze 6 800 bloků
- V/V: čtení 250 000 + zápis 250 000

Dvoufázový MergeSort

■ Fáze 2

- Původní spojování po dvou dávkách je pomalé!
 - Tj. $\log_2(\text{počet dávek})$ čtení a zápis celé relace
 - Pro 20 dávek – $5\times$ čtení a zápis celého souboru
- Vícecestné spojování
 - Čti všechny dávky po jednom bloku
 - Proveď spojení do výstupního bloku

Dvoufázový MergeSort

■ Fáze 2

□ Opakuj

- Najdi nejmenší hodnotu ze všech dávek
- Zapiš do výstupního bloku
 - Plný blok → zápis na disk
- Prázdný blok dávky → načtení dalšího bloku

□ Výsledkem 1 čtení a 1 zápis pro celou relaci.

■ Celkově

□ $4 \cdot B(R)$ náhodných V/V , tj. $O(n)$

Dvoufázový MergeSort – omezení

■ Parametry

- M – operační paměť v blocích
- $B(R)$ – velikost relace R v blocích

■ Omezení

- Max. počet dávek: $M-1$
- Max. délka dávky: M
- Max. velikost relace: $M \cdot (M-1)$

■ Pro náš příklad 100B záznamů

- Max. 151 027 200 bloků
- Max. 6 041 088 000 záznamů