Kap. 7 Sortierverfahren
Kap. 7.0 Darstellung, Vorüberlegungen

$\Sigma = \{(x,\alpha)\}$ mit <

internes Sortieren	externes Sortieren
Σ paβt	Σ paβt nicht in AS

Darstellung von <:

3 6 1 7 ungeordnet, < berechnen

1 3 6 7 < ~ phys. Reihung

⊡ 3 6 1 7 Liste verkettet

sortierter Baum mit Inordnung
- mit Zeigern
- Adreβrechnung

1

1. **Einfache Feldorganisation:**

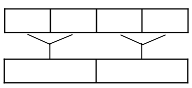
   S[1], ..., S[n]

   Für Sortieren am Platz: Tausch-Operation!

2. **Doppelfeld:**

   S[1], ..., S[n]

   T[1], ..., T[n]



für Misch-Verfahren:

Speicher: 2n;

Zeit: $O(n \log n)$

Sätze variabler Länge?

2

3. Listenorganisation:

(x, α) erweitern um Verweis v auf Nachfolger

Tauschoperation?
Anpassung von Quicksort?

4. Zeigerfeld: für lange Sätze und

Sätze variabler Länge. Sortiere Zeiger!

$v_x \longrightarrow (x, \alpha)$

v_x gespeichert in Feld V[1] ... V[n]

$v_x <\cdot\, v_y \;\Leftrightarrow\; x < y$

sortiere V nach $<\cdot$, kopiere S nach $<\cdot$

3

5. Feld von Paaren: (x, v_x)

falls α variabel lang
x feste Länge
speichere (x, v_x) in V,
sortiere V nach < auf X.

Auch o.k. falls Σ nicht in AS paβt,
aber $\Sigma_v = \{(x, v_x)\}$ paβt

Problem: gut für random Zugriff auf Σ, aber sequentielle Verarbeitung von Σ nach < problematisch: 1 Plattenzugriff pro Element von Σ.

4

Kap. 7.1 Zu internen Sortierverfahren

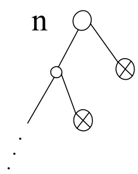
Einfache Verfahren mit $O(n^2)$

- Auswahl
- Einfüge (bubble)
- Austausch

Quicksort: Zeit?

Speicher: Kaskadenrekursion erfordert Rückstellung einer

Partitionsklasse für Laufzeitkeller



$\otimes$ zurückgestellt, Kellertiefe?

? 2n

? *n log n*

Baumsortierung: Heapsort (Stadelsort)

Speicher: am Platz, schichtenweise sequentielle Darstellung des Binär-Baums

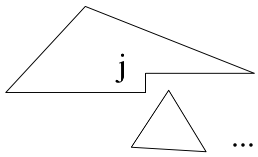
Zeit: $O(n \log n)$ garantiert

Zentral für externe Sortierverfahren

Umwandlung heap in sortiertes Feld:

M [1:n] Baum $\Rightarrow$ M [1:i] mit $i \leq n$ ist Baum mit denselben Vater-Sohn Beziehungen

Lemma: Falls M[1:n] Stadel ist, ist auch M[1:i] und M[j:n] Stadel für alle i, j.



Sort – Merge intern:

Zeit: $O(n \log n)$

Speicher: 2n bzw. 1.5 n

7

Kap. 7.2 Externes Sortieren

Phase 1: Initialläufe: sortierte Teilmengen durch internes Sortieren

$A = (a_1, a_2, \ldots a_{e_1})$; i < j $\Rightarrow$ $a_i \leq$ a

$B = (b_1, b_2, \ldots, b_{e_1})$

$C = \overline{AB}$ durch Mischung (merge) von A und B

Seien a_j, b_i kleinste, ungemischte Elementen von A, B

$$c_{k+1} := \min\{a_j, b_i\}$$

Bei Mischung von m Initialläufen z.B. m = 5000

$$c_{k+1} := \min_{i=1\ldots5000} \{a_i\}$$

verwende Heap!

8

Phase 2: Reines Mischen: Datei mit N Elementen

0. Läufe der Länge $1 = 2^0$ vorhanden
1. Läufe der Länge $2 = 2^1$

$\vdots$

d. Läufe der Länge 2^d

für kleinstes d mit $2^d \geq N \quad \Rightarrow d = \lceil \log_2 N \rceil$

Ziel: bei 1. Durchgang möglichst lange Initialläufe durch internes Sortieren

Bei S Speicher:

Länge S: in place Verfahren, feste Satzlänge

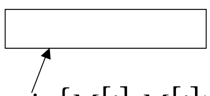
Länge S/2: internes Mischen

Länge 2S: Stadelsort, erspart bis zu 2 externe Mischdurchgänge

9

Mit Auswahlverfahren:

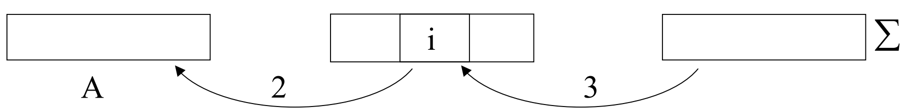
M [1] ... M [s]



$\min_i \{M[i], M[i] \geq y\}$

1. min. suchen i
2. Ausgabe: y := M[i]
3. Eingabe: M[i] := x

x kann an nächster min-Suche teilnehmen, falls $x \geq y$



Problem: min-Suche in M von O(S)

10

Mit Heap:

1	i ... s
1 Stadel für Ausgabekandidaten	viele Stadel für zu kleine Schlüssel
für Elemente x mit	für Elemente x mit
$x \geq y$	$x < y$

11

Algorithmus für Initialläufe:

```
var L: {für letzten ausgegebenen Schlüssel}
    H: {für neue Eingabe von Σ}
    {Stadel in M[1:S]}
starte neuen Initiallauf:
i := S ;
while i > 0 do   {Ausgabestadel M [1:i] ≠ ϕ}
begin L := M [1]; gebe L aus;
      H := Eingabe vom Σ;
      if H ≥ L then {H noch zu Initiallauf}
         begin M[1] := H; SENKE (1,i) end
      else begin M[1] = M[i];
              M[i] = H;
              SENKE (i,S);
              i := i-1;
              SENKE (1,i)   Ver-
                            besserung?
           end
end        {keine E/A Stockung!!!}
```

12

Externes Mischen mit Bändern

2 Wege, 3 Bänder:

$U = A_1, A_2, A_3$

$V = B_1, B_2, B_3$

$W = C_1 = \overline{A_1B_1}, C_2 = \overline{A_2B_2}, C_3 = \overline{A_3B_3}$

Merge von C_1, C_2, C_3?

1. Symetrisches Mischen, vollständiges Kopieren

$\{W = C_1, C_2, C_3\}$	$U = \overline{C_1C_2}$
$U = C_1, C_3$	$V = C_3$
$V = C_2, \phi$	$W = \overline{\overline{C_1C_2}C3}$
$W = \overline{C_1C_2}, C_3$	=> 6 Durchgänge
	Kopiervorgänge fur C_3?

13

Aufwand für N Initialläufe:

$\lceil \log_2 N \rceil$	Mischdurchgänge	$\lceil \log_m N \rceil$
$\lceil \log_2 N \rceil - 1$	Kopierdurchgänge	$\lceil \log_m N \rceil - 1$
1	Initiallauf-Durchgang	1

gesamt: $2 \cdot \lceil \log_m N \rceil$ **Durchgänge**

1 Durchg. = 1 x lesen ganz Σ

1 x schreiben

für m–Wege Mischen, m+1 Bänder

14

2. Symmetrisches Mischen, unvollständiges Kopieren

nach 1. Mischdurchgang:

$$\{W = C_1, C_2, C_3\}$$

$$U = C_1$$

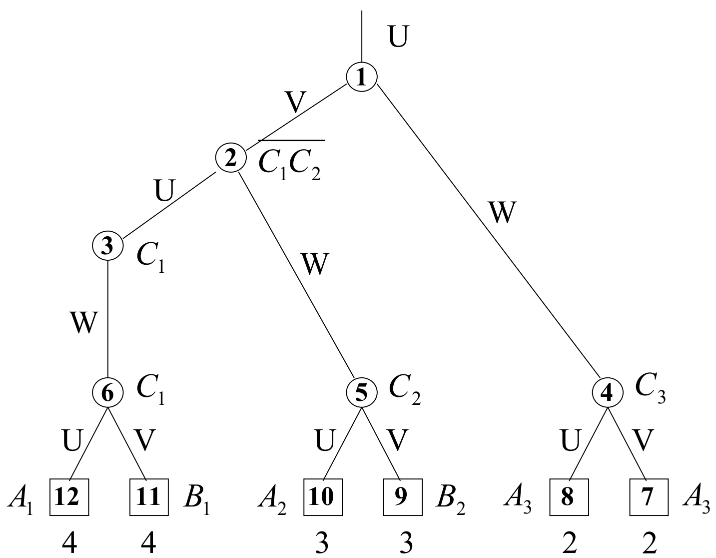
$$V = \overline{C_1 C_2}$$

$$U = \overline{\overline{C_1 C_2} C3}$$

=> 4 Durchgänge statt 6

15

Mischbaum nach Knuth



16

Symmetrisches Mischen, 2m Bänder

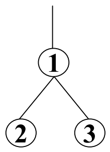
fast ohne Kopieren,

je m Bänder alternierend als Eingabe und Ausgabe

Baum für vorherige Initialläufe, 4 Bänder, symm. Mischen Kosten?

Mehrphasen Mischen (polyphase):

Sortieren ohne zu kopieren?



U	V	W
0	0	$\underline{1}$
1	$\underline{1}$	0
$\underline{2}$	0	1
0	2	3

letzter Mischdurchgang, Zustände vorher?

Rekursionsgleichung für Anzahl Läufe auf Band

$$0 \qquad C_i \qquad C_{i+1}$$

$$C_{i+1} \qquad (C_{i+1} + C_i) \qquad 0$$

$$= C_{i+2}$$

i.e Fibonacci

17

Problem: Länge der entstehenden Zwischenläufe:

z.B. 5-Wege, 65 Läufe

0	0	0	0	0	$\underline{1_{65}}$	65
1_1	1_5	1_9	1_{17}	$\underline{1_{33}}$	0	33
2_1	2_5	2_9	$\underline{2_{17}}$	0	1_1	34
4_1	4_5	$\underline{4_9}$	0	2_1	3_1	36
8_1	$\underline{8_5}$	0	4_1	6_1	7_1	40
16_1	0	8_1	12_1	14_1	15_1	65
						273

18

Ohne Rückspulung:

Bänder vorwärts schreiben, $\uparrow$

rückwärts lesen: $\downarrow$

0	0	1 $\uparrow$
1 $\downarrow$	1 $\downarrow$	0
2 $\downarrow\uparrow$	0	1 $\uparrow$
0	2 $\downarrow\uparrow$	3 $\uparrow\downarrow\uparrow$

19

Allg. Fall.:

m+1 Bänder

m Wege

m-1 Fib. Zahlen für nächste Phase

$$\begin{array}{cccccc} C_{k-m+1} & \cdots & C_{k-m+i} & \cdots & \underline{C_k} & 0 \\ (C_{k-m+1}+C_k)\cdots & & (C_{k-m+i}+C_k) & \cdots & 0 & C_k \end{array}$$

i.e. $C_{k-m+i} + C_k = C_{k+i}$ für i = 1 , ... , m-1

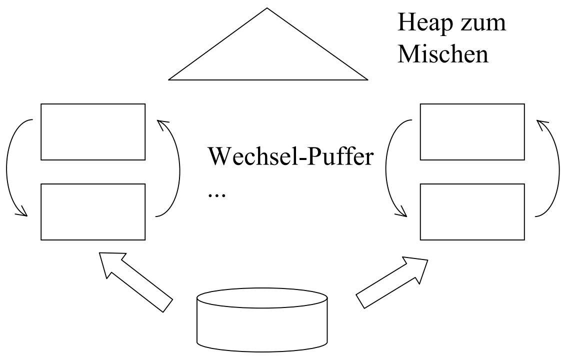
m-1 neue Fib.-Zahlen der Ordnung m-1

20

Kap. 7.3 Sortieren, Platten, Parallelität

Annahme: AS mit 10^n Bytes,
Seitengröße 10^3 Bytes
i.e. 10^{n-3} Seiten

Länge Initialläufe: $2 \cdot 10^n$ Bytes



Mischgrad:
½ $\cdot 10^{n-3} = 5000$
bei AS = 10MB

1 Mischlauf:
½ $\cdot 10^{n-3} \cdot 2 \cdot 10^n$
$= 10^{2n-3}$ Bytes

21

Workstations: n=7
d.h. 10^{11} B = 100 GB in1 Lauf sortiert
Problem: UNIX File-System

Mainframes: n=8
d.h. 10^{13} B = 10 TB in 1 Lauf sortiert
Fazit: Bei heutiger Rechner-Technologie ist Sortieren ein linearer Algorithmus!

CPU-Last: AS = 10^n B
1 Element sei z.B. 100 Bytes
$\Rightarrow$ Heap hat 10^{n-2} Knoten,
z.B. Bei n=7: 10^5 Knoten

22

Heap Höhe: $\log_2 10^5 \approx 17$ SENKE Aufrufe pro Element
mit je 4 Vergleichen:

string-Vergleich:

$\approx$ 500 Befehle
- extrahiere Attribut aus Tupel, Additionen, Multiplikationen
- Adressrechnung für M[k], M[2k], M[2k+1]

$\Rightarrow$ Pro El. $\approx 10^4$ Befehle?

Festplatte: 500 KB/s Nutzleistung ~ ~ 5000 El/s

CPU: 5000 El/s * 10^4 Befehle/El = 50 MIPS oder mehr

$\Rightarrow$ Potential für paralleles Sortieren!
1991: CPU bremst um Faktor 5!!
2000: CPU bremst bei RAID

23

Beispiel: Sortiere 100 MB auf SUN 4

Initialläufe:	1 x lesen	$\approx$ 200 s
	1 x schreiben	$\approx$ 200 s
		400 s

Beim Einlesen Bremsfaktor 5
wegen langsamer CPU: **1000 s**
Prozessor-Engpass!!

Mischlauf: 1 x lesen, 1 x schreiben — random Zugriff zur Platte

$\leq$ 50 Seiten/s, d.h. Plattenleistung:
$\leq \leq$ 50 KB/s bei 1KB/Seite
$\leq \leq$ 200 KB/s bei 4 KB/Seite

24

1 KB/Seite:	1 x lesen	~	2000 s
	1 x schreiben	~	2000 s
		für Mischen	4000 s
4 KB/Seite:	1 x lesen	~	500 s
	1 x schreiben	~	500 s
		für Mischen	1000 s

Mischgrad 5 bis 20, kleiner Heap, CPU bremst nicht!
Mischphase ist Plattenengpass!!

Gesamtzeit: 2000 s mit 4 KB/Seite
5000 s mit 1 KB/Seite
5000 s = 83 min

Hinweis: bei niedrigem Mischgrad, z.B. 10, verwende maximale E/A-Pufferung, mehrere Seiten pro Wechselpuffer:

25