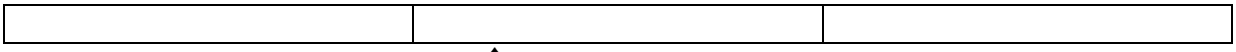
Informatik

Quick Sort:



≤ ↑ korrekte Position >

worst case:

- jeweiles „Randelement“ als Trennelement

⇒ n-1 Schritte

Aufwand pro Schritte (i –ter Schritt → n-1 Vergleiche)

Gesamtaufwand

$$\sum_{i=1}^{n-1} (n-i) = \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$$

(Bild 4.1)

best case:

- Trennelement in der Mitte

⇒ log2 (n) Schritte

Aufwand pro Schritt <= n

Gesamtaufwand = n * log2 (n) = O (n log n)

Das k-Kleinste-Element:

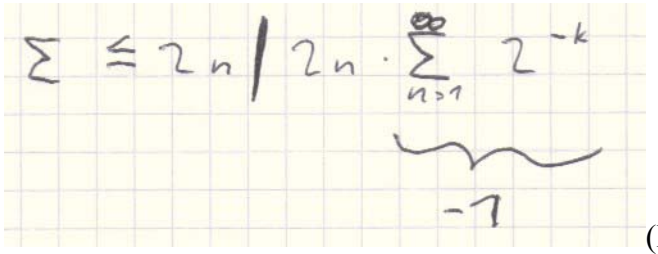
Trennalgorithmus (Quick Sort):

k < iX : suche links
k = iX : gefunden
k > iX : suche rechts

best case:

1. Schritt:	n	k –ter Schritt
2. Schritt:	n/2	n/(2^(k-1))
3. Schritt:	n/4	

Text: M. Wilke
Zeichnungen: M. Löschinger



(Bild 4.2)

Prinzipien zum sortieren:

- Vergleichen (+ Vertauschen), bestenfalls O (n * log n)
- Zählen (wie oft kommt jeder Schlüsselwert vor)
- Berechnung (der vermeintlich korrekten Position)

<u>Sortieren durch zählen (Bottom-up Radix sort):</u>

Schlüsselwerte (5, 3, 4, 5, 2, 1, 5, 0, 1, 2, 3)

0	I	0	
1	II	1	
2	II	3	
3	II	5	~~5~~ 6
4	I	7	~~7~~ 8
5	III	8	~~8~~ 9

0	
1	
2	
3	
4	
5	
6	3
7	4
8	5
9	5
10	

Gegebene Zahlen

	1	5	2	1
	1	6	3	2
	1	5	3	1
	2	4	2	3
	3	1	1	2
	2	5	3	2
Stelle	4	3	2	1

1. Schritt (sortieren der 4. Stelle)

	1	5	2	1
	1	5	3	1
	1	6	3	2
	3	1	1	2
	3	5	3	2
	2	4	2	3
Stelle	4	3	2	1

2. Schritt (sortieren nach der 3. Stelle)

3. Schritt (sortieren der 2. Stelle)

Text: M. Wilke
Zeichnungen: M. Löschinger

	3	1	1	2
	1	5	2	1
	2	4	2	3
	1	5	3	1
	1	6	3	2
	2	5	3	2
Stelle	4	3	2	1

	3	1	1	2
	2	4	2	3
	1	5	2	1
	1	5	3	1
	2	5	3	2
	1	6	3	2
Stelle	4	3	2	1

4. Schritt (letzte Stelle sortieren)

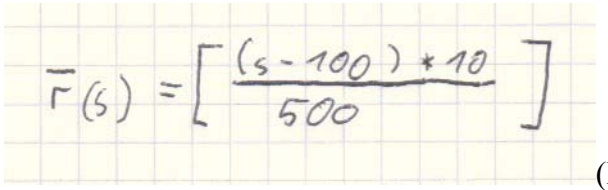
	1	5	2	1
	1	5	3	1
	1	6	3	2
	2	4	2	3
	2	5	3	2
	3	1	1	2
Stelle	4	3	2	1

Zeitkomplexität: d * n * 2
Mit d = Länge der Strings,
n = Anzahl der Strings

Quicksort: C * n * log n

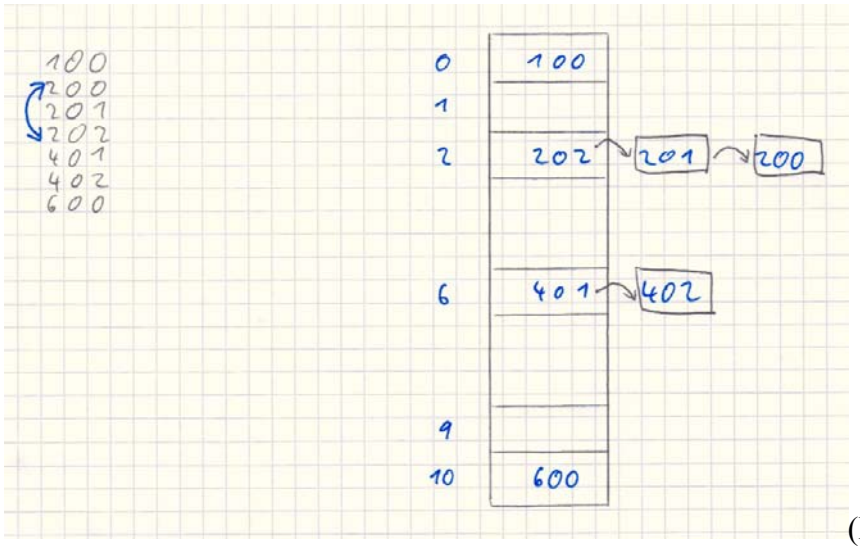
Wann ist Bottom-Up Rradix sort besser (als QS)?
d * ~~n~~ * ~~2~~ < (C / 2) * ~~n~~ * log n
Bei kurzen Strings

Interpolation sort:



(Bild 4.3)

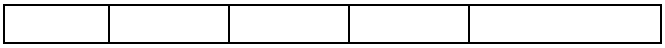
Schritt 1 Schritt 2
Sortieren falls overflow
Vorliegt (pro Bucket)

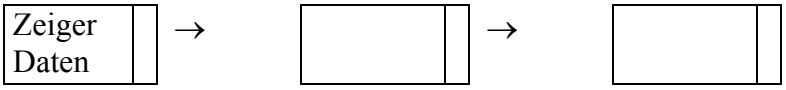


(Bild 4.4)

Text: M. Wilke
Zeichnungen: M. Löschinger

Lineare Listen:



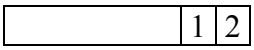


- Einfügen am Anfang *siehe Informatik23*
- Einfügen (nach einem bestimmten Element)
- Einfügen am Ende
- Löschen

Doppelt verkettete Listen (zirkular mit Dummy)

i. Einfügen nach
ii. Einfügen vor

1 Zeiger auf die vorheriges Element
2 Zeiger auf die nächstes Element



i) Neuen Knoten anlegen
Zeiger in neuen Knoten setzen
Zeiger auf neuen Knoten setzen

iii. Löschen:
zu löschenden Knoten aus der Verkettung nehmen
Speicher freigeben

Text: M. Wilke
Zeichnungen: M. Löschinger