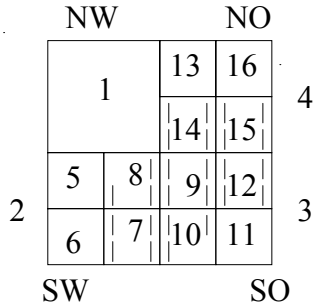
Kap.9 Quad – Trees

Kap. 9.1 Region Quad - Trees

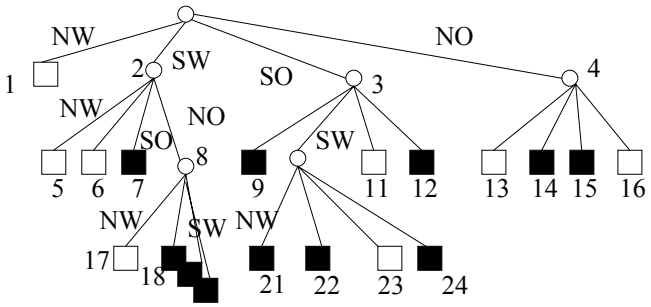
Darstellung von gerasterten Flächen mit Eigenschaften pro Flächenpunkt, z.B. Farbe, Höhe,... Pixel-Wert = {0,1} für s/w
Pixel = **P**icture **E**lement

Idee: Rekursive Flächenzerlegung in Teilflächen mit konstanten Pixelwerten



Zíel: Komprimierte Darstellung u. leichte Manipulation

1



Hinw.: Quad-Tree für Pixelbild eindeutig.
Baumhöhe ~ ld von Auflösung, z.B. bei Bildschirm 1000 x 1000 ≈ 10

Überlegung: geeignet für Darstellung von Schrift auf Bildschirm?

Algorithmen:

Farbbestimmung: i.a. Pixeleigenschaft

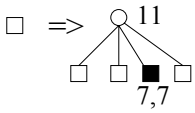
Ist Pixel [3,5] schwarz oder weiß?

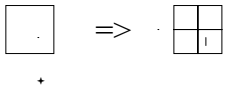
1. [3,5] liegt in NO = (4)
2. [3,5] in (4) in SW = ■

⇒ Pixel [3,5] = schwarz

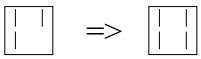
2

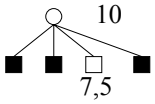
Update: Pixel [7,7] := 1





Pixel [7,5] := 1





=> ■ 10

Fortpflanzung
Richtung Wurzel?

Wichtig: lokale Bildänderung ~
lokale Trafos in Quad-Tree
längs 1 Pfad

3

Bild zeigen: paint (QT)

if QT = schw. Knoten **then** alle Pixel schwarz

else **if** QT = weißer Knoten **then** alle Pixel weiß

else ⌈ {Wurzelknoten von QT ist grau}

paint (QT. NW); paint (QT. SW);

⌊ paint (QT. SO); paint (QT. NO)

4

Randbestimmung: Finde größeren westlichen Nachbarn von x, z.B. x = 21 :

Pfad zu ■ 21 von Wurzel aus:

SO
SW
NW
■ 21
Keller

Alg: Pfad zu 21 kellern, zurück zu letzter Ost-Abzweig., nach Westen, dann immer nach Osten

SW
SO ■ 7
~~NO~~

■ 7 größer als ■ 21 weil Pfad kürzer, aber gleich gefärbt, Zwischenkante, i.R. westliche Grenze von 21 gehört nicht zum Rand

westlicher Nachbar von 18:

SW SW
NO NW □ 5
SW SO
■ 18 □ 5 > ■ 18, unterschiedlich gefärbt

=> westl.Kante von ■ 18 im Rand

5

Falls gesuchter Nachbar echt kleiner, keine Entscheidung, entscheide vom kleineren Nachbarn aus gesehen.

Bei gleichgroßen Nachbarn: Zwischenkante erscheint zweimal im Rand, nicht schädlich!

Alg. für Rand:

∀ Blätter X
bestimme alle 4 Nachbarn Y
if Y ≥ X u. Farbe verschieden
then Kante von X zu Y gehört zum Rand zwischen weißem u. schwarzem Gebiet.

Alg. für Rand und Gebiet mit Farbe f:

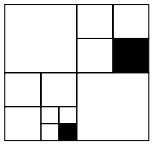
∀ Blätter X mit Farbe f
:
wie oben

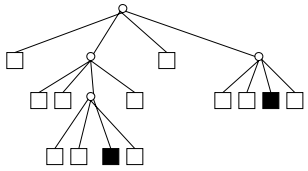
6

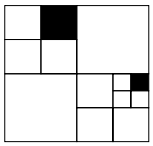
Weitere Algorithmen

Bild Drehungen um 90 °

~ Kante +1 mod 4
i.e. Umnummerierung der Kanten

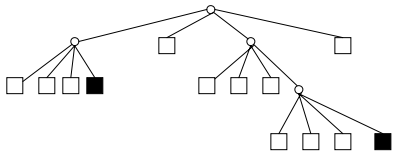






Dreh-Alg:
1. Verschieben
2. Drehen, i.e. rek. Aufruf f. Unterbaum

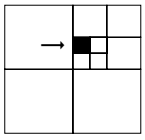
nicht konstruieren sondern Kantendrehung mitrechnen

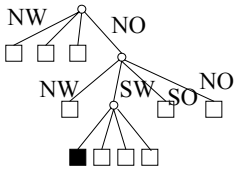


Hinw: geht, weil Blätter gegen Drehung invariant sind.

7

Raster-Display: zeilenweise





Auffinden von östlichen Nachbarn:
1. Zurück zu westlicher Kante
2. Einmal östlich gehen : NW → NO
   SW → SO
3. nur westliche Zweige verfolgen:
   ob SW oder NW aus
   Zeilenzahl berechnen

Auffinden von südlichen Nachbarn:
0. Knoten am linken Rand merken
1. Zurück zu nördlicher Kante
2. einmal südlich gehen
3. allen NW-Zweigen folgen: i.e. südl. Nachbar am weitesten westlich

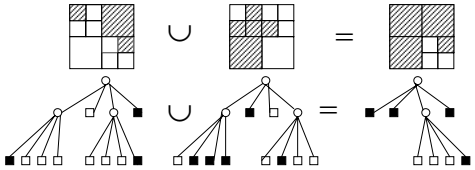
8

Zoom: auf einen Quadranten:
paint (QT.NW), ... , paint (QT.NO)

Mengen Operationen: $\cup$, $\cap$
dunkel dominiert bei $\cup$,
entsprechende Knoten rekursiv vergleichen:

$s \cup x = s$ x beliebig
$w \cup x = x$
$g \cup s = s$
$g \cup w = g$
$g \cup g =$ { s : wenn nach $\cup$ alle Söhne s; g sonst }

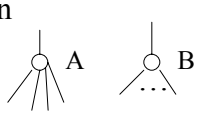
Beispiel



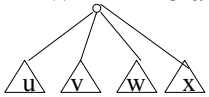
9

Verallg. auf beliebige Farbmischungen?
blau $\cup$ gelb = grün
⋮

Alg. für Union:



Union (A,B) :
if black (A) **or** black (B) **then** black node
else if white (A) **then** B
else if white (B) **then** A
else ⌈ U := Union (NW (A), NW (B));
V := Union (SW (A), SW (B));
W := Union (SO (A), SO (B));
X := Union (NO (A), NO (B));
if U = V = W = X = black **then** black node
else



⌊

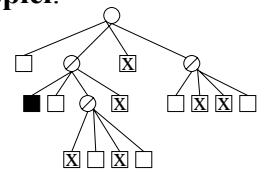
Alg. für $\cap$ und Differenz analoge rekursive Struktur.

Übung: Flächenberechnung!

10

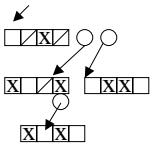
Speicherdarstellung für Quad-Trees

Beispiel:



weiß ~ 00
schwarz ~ 11
grau ~ 01
illegal 10

16 Zeiger



4 Zeiger

i.e. Reduktion auf ¼

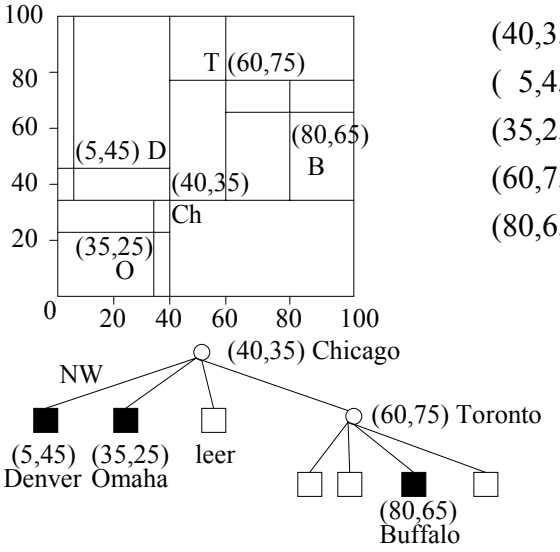
$$\text{allg: Anz.Zeiger} = \frac{\text{Anz.Knoten} - 1}{4}$$

$$\text{Farbinfo} = \frac{\text{Anz.Knoten} - 1}{4} \quad \text{Bytes}$$

11

Kap. 9.2 Point Quad-Tree:

Suche mit Koordinaten,
Partionierungspunkt mit Koord.



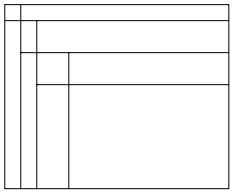
(40,35) Chicago
(5,45) Denver
(35,25) Omaha
(60,75) Toronto
(80,65) Buffalo

12

Suche mit vorgegebenen Ortskoordinaten.
Intervall-Suche mit vorgeg. Längen- u. Breitenintervallen

Hinweis: Struktur des Baumes u. Qualität (Balanzierung) hängt von Reihenfolge der Einfügung ab.
Heuristik: Wähle Partitionierungspunkt so, daß jeder Quadrant möglichst gleichviel Punkte enthält.

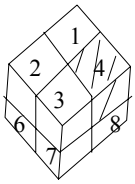
Problem:

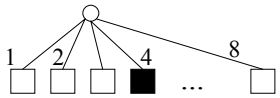


13

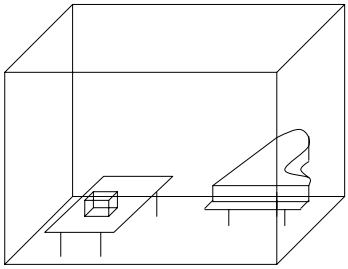
Kap. 9.3 Oct-Trees

für 3-dim. Daten : unterteile Raum in Würfel





z.B. Raum mit Objekten:

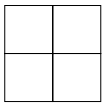


Robotik:
Planung

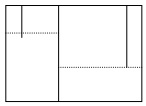
Aufgaben: Drehungen, Begrenzungsflächen, Vereinigung etc. analog zu 2-dim. Raum

14

Kap. 9.4 k-d-Bäume

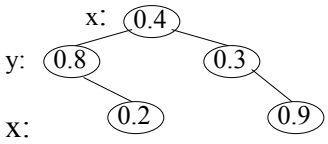


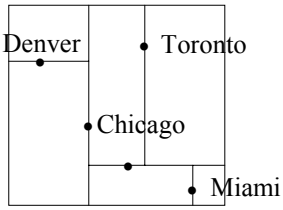
Quad-Tree:
Partitionierung
bzgl. aller Dim.



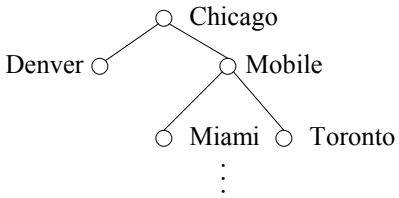
k-d-Tree:
Partitionierung
bzgl.1 Dimension,
zyklisch alle Dim.

nur Blätter sind
informations-
tragend



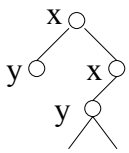


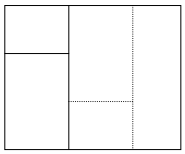
alterierende Dimension



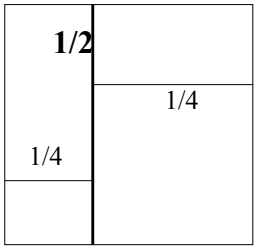
Adaptiver k-d-Baum

Zu spaltende Dimension an jedem Knoten frei wählbar, Angabe im Knoten





Balanzierte k-d-Bäume: statisch einfach, dynamisch schwierig



Spalte so, daß Anzahl der Elemente pro Bereich halbiert wird, einfacher als bei Point-Quad-Tree!

Dynamische Balanzierung bei insert/delete Folgen?

17

Kap. 9.5 dd-Bäume

Mehlhorn, Vol.3, S.26 ff.

Suche in d-dimensionalen Räumen

Def: U_i : geordnete Menge, $1 \leq i \leq d$

$U = U_1 \times U_2 \times \ldots \times U_d$

Punkt $x = (x_1, \ldots, x_d) \in U$

Record, Tupel

x_i : Koordinate, Attribut

Standardfall bei DBSen

Gegeben: $S \subseteq U$; S = Menge von Punkten, Relation, Tabelle, Datenbank

Anfragen: $R \in \Gamma \subseteq 2^U$

Problem: Berechne $R \cap S$ bzw. zähle $|R \cap S|$

R heißt "region-query"

"Bereichs-Anfrage"

18

Spezialfälle für R:

a) **orthogonale Range-Queries**

Intervall-Anfrage : $R = \{x : l \leq x \leq h\}$
im 1.dim. Fall

$\Gamma_{OR} = \{R : R = [l_1:h_1]$ **x** $[l_2:h_2]$ **x ... x** $[l_d:h_d]\}$
mit $l_i, h_i \in U_i$ und $l_i \leq h_i$

19

b) **partial Match-Queries**

einige Attribute sind Konstante,
andere unspezifiziert:
$l_i = h_i$ oder $[l_i : h_i] = U_i$
$R \equiv a_{i_1} = c_1 \wedge a_{i_2} = c_2 \wedge ... \wedge a_{i_k} = c_k$

Siehe auch: Vektorraum - Modell
beim Information - Retrival

$\Gamma_{PM} = \{ R : R = [l_i:h_i]$ **x ... x** $[l_d :h_d]\}$
mit $l_i, h_i \in U_i$ und
$l_i = h_i$ oder $l_i = -\infty$ und $h_i = +\infty$

c) **exact Match-Queries**

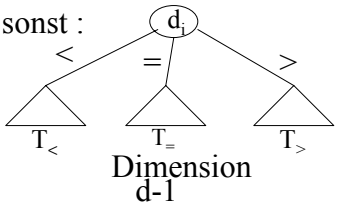
$\Gamma_{EM} = \{ R : R = \{x\}$ und $x \in U\}$
d.h. alle Attr. als Konstante spezifiziert
Antwort nur **ja** oder **nein**
DB-Anwendungen : Γ_{OR} oder Γ_{PM}
mit vorher bekannten suchbaren Attributen.

20

Def: dd-Baum: d = Dimensionalität, n = DB-Größe

- d = n = 1 : x

- sonst :



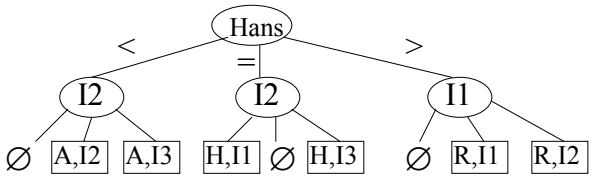
$S_< = \{x : x_i < d_i\}$ $S_= = \{x : x_i = d_i\}$ $S_> = \{x : x_i > d_i\}$

i.e. Partionierung von S bzgl. d_i

Beispiel: Angelika, Hans, Richard
I1, I2, I3

} 2 Dimensionen mit möglichen Werten

S = { (A,I2), (A,I3), (H, I1), (H,I3), (R,I1), (R,I2)}



andere Wahl der Spaltwerte ?

Def: Ideale dd-Bäume:

Wähle di so, daß $T_<$ und $T_>$ höchstens

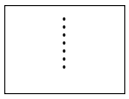
½ n Elemente haben,

keine Garantie für $T_=$, aber

$T_=$ hat kleinere Dimension, keine weitere

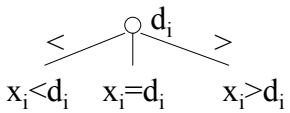
Partionierung bzgl. i-ter Dim.

Beispiel:



23

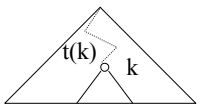
Exact Match Anfrage:



$x_1, x_2, \ldots, x_d$

verfolge Kante entsprechend:

Def:



t(k) : Tiefe von Knoten k

= 1 + Pfadlänge zu k

st(k) : Anzahl von

< - bzw >- Kanten zu k

Lemma: Sei T idealer dd-Baum für Menge S mit |S| = n

a) st (k) ≤ *ld* n für alle k

b) t (k) ≤ d + *ld* n für alle k

24

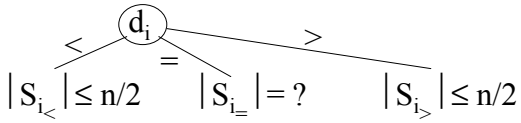
Konstruktion von dd-Bäumen

in Zeit Q (n· (d + log n)) für S mit $|S| = n$

für Spaltung bzgl. i-ter Dimension:

$S_i = \{x_i : (x_1, x_2, ..., x_i, ..., x_d) \in S\}$
als Multiset

bestimme mittleres Element d_i von S_i in Zeit $|S_i| = n$



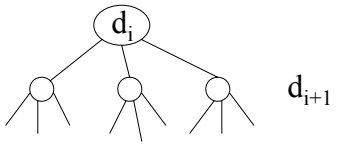
jetzt rekursive Fortsetzung: spalte
bzgl. Dim. i+1, i+2, ...

Kosten: Auf jeder Baumebene
Median-Bestimmung bzgl. ganz S in O ($|S|$);
Baumhöhe ≤ d + log n

Gesamtkosten: O (n· (d + log n))

25

Partial Match Anfragen:



- falls i-tes Attribut in Anfrage
spezifiziert ist, verfolge 1 Pfad
- sonst verfolge 3 Pfade

=> depth-first Suche durch Baum
bis zu Blättern

Detaillierte Kostenanalyse: für Suchaufwand,
siehe Mehlhorn, Bd 3, S. 30ff

$Q(n^{\varepsilon})$ für $0 < \varepsilon < 1$

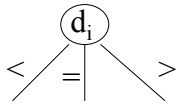
abhängig von Detaillierung der Anfrage
$\varepsilon = 1$ falls völlig unspezifiziert

26

Exact Match Anfragen:

genau 1 Pfad durch Baum

Range Queries:



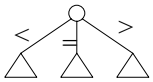
$[l_i : k_i]$

$d_i < l_i$: verfolge > - Kante
$d_i = l_i$: verfolge = - und > - Kante
$l_i < d_i < h_i$: " 3 Kanten
$d_i = h_i$: " < - und = - Kante
$d_i > h_i$: " < - Kante

27

Kap. 9.6 Median-Bestimmung

siehe Kap. 8.5 dd-Bäume



Satz: P(n) sei ein Algorithmus für Problem P der Größe n
R(n) sei ein Alg. zur Reduktion von P(n) auf ein kleineres Problem $\alpha \cdot n$ mit $\alpha < 1$
P(n) habe die Struktur $P(n) \equiv P(R(n))$

P(n) habe Kostenfunktion p(n) bzl. Zeit, Speicher
R(n) " " r (n)

Falls R lineare Komplexität hat, dann
hat auch P lineare Komplexität

(Spezialfall des Master-Theorems der Komplexitätstheorie)

28

Bew: $r(n) \leq c \cdot n$

$p(n) = r(n) + p(\alpha \cdot n)$

$= r(n) + r(\alpha \cdot n) + p(\alpha^2 n)$

$= r(n) + r(\alpha \cdot n) + \ldots + r(\alpha^i \cdot n) + n \cdot c_o$

$\leq c \cdot n + c\alpha n + \ldots + c\alpha^i n + n \cdot c_o$

$= c \cdot n \left(1 + \alpha + \ldots + \alpha^i + \frac{c_o}{c}\right)$

Hinweis: i hängt von n ab, kann beliebig groß werden.

$\leq n \cdot c \left[\frac{1}{1-\alpha} + \frac{c_o}{c}\right]$

$= n \cdot c_p$

$\Rightarrow p(n) \leq n \cdot c_p$ ✓

29

Variante: Zerlegung in mehrere Teilprobleme:

$M = \{M_1, M_2, \ldots, M_k\}$; M_i, M_j disjunkt,

d.h. unabhängig voneinander lösbar.

Größen:

$m \quad m_1 \; m_2 \ldots m_k$ mit:

$\sum_{i=1}^{k} m_i \leq \alpha m$ und $\alpha < 1$

30

Zerlegung koste $c \cdot m$,
nächste kostet $\leq c \cdot \alpha\, m$ etc ,
=> alle Zerlegungen kosten

$\leq c \cdot m\,(1 + \alpha + \alpha^2 + \ldots + \alpha^i)$

wobei i abhängig von m, aber

$\leq c \cdot m \sum_{i=0}^{\infty} \alpha^i = cm \frac{1}{1-\alpha}$

$= m \frac{c}{1-\alpha}$

abschließende Bearbeitung pro Element von M : $m \cdot d$

=> Gesamtkosten: $m \cdot \left[\frac{c}{1-\alpha} + d \right]$

$= O(m)$ => linearer Alg.

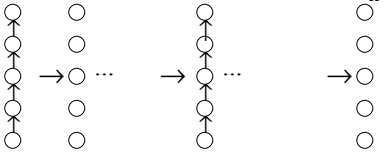
Hinw: Reihenfolge von Zerlegungen u. Bearbeitung von Teilproblemen unwichtig.

31

Median-Alg : Auswahl (M, l) : finde l-kleinstes Element von M

Menge $M = \{ e_1, e_2, \ldots e_m \}$

1. Sortiere k aufeinanderfolgende Elemente,
   k ungerade $\wedge$ $k > 1$
2. M_k sei Menge der mittleren Elemente,
3. bestimme mittleres El. μ_k von M_k



Problemgröße $\lceil \frac{m}{k} \rceil$

4. Zerlege M in 3 Teilmengen :

$M^k_< := \{ e_i : e_i \in M \wedge e_i < \mu_k \}$

$M^k_> := \{ ei : ei \in M \wedge ei > \mu_k \}$

$M^k_= := \{ \mu_k \}$

32

5. Sei $|M^k_<| = j$

if $l \leq j$ **then** bestimme l-kleinstes Element

von $M^k_<$: Auswahl ($M^k_>$, l)

else if $l = j+1$ **then** μ_k

else bestimme $l - (j + 1)$ – kleinstes Element von $M^k_>$:

Auswahl ($M^k_>$, l-(j+1))

Ursprünglicher Aufruf für Median-Bestimmung:

Auswahl $\left(M, \left\lceil \frac{m}{2} \right\rceil \right)$

bzw: $\left\lfloor \frac{m}{2} \right\rfloor$

auf 2 Subprobleme M^k bzw. $M^k_<$, $M^k_>$

33

Analyse:

Schritte 1., 2., 4. sind Reduktionsschritte
auf Suchprobleme M_k bzw. $M^k_<$, $M^k_>$

Schritt 3: Größe von Problem

$|M_k| = \left\lceil \frac{m}{k} \right\rceil$

Schritt 5: Größe von Problem $M^k_<$:

$|M^k_<| \geq \frac{m}{k} \cdot \frac{1}{2} \cdot \frac{k+1}{2} = \frac{m}{4} \cdot \frac{k+1}{k}$

$|M^k_>| \geq$ ...

bis auf $\lceil \rceil$ bzw. $\lfloor \rfloor$, i.e. $+ k+1$

$|M^k_<| = m - |M^k_>| \leq m - \frac{m}{4} \cdot \frac{k+1}{k} = \frac{m}{4k} (3k-1)$

34

Gesamtgröße der Subprobleme

$$\begin{matrix} & < & & > \\ |M^k| + |M^k| & \text{bzw.} & |M^k| + |M^k| & \end{matrix}$$

$$\leq \left\lceil \frac{m}{k} \right\rceil + \frac{m}{4k}(3k-1) \leq \frac{m}{k} + 1 + \frac{m}{4k}(3k-1)$$

$$= m\left(\frac{1}{k} + \frac{1}{m} + \frac{3k-1}{4k}\right) = m \underbrace{\left(\frac{3}{4} \cdot \frac{k+1}{k} + \frac{1}{m}\right)}_{c(k)}$$

35

$$c(3) = 1 + \frac{1}{m} > 1$$

$$c(5) = \frac{3}{4} \cdot \frac{6}{5} + \frac{1}{m} = \frac{18}{20} + \frac{1}{m} = \frac{9}{10} + \frac{1}{m}$$

$$< 1 \text{ für } m > 10$$

$$c(7) = \frac{3}{4} \cdot \frac{8}{7} + \frac{1}{m} = \frac{6}{7} + \frac{1}{m} < 1 \text{ für } m > 7$$

=> für große k : c(k) ➨ ¾ ,
d.h. Kosten für Lösung von M^k
werden unwesentlich.

Warum nicht für Quicksort?

0 (nlogn) Garantie für Quicksort?

36