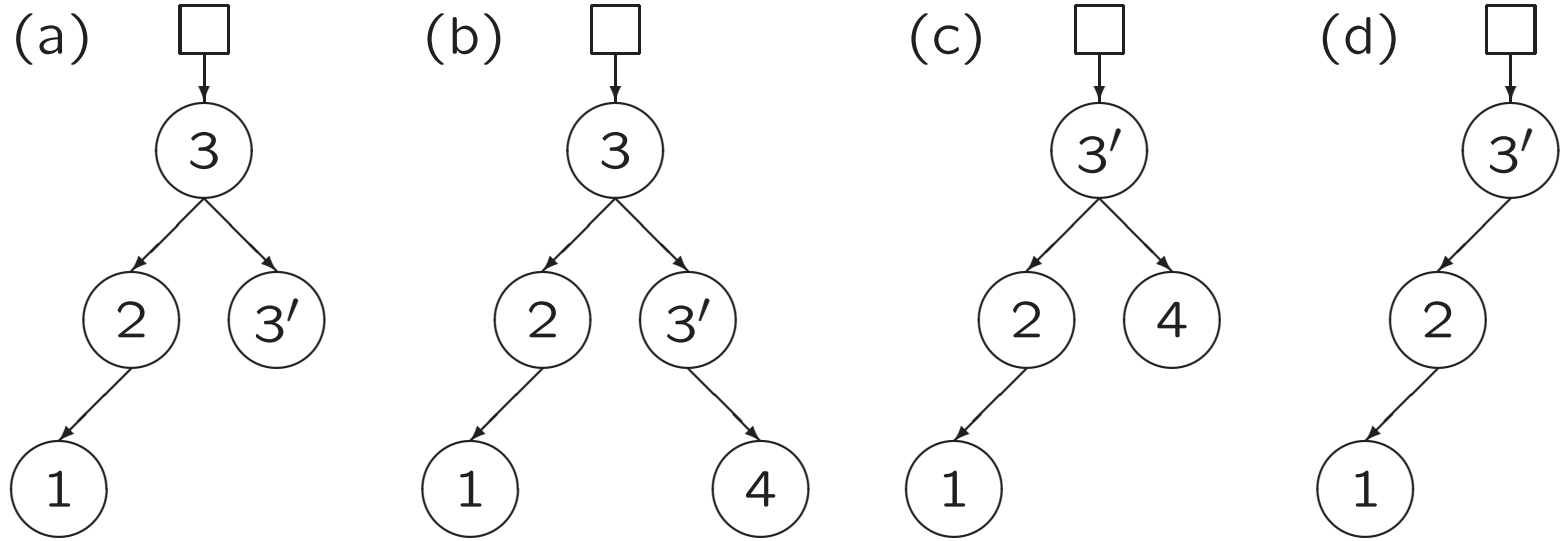
Binärer Baum



Algorithmische Kosten

Kostenabschätzungen

- grobe Abschätzung von Laufzeit und Speicherverbrauch
- konstante Faktoren vernachlässigt

  $\Rightarrow$ geschätzte Kosten unabhängig von Hardware, Compiler
- Anzahl der Elemente in Datenstruktur berücksichtigt
- Abschätzung maximaler und durchschnittlicher Kosten
- Kosten typisch für Algorithmen bzw. Datenstrukturen

Beispiele für Kosten (Laufzeit)

Operation	Schnitt	Max.
Einfügen in Liste:	$O(1)$	$O(1)$
Suche in Liste:	$O(n)$	$O(n)$
Löschen aus Liste:	$O(n)$	$O(n)$
Einfügen in Baum:	$O(\log(n))$	$O(n)$
Suche in Baum:	$O(\log(n))$	$O(n)$
Löschen aus Baum:	$O(\log(n))$	$O(n)$

$O(1)$ = konstant

$O(\log(n))$ = logarithmisch

$O(n)$ = linear

$O(n^2)$ = quadratisch

$O(n^k)$ = polynomial

$O(2^n)$ = exponentiell

Beispiele für Kosten (Speicher)

- Kosten für Liste und Baum: $O(n)$
- kosten für Operationen auf Liste und Baum:
  - iterativ: $O(1)$
  - rekursiv: gleich den Kosten für Laufzeit
  - Relevanz gering (höhere Kosten entscheidend)
- Ausgewogenheit zwischen Laufzeit und Speicher wichtig

Beispiel: Sortiertes Ausgeben

Verkettete Liste: (Sortieren notwendig)

- Liste aufbauen: $n \cdot O(1) = O(n)$
- Sortieren: $O(n^2)$ für Bubble Sort
- Ausgeben: $O(n)$
- insgesamt: $\max(O(n), O(n^2), O(n)) = O(n^2)$

Binärer Baum: (schon bei Einfügen sortiert)

- Baum aufbauen (∅): $n \cdot O(\log(n)) = O(n \cdot \log(n))$
  Baum aufbauen (maximal): $n \cdot O(n) = O(n^2)$
- Ausgeben: $O(n)$
- insgesamt (∅): $\max(O(n \cdot \log(n)), O(n)) = O(n \cdot \log(n))$
  insgesamt (maximal): $\max(O(n^2), O(n)) = O(n^2)$