Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

Übung Datenbanksysteme II

Index-strukturen

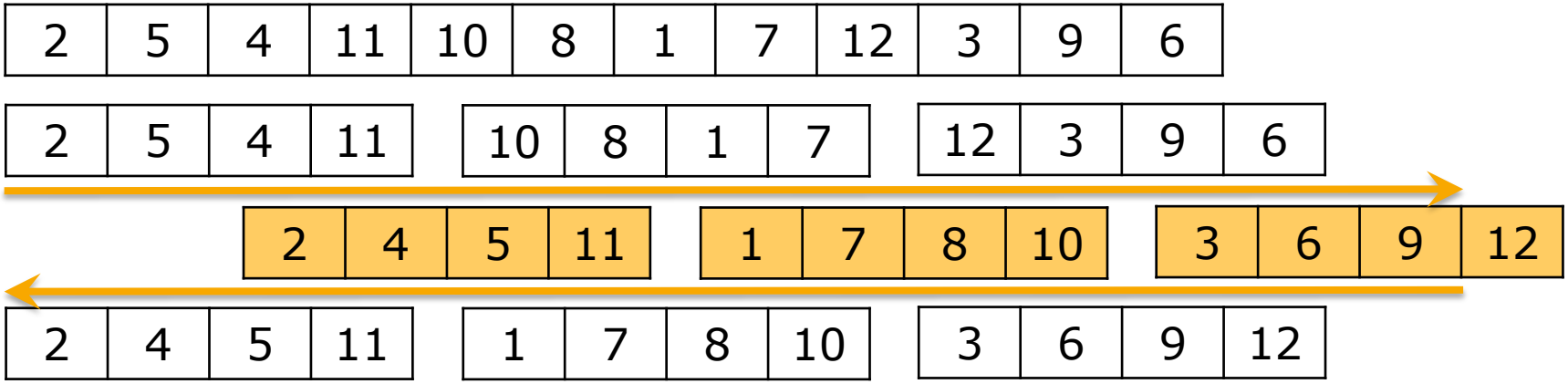
Thorsten Papenbrock

Rückblick:
Hausaufgabe 1

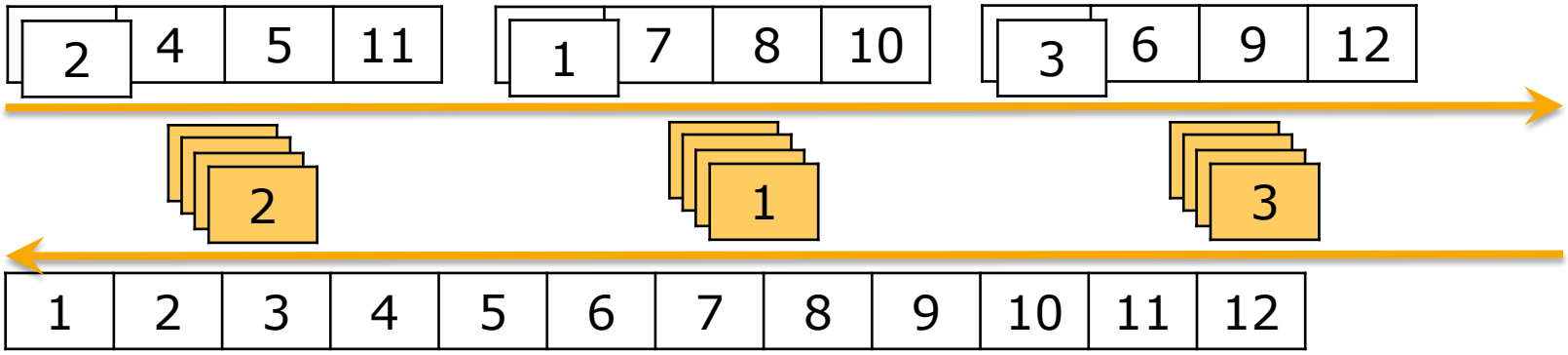
- Submit:
  - Bitte beachten:
    Bei JEDER Submission BEIDE Autoren angeben!
  - Notifications sind raus; Korrekturen einsehbar; Anmerkungen jetzt natürlich in Freitextfeldern statt direkt auf Zetteln

Rückblick: TPMMS

- Phase 1:



- Phase 2:



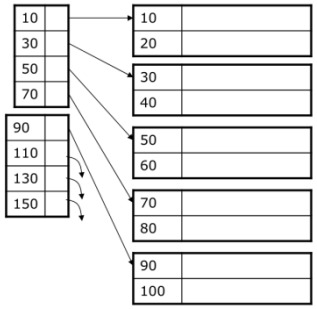
Rückblick: TPMMS

- Phase 1:
  - n Blöcke hat die gesamte Relation
  - m Blöcke passen gleichzeitig in den RAM
  - ➢ I/O Zeit Phase 1 = $\left\lceil \frac{n}{m} \right\rceil \cdot$ m Blöcke *sequentiell* lesen +
    $\left\lceil \frac{n}{m} \right\rceil \cdot$ m Blöcke *sequentiell* schreiben

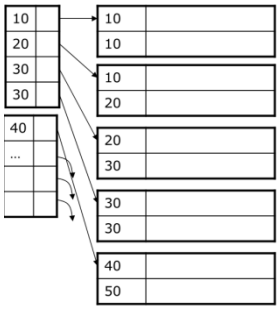
- Phase 2:
  - n Blöcke hat die gesamte Relation
  - m Blöcke passen gleichzeitig in den RAM
  - ➢ I/O Zeit Phase 2 = n Blöcke *selektiv* lesen +
    n Blöcke *selektiv* schreiben

Quiz: Zuordnung
Indexstrukturen

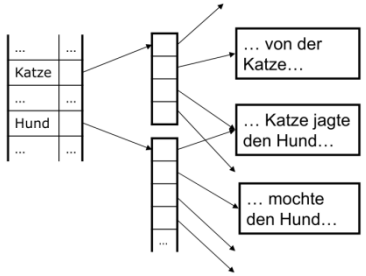
b.



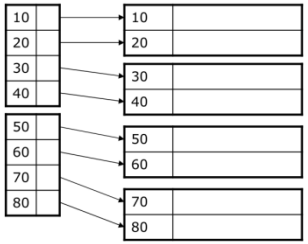
d.



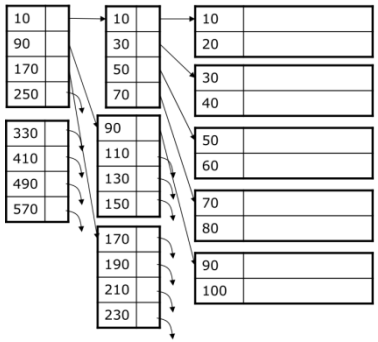
f.



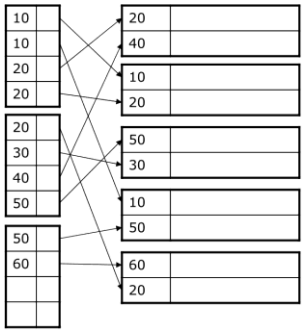
a.



c.



e.



a. dichtbesetzter Index
b. dünnbesetzter Index
c. mehrstufiger Index
d. nicht-eindeutige Attribute
e. Sekundärindex
f. invertierter Index

Dicht- und dünnbesetzte Indexe

- Es ist für jedes Datenfile möglich, zwei separate *dünnbesetzte* Level-1-Indexe auf unterschiedlichen Attributen anzulegen.

  Falsch Bei zwei Schlüsseln kann nur nach einem Schlüssel sortiert werden. Beide dünnbesetzte Level-1-Indexe benötigten daher eine Sortierung nach ihrem Schlüssel-Attribut. Das ist gleichzeitig nicht möglich!

- Es ist für jedes Datenfile möglich, zwei separate *dichtbesetzte* Level-1-Indexe auf unterschiedlichen Attributen anzulegen.

  Wahr Dichtbesetzte Level-1-Indexe benötigen keine Sortierung. Es können daher mehrere dichtbesetzte Indexe auf unterschiedlichen Schlüsseln angelegt werden!

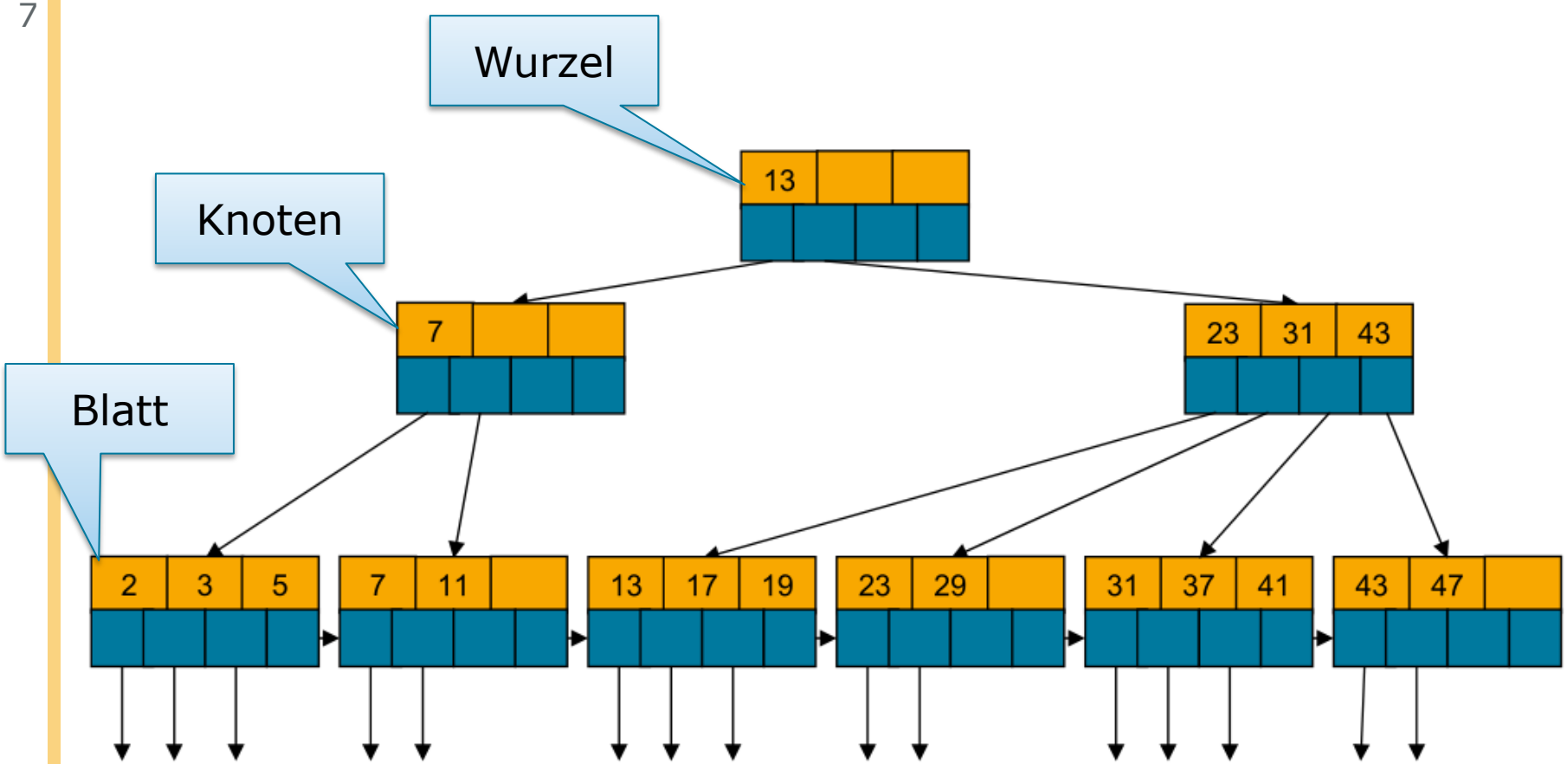
- Es ist für jedes Datenfile möglich, einen *dünnbesetzten* Level-1-Index mit einem *dichtbesetzten* Level-2-Index anzulegen. Beide Indexe sind sinnvoll.

  Falsch Ein dichtbesetzter Level-2-Index auf einen Level-1-Index ist sinnlos, da er alle Werte noch einmal indexiert, dabei zusätzlich Speicher belegt und keinen Mehrwert schafft.

- Es ist für jedes Datenfile möglich, einen *dichtbesetzten* Level-1-Index mit einem *dünnbesetzten* Level-2-Index anzulegen. Beide Indexe sind sinnvoll.

  Wahr Ein dünnbesetzter Level-2-Index auf einem dichtbesetzten Level-1-Index ist sinnvoll, da der dünnbesetzte Index nur die ersten Elemente der Blöcke indexiert und so letztendlich Speicher und Zugriffszeit spart.

B+-Baume: Aufbau



B⁺-Baume

- B⁺-Bäume können Overflow-Buckets benötigen.

  Falsch Statt Overflow-Buckets zu generieren nutzen B⁺-Baume *split*-Operationen um den Index beim Überlauf eines Knotens oder Blattes zu erweitern.

- Ein Block eines B⁺-Baums speichert n Pointer und n+1 Suchschlüssel.

  Falsch Ein Block (= Knoten bzw. Blatt) speichert *bis zu* **n Suchschlüssel** und *bis zu* **n+1 Pointer**.

- Der durch Löschen eines Schlüssels entstehende B⁺-Baum ist nicht eindeutig bestimmt.

  Wahr Beim Geschwister-Klauen und beim Merge ist nicht eindeutig bestimmt, mit welchem anderen Knoten die Operation durchgeführt wird!

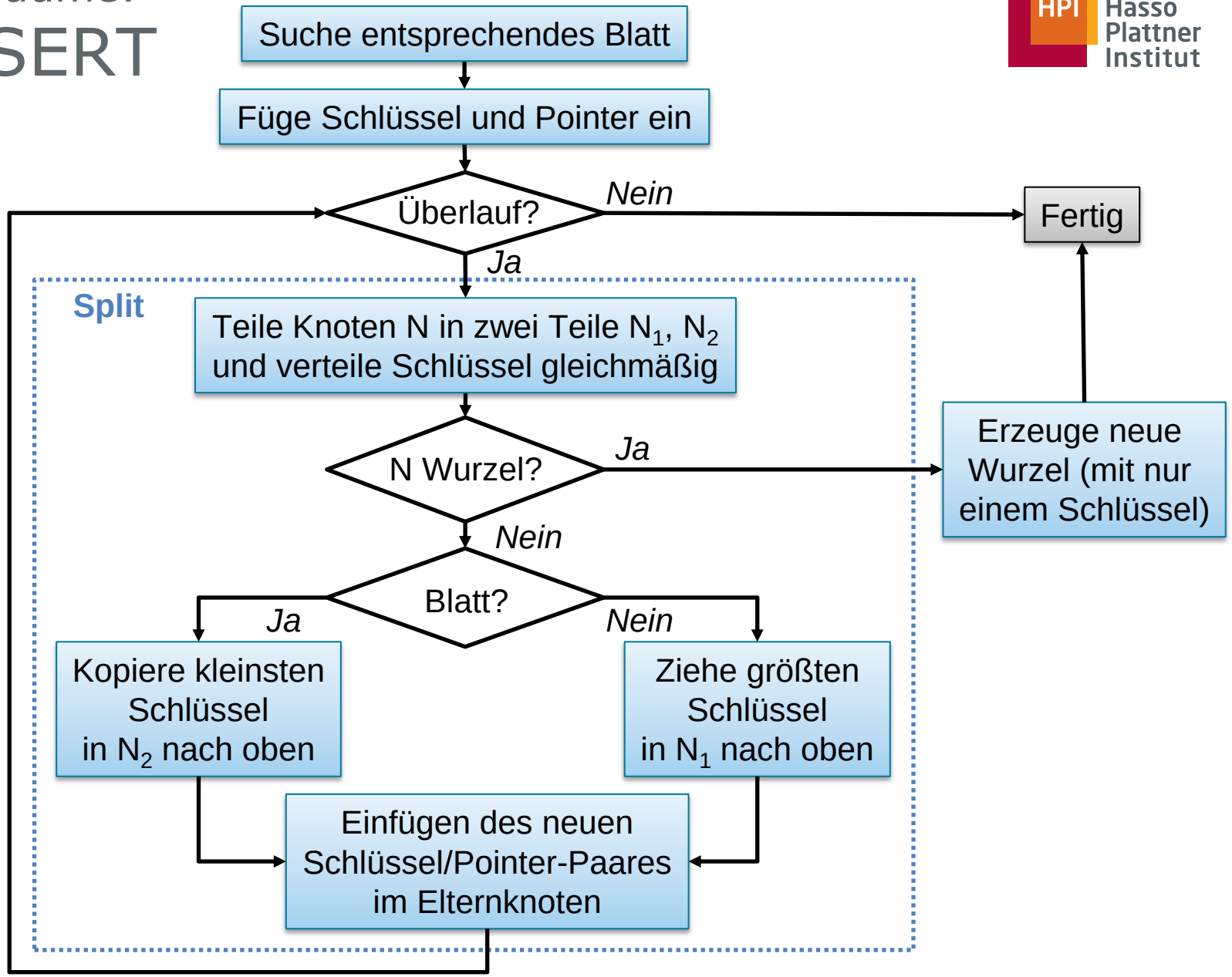
- B⁺-Bäume können nur als Primärindexe verwendet werden.

  Falsch B⁺-Baume können verschiedene Index-Rollen übernehmen. Dabei können sie auch eine Sortierung des indexierten Attributes nutzen, benötigen diese aber nicht zwangsläufig. B ⁺-Baume sind daher nicht nur als Primärindex einsetzbar.

- B⁺-Bäume können auf eindeutigen und nicht-eindeutigen Attributen angelegt werden.

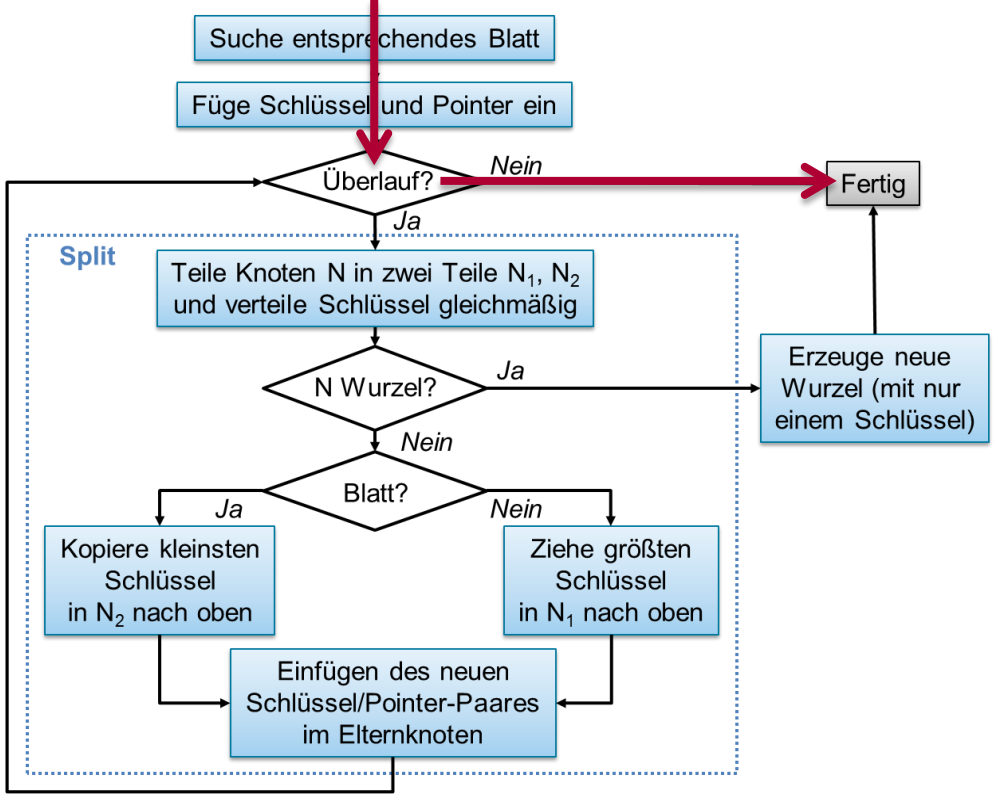
  Wahr Ein B+-Baum ist ein mehrstufiger Index, der sowohl auf eindeutigen als auch auf nicht-eindeutigen Attributen aufgebaut werden kann. Bei doppelten Attributwerten müssen wir uns mit einem zusätzlichen Level unter den Blättern behelfen, da doppelte Schlüsselwerte nicht erlaubt sind!

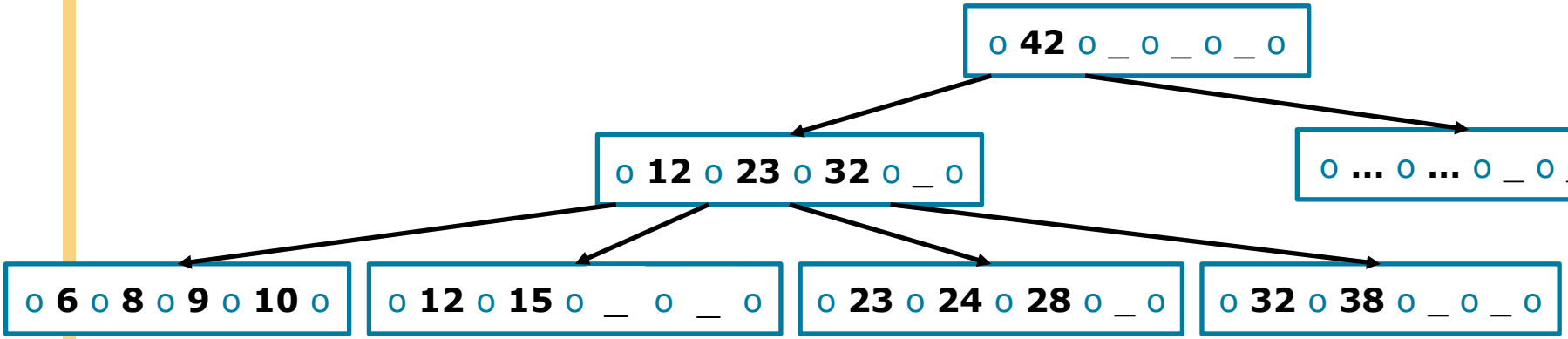
B⁺-Bäume: INSERT



B+-Bäume: INSERT

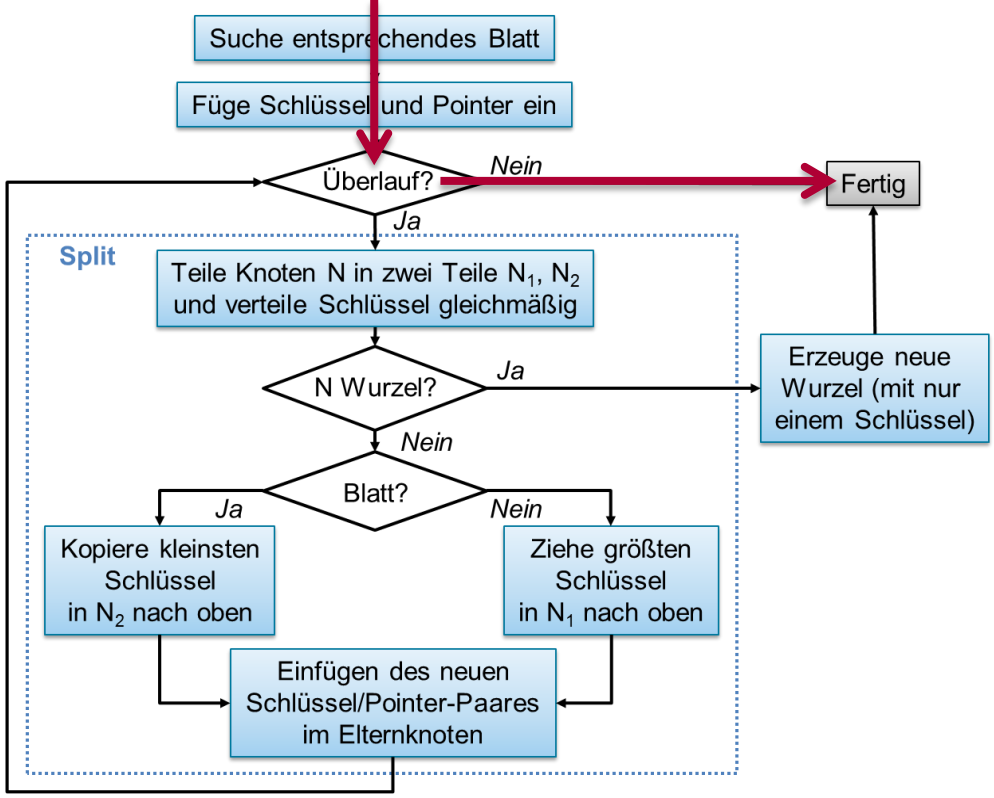
- INSERT(17)

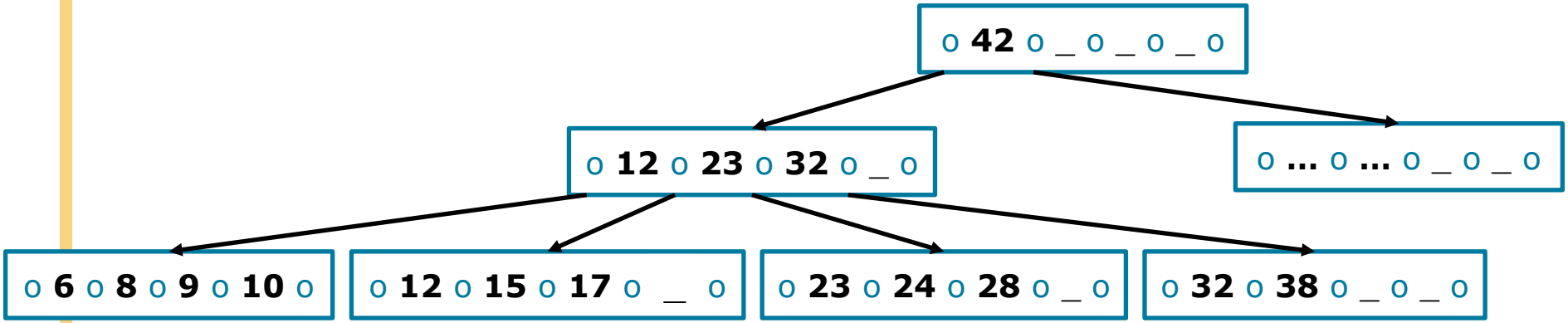




B+-Bäume: INSERT

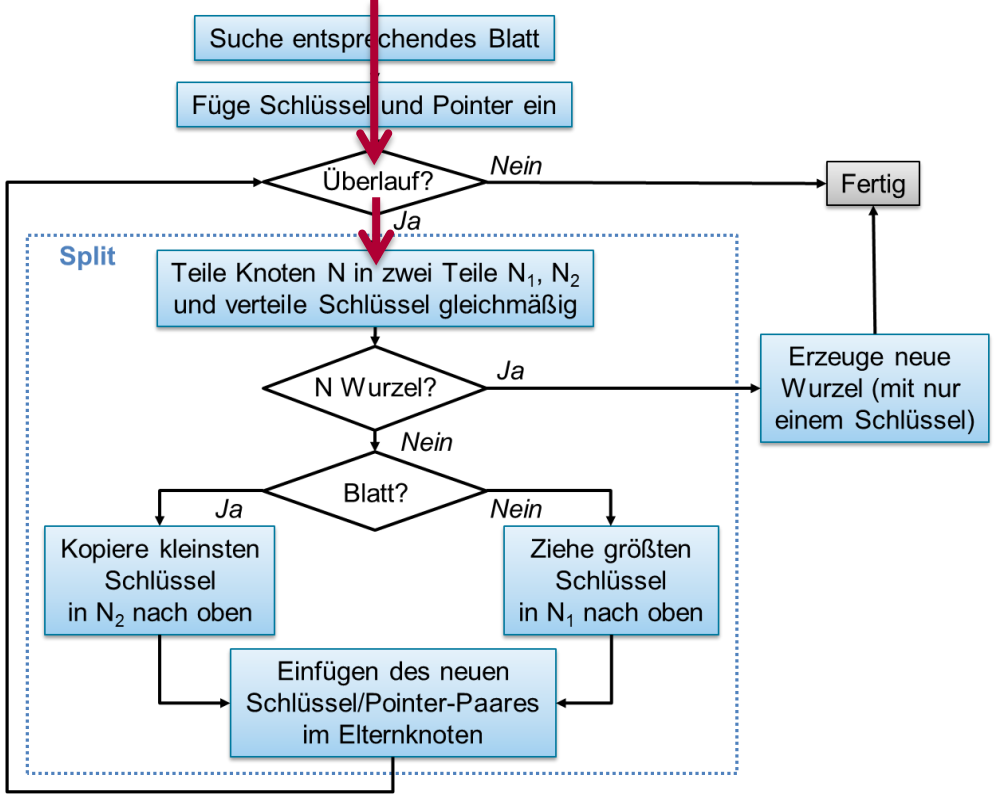
- INSERT(17)
- INSERT(18)

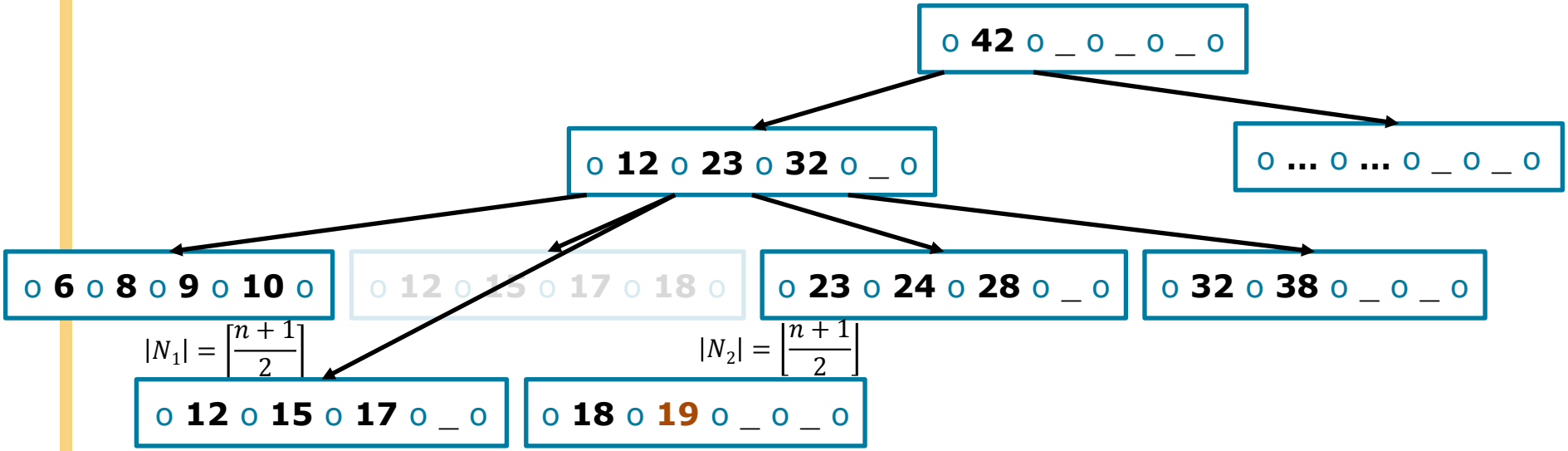




B⁺-Bäume: INSERT

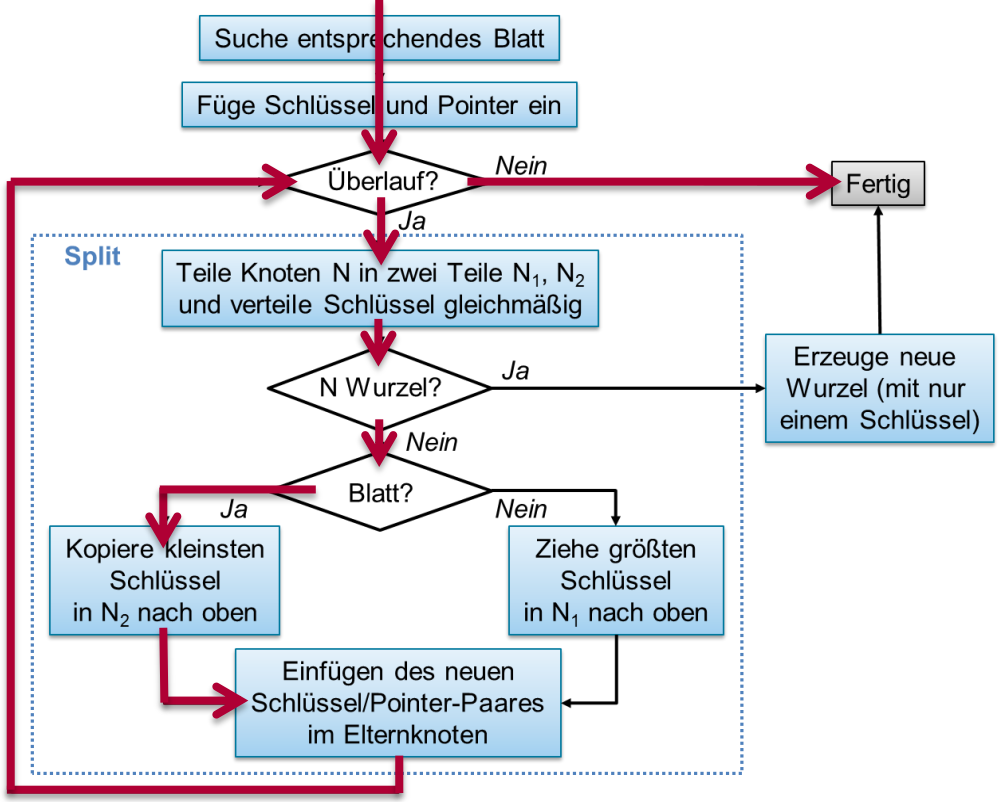
- INSERT(17)
- INSERT(18)
- INSERT(19)

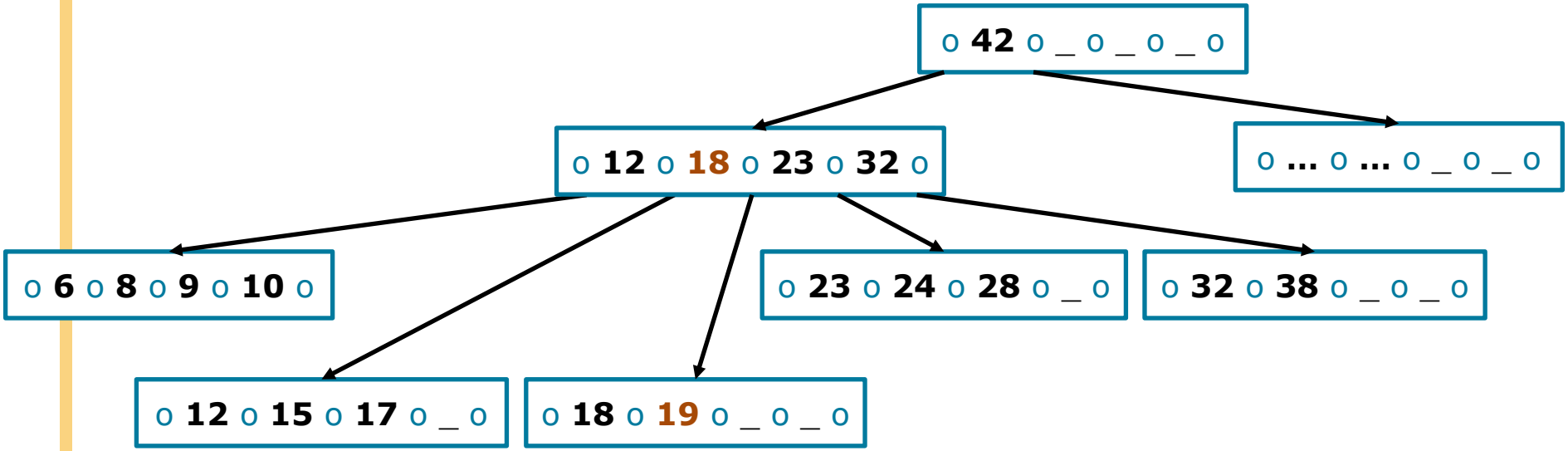




B+-Bäume: INSERT

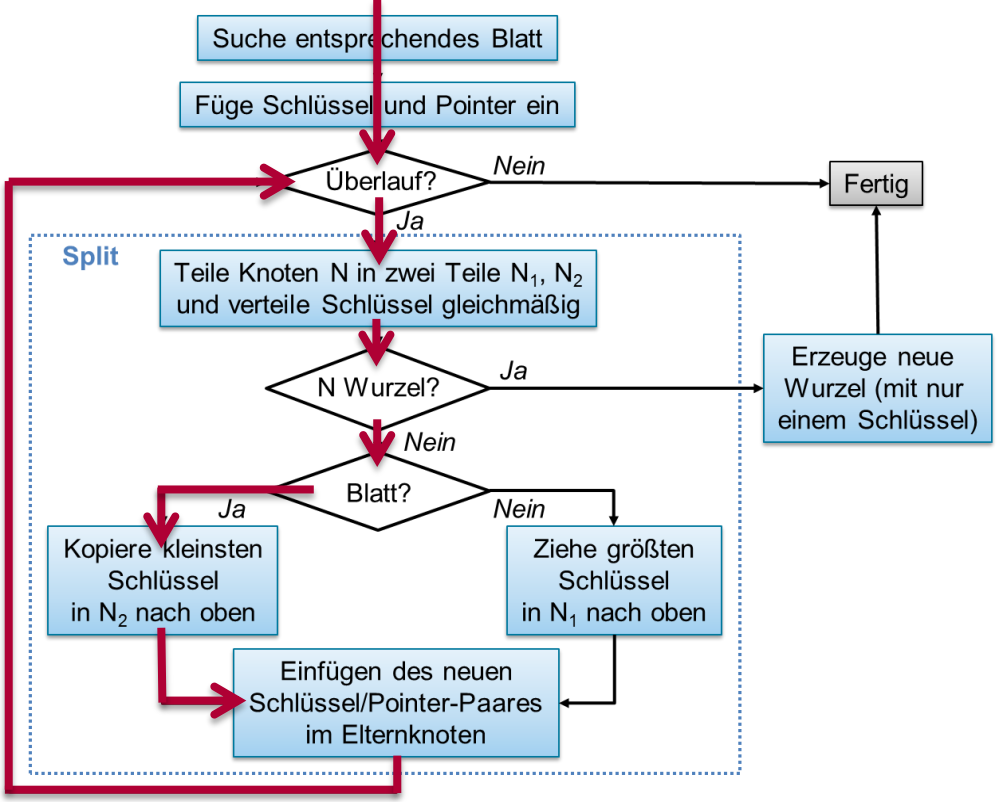
- INSERT(17)
- INSERT(18)
- INSERT(19)

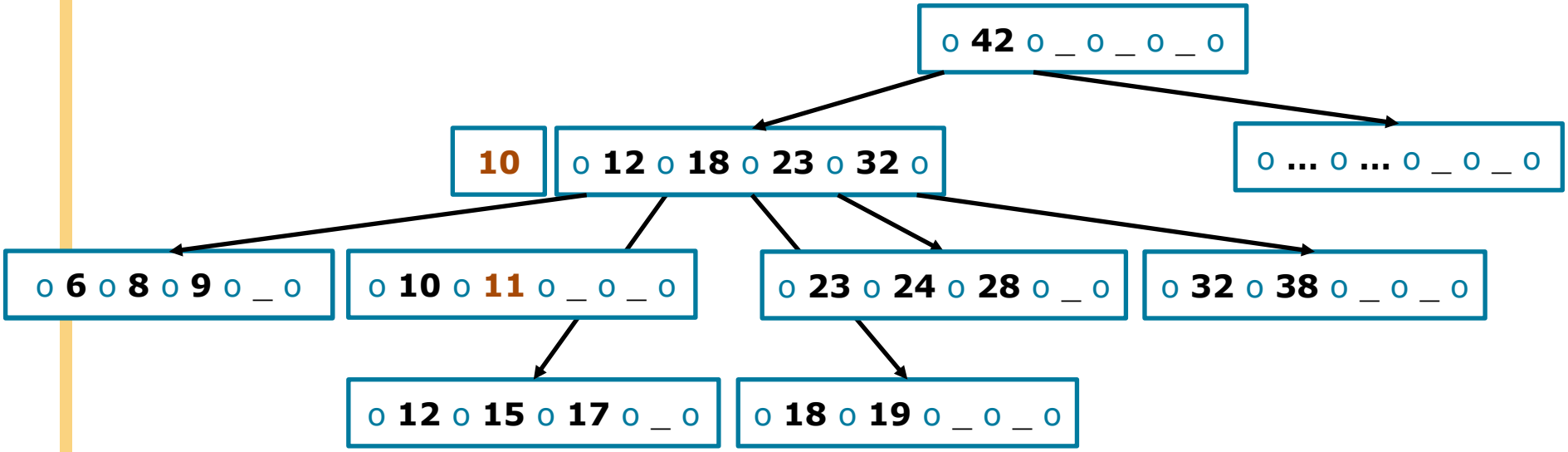




B+-Bäume: INSERT

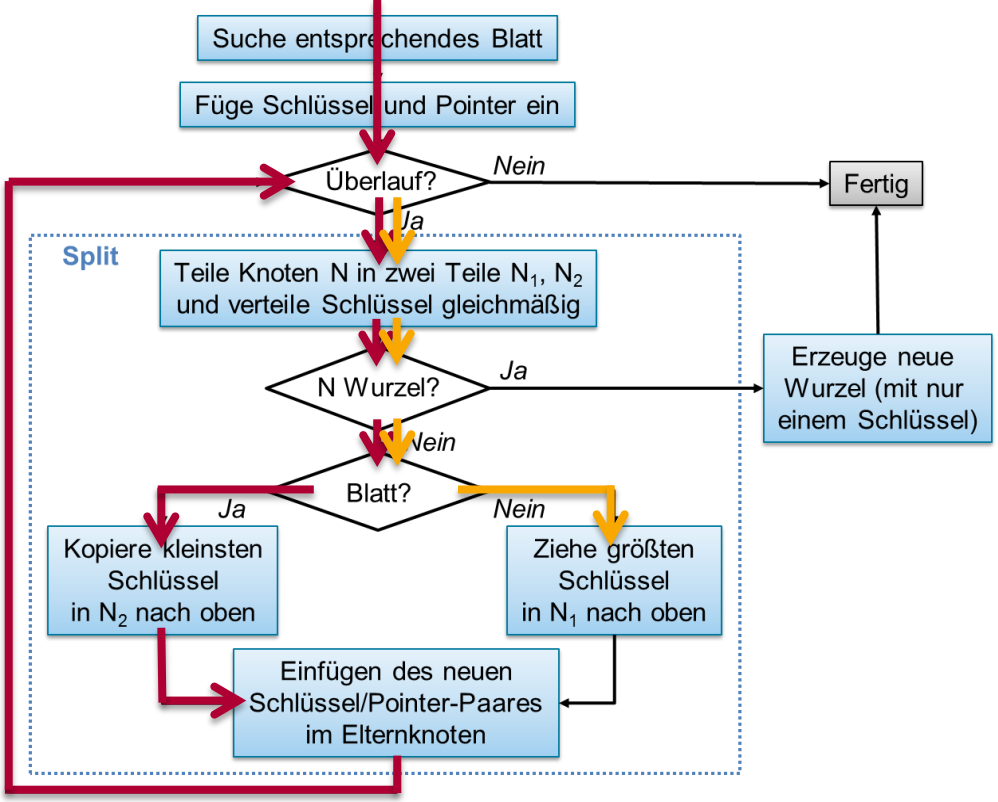
- INSERT(17)
- INSERT(18)
- INSERT(19)
- INSERT(11)

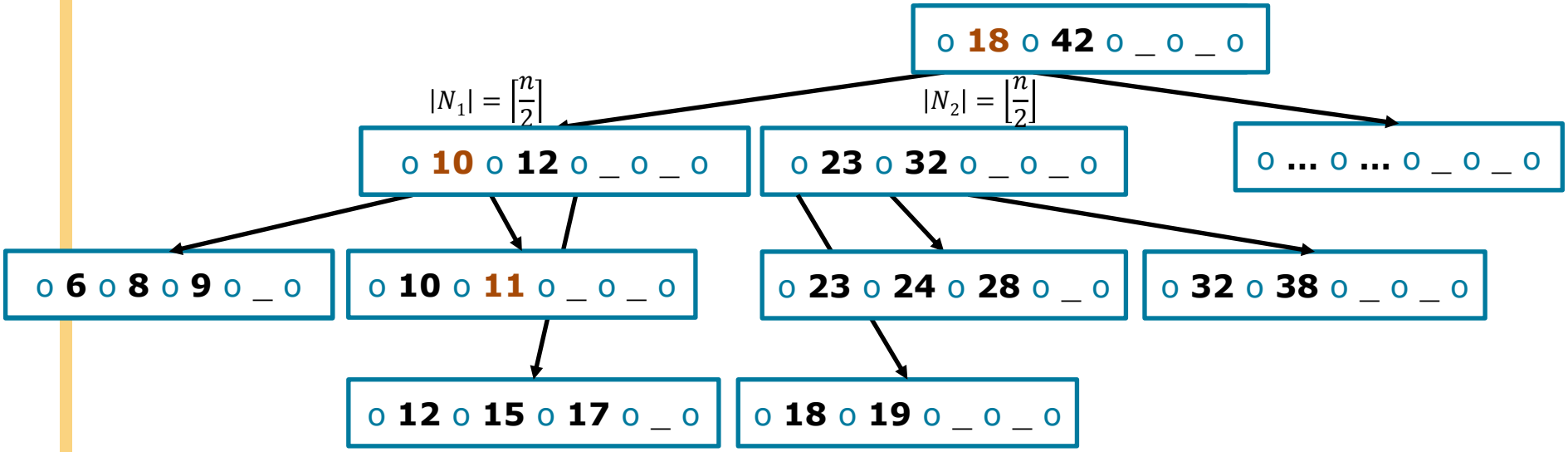




B⁺-Bäume: INSERT

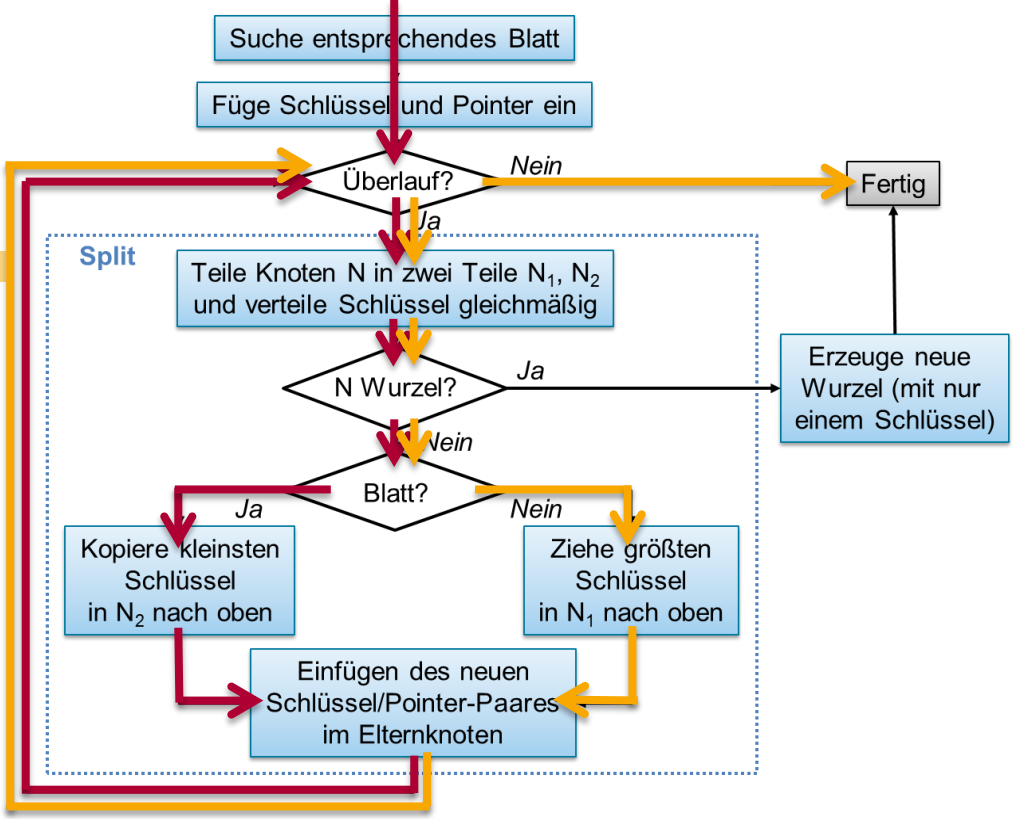
- INSERT(17)
- INSERT(18)
- INSERT(19)
- INSERT(11)

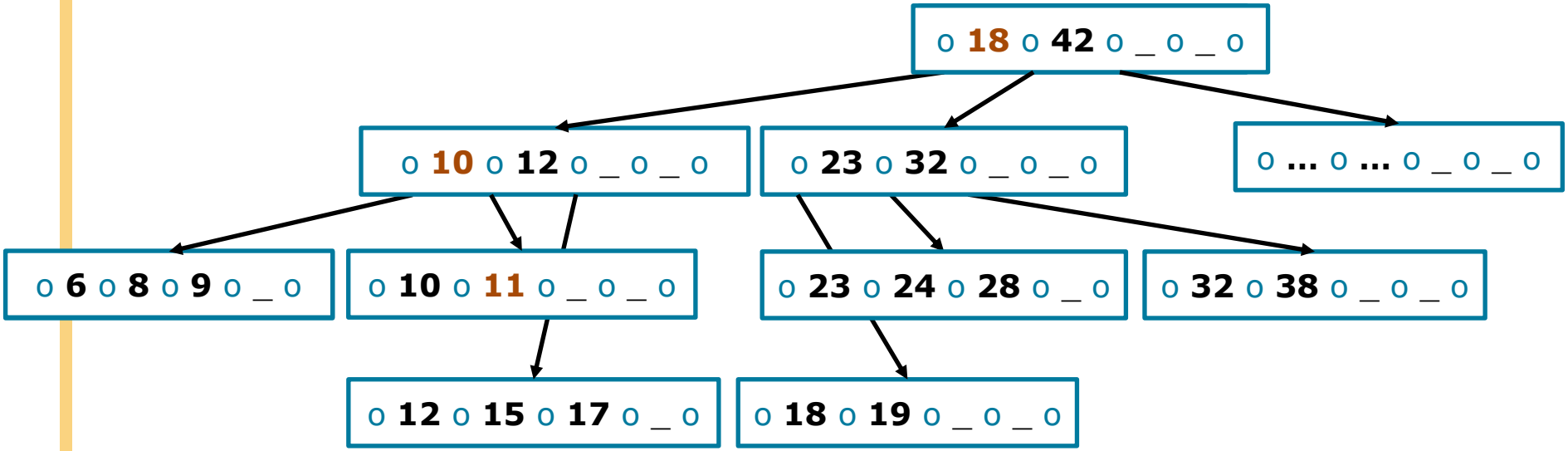




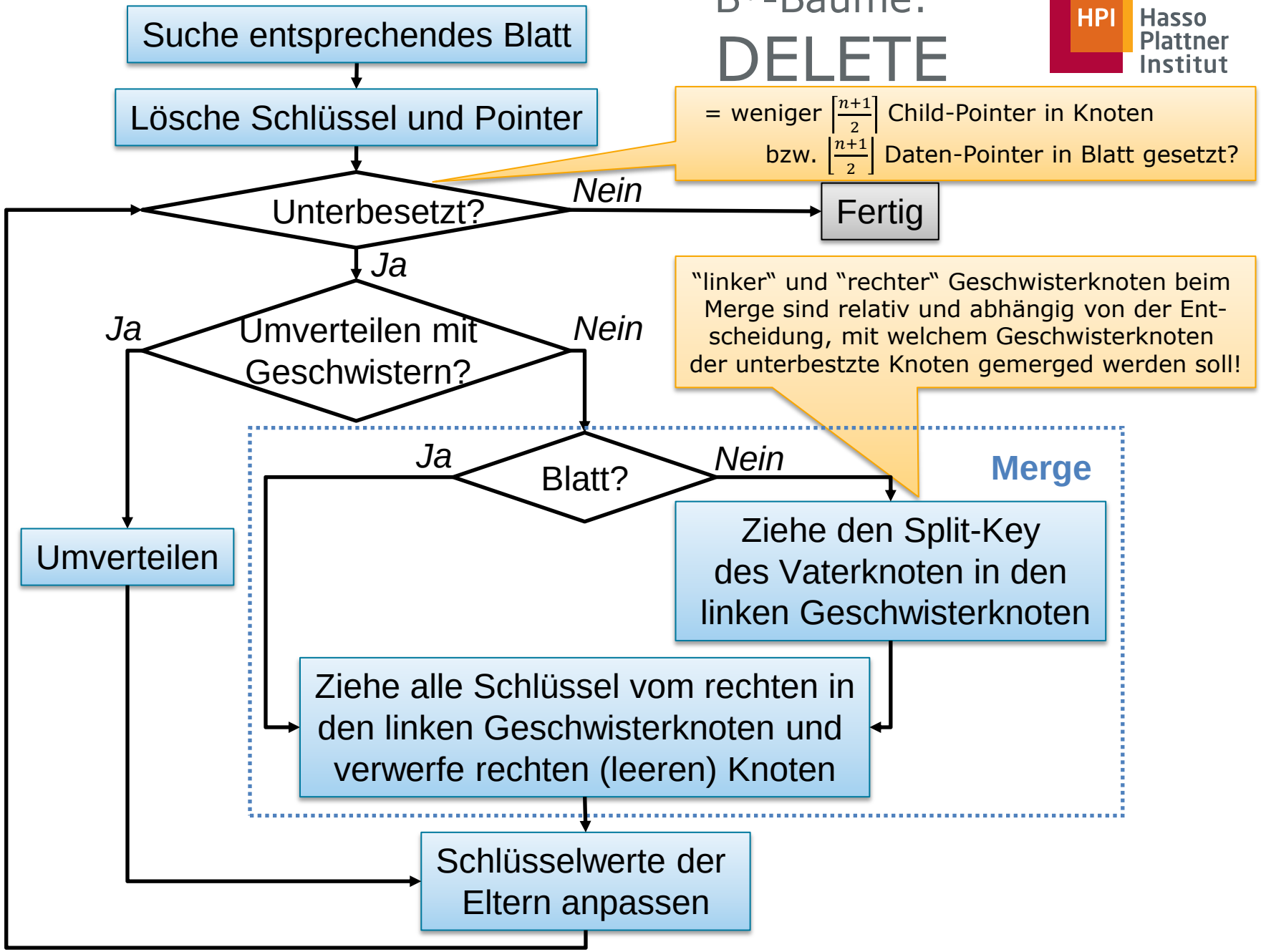
B+-Bäume: INSERT

- INSERT(17)
- INSERT(18)
- INSERT(19)
- INSERT(11)



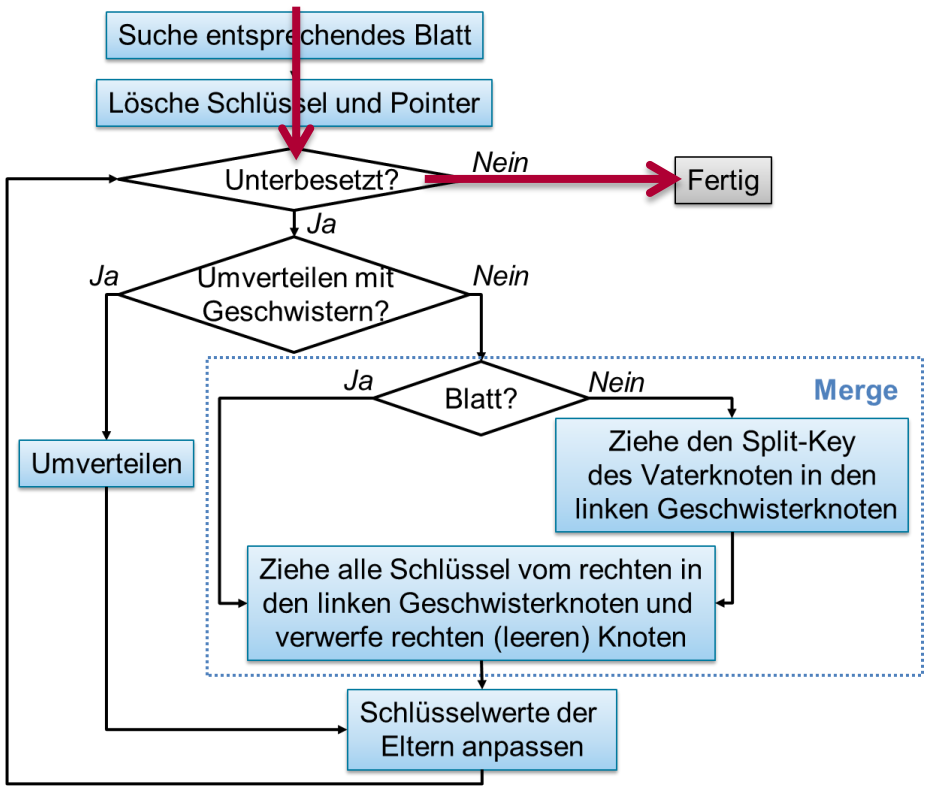


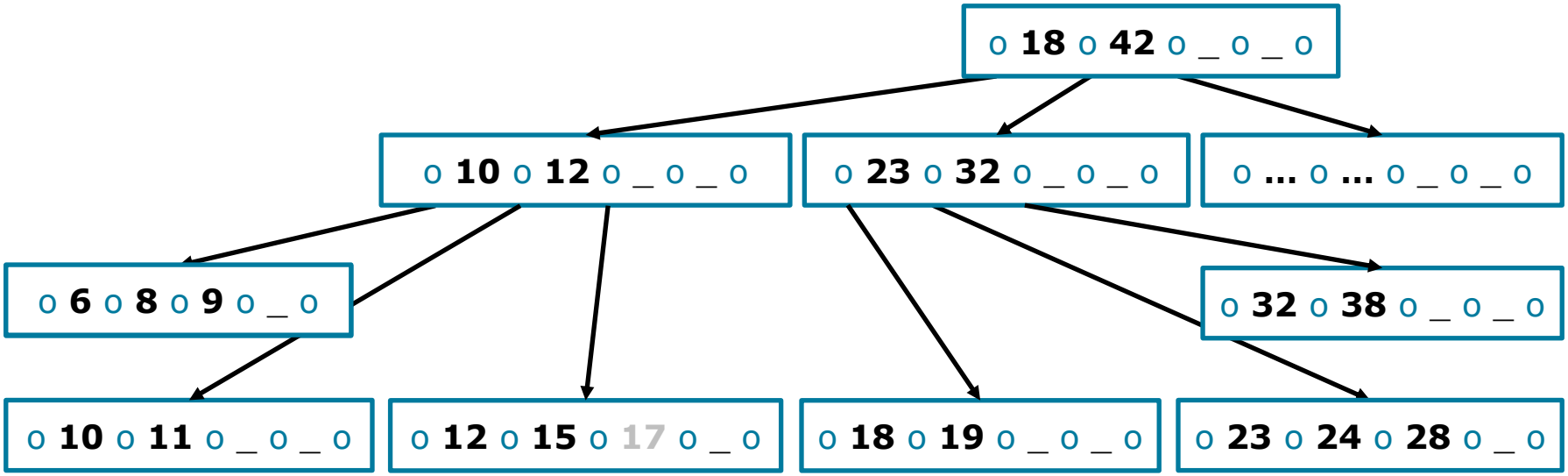
B⁺-Bäume: DELETE



B⁺-Bäume: INSERT

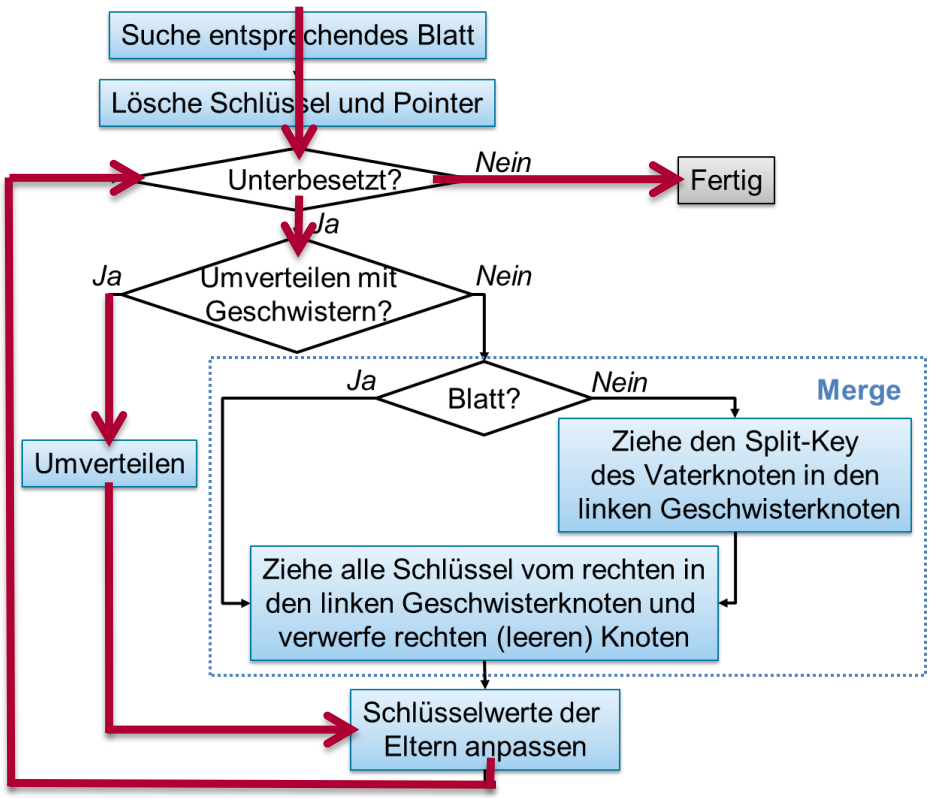
- DELETE(17)



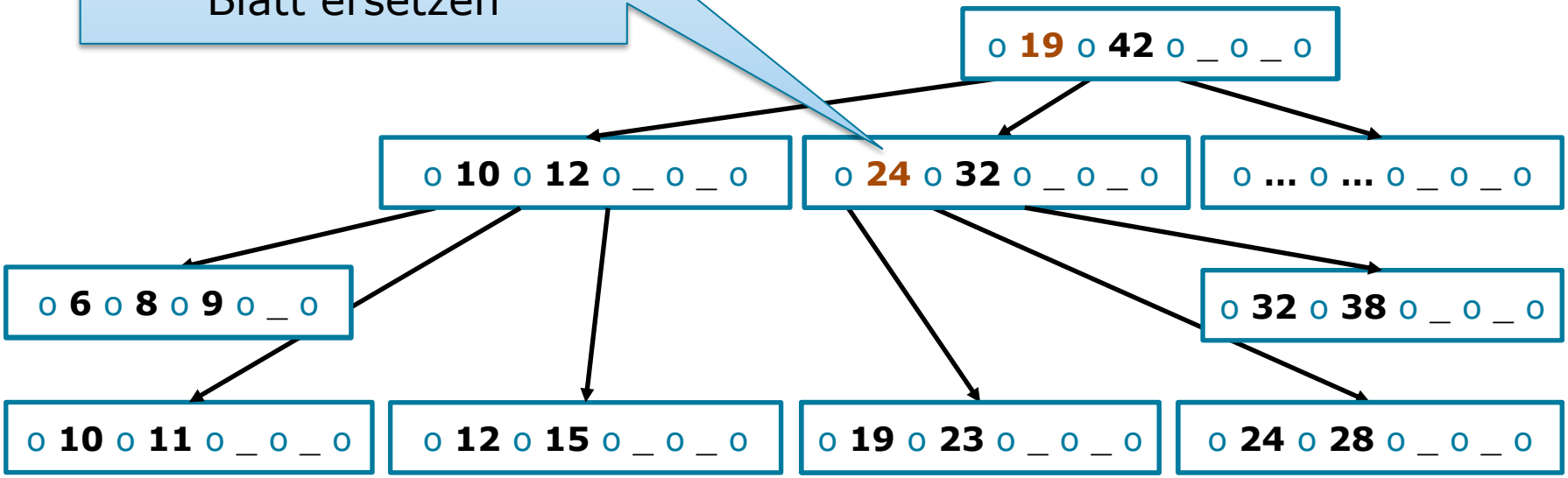


B⁺-Bäume: INSERT

- DELETE(17)
- DELETE(18)

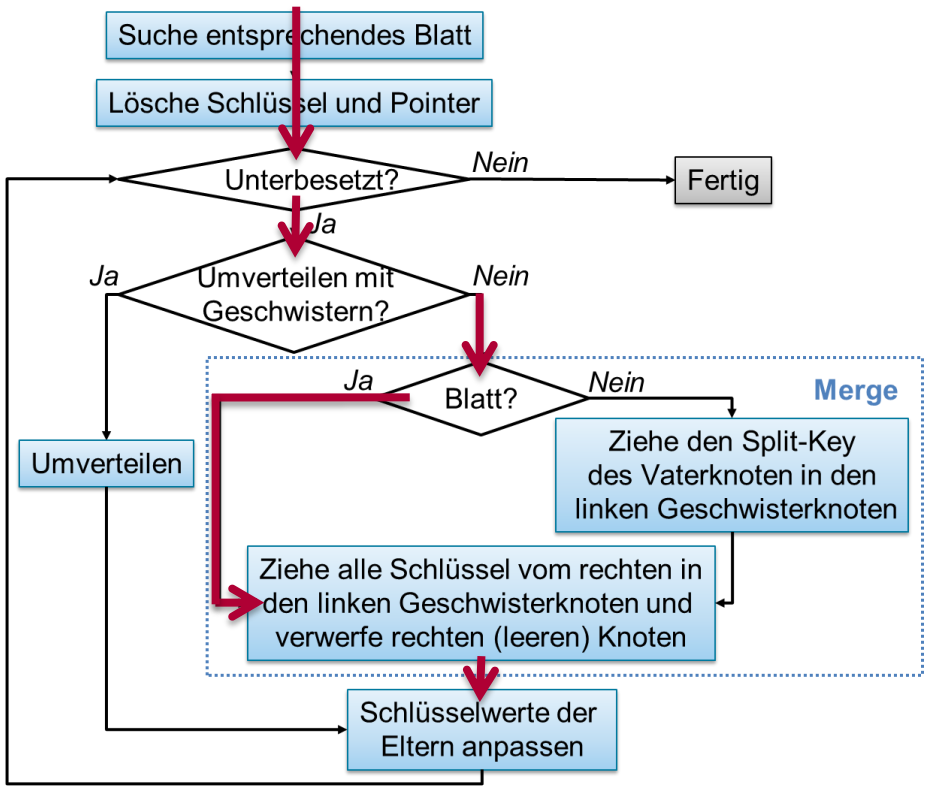


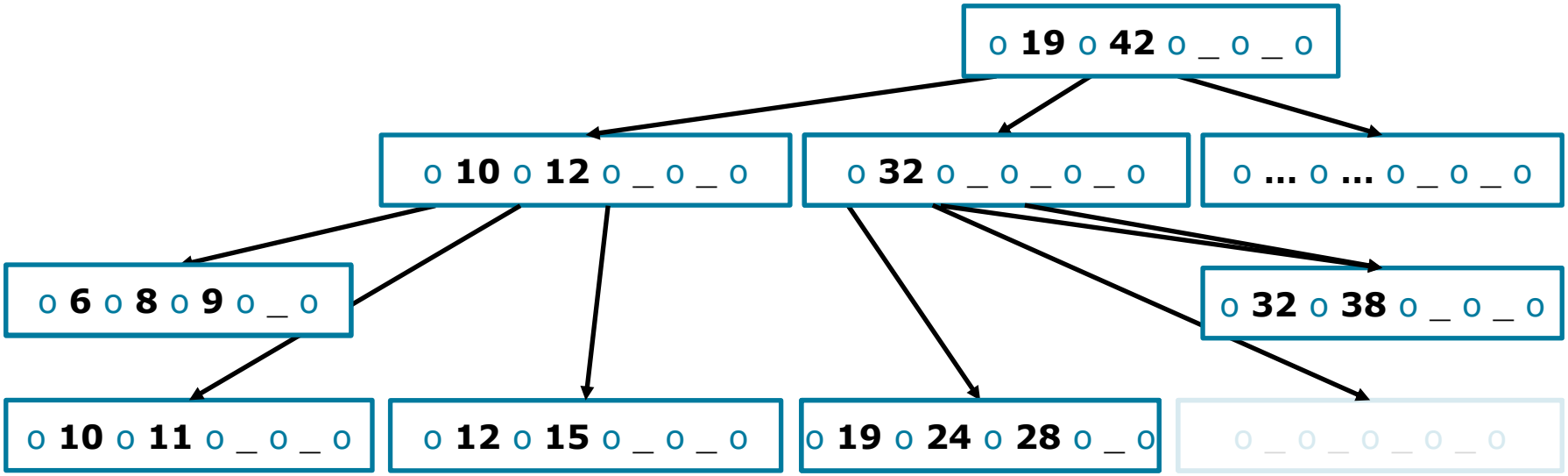
Ursprünglichen Schlüssel durch neuen kleinsten Schlüssel im rechten Blatt ersetzen



B⁺-Bäume: INSERT

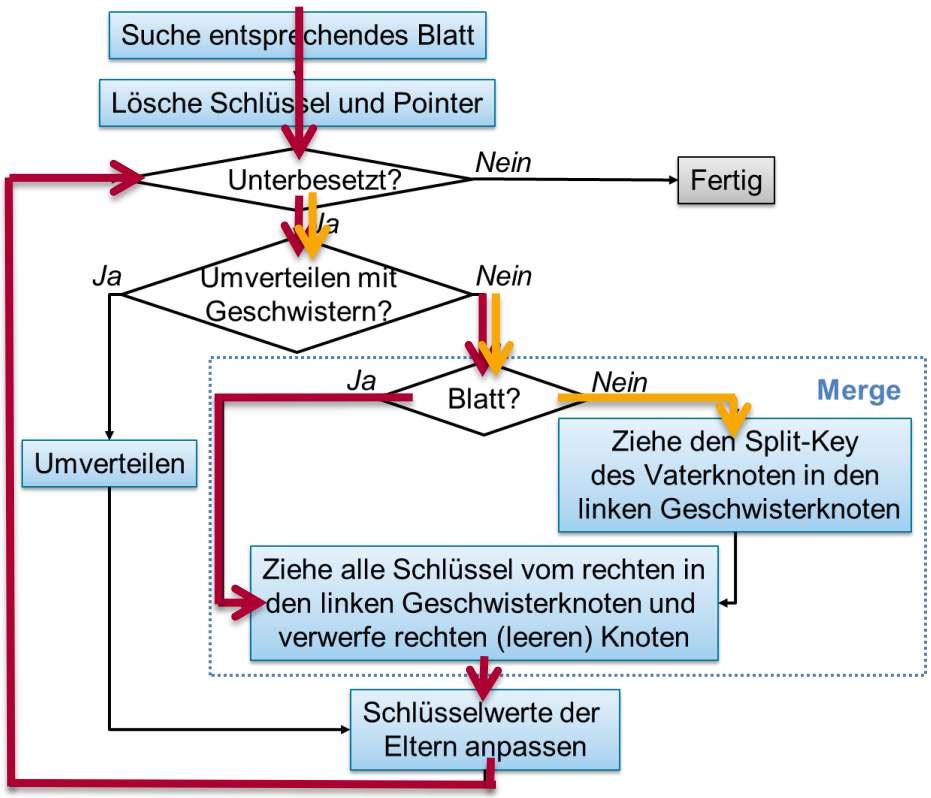
- DELETE(17)
- DELETE(18)
- DELETE(23)

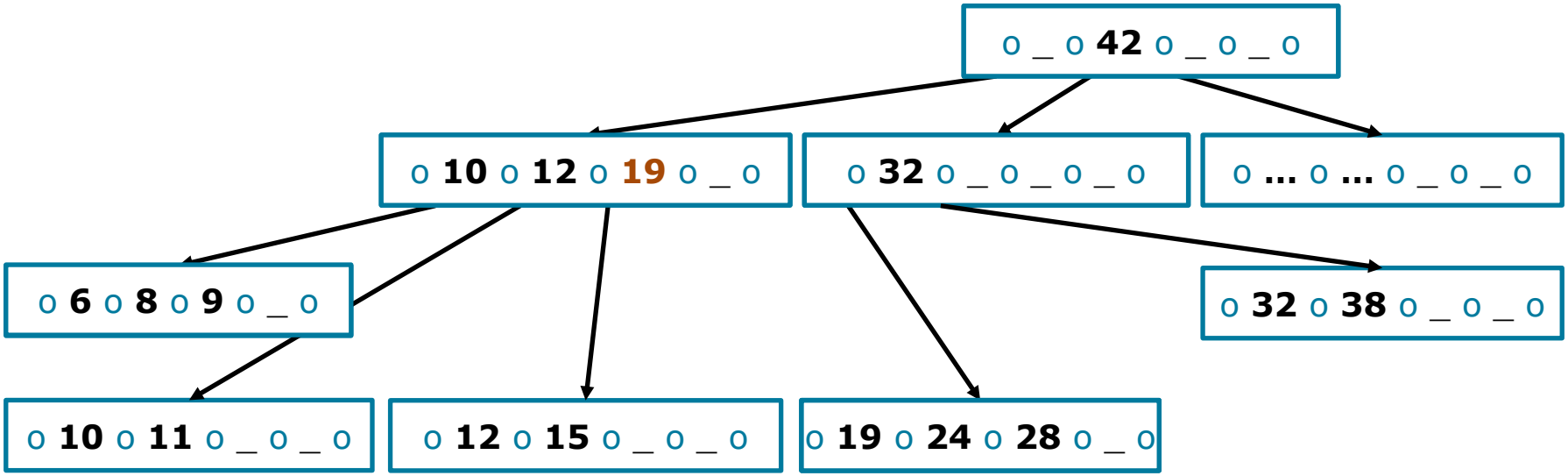




B⁺-Bäume: INSERT

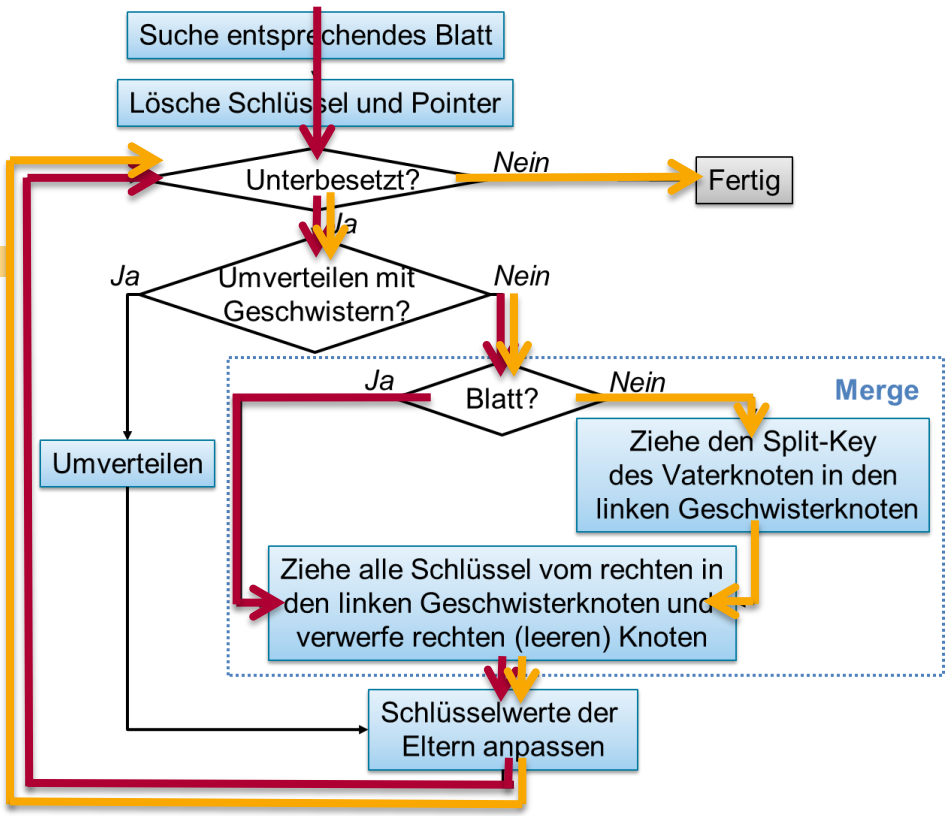
- DELETE(17)
- DELETE(18)
- DELETE(23)

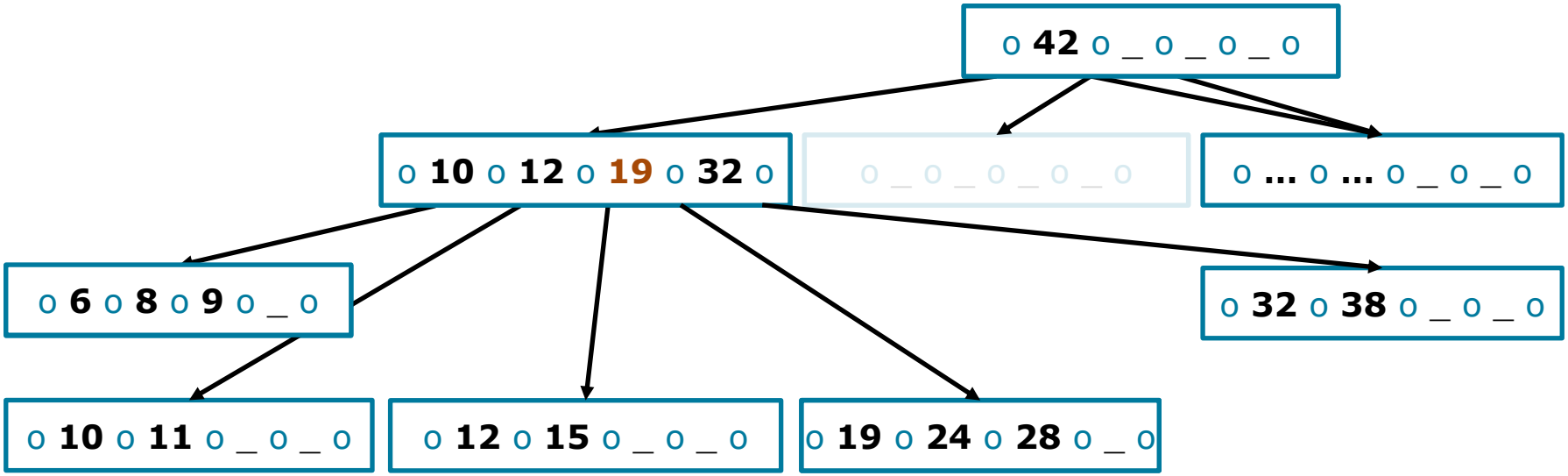




B⁺-Bäume: INSERT

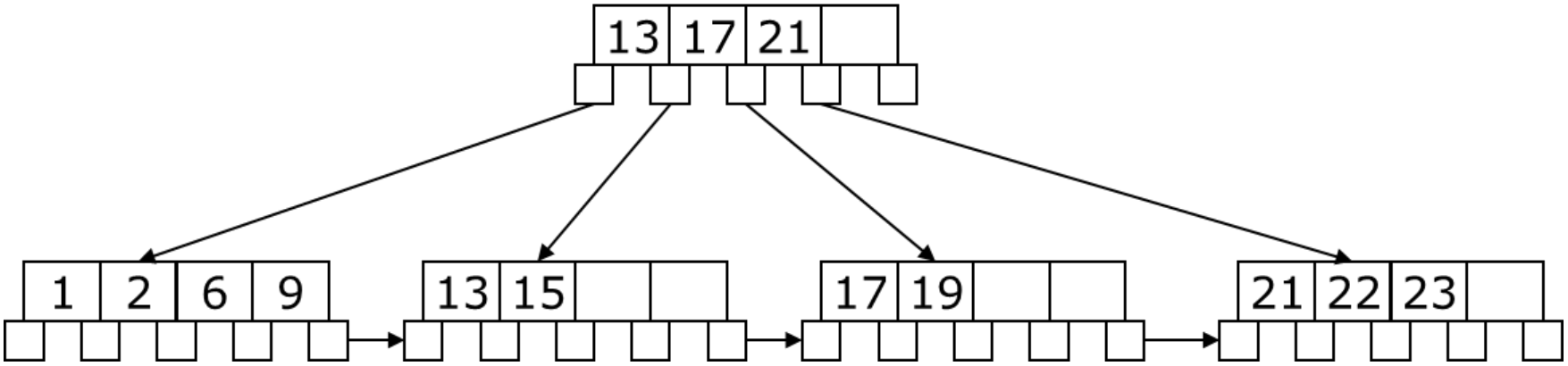
- DELETE(17)
- DELETE(18)
- DELETE(23)





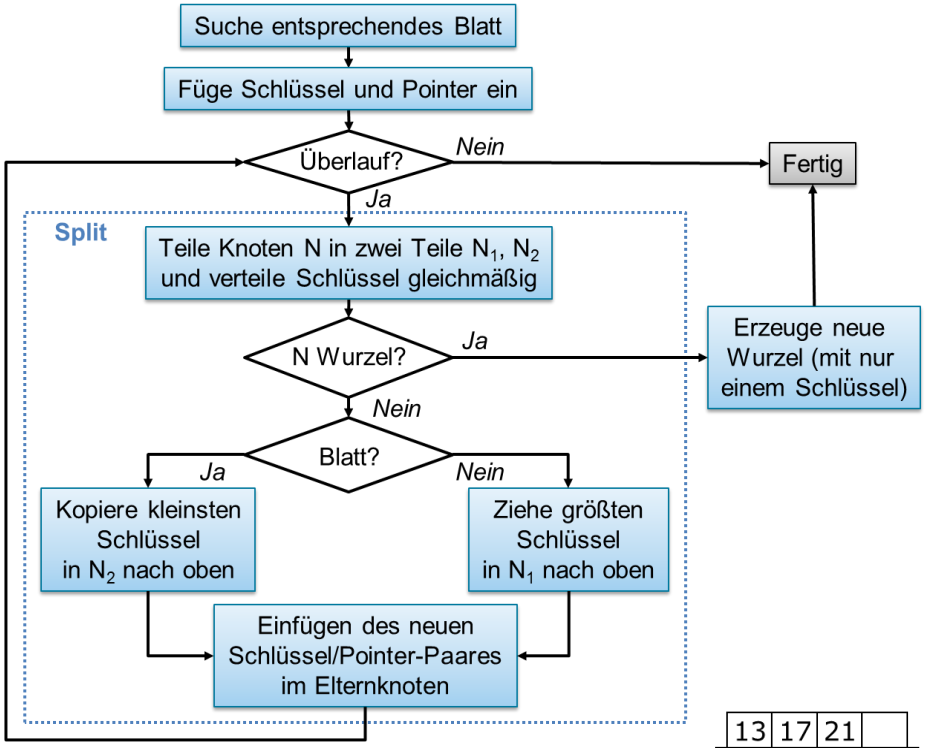
B+-Bäume: Übungsaufgabe

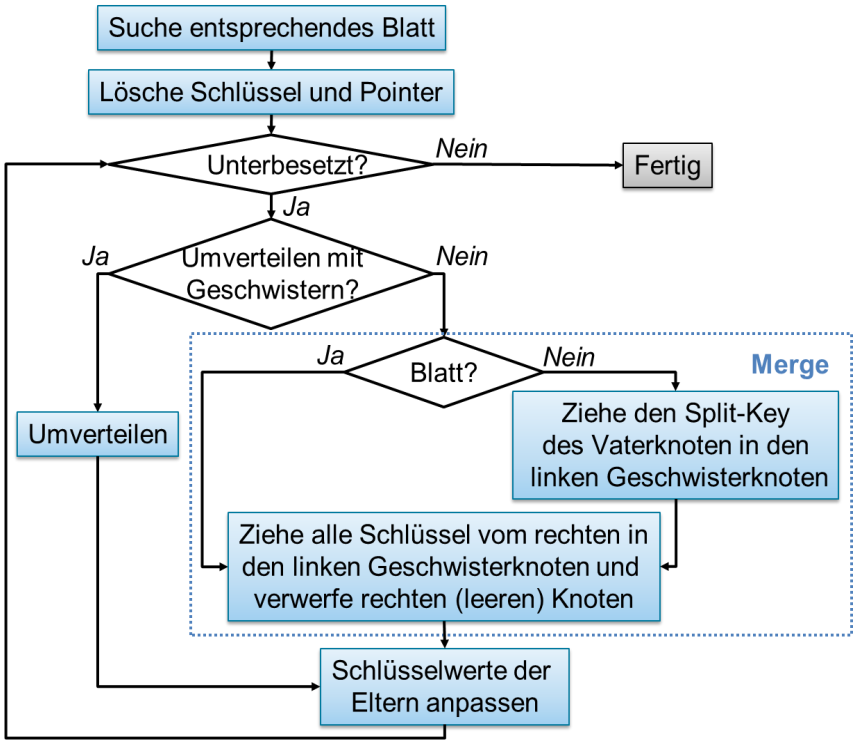
- Gegeben: B+-Baum, entstanden durch Einfügen von 9, 22, 19, 21, 13, 17, 1, 6, 23, 15

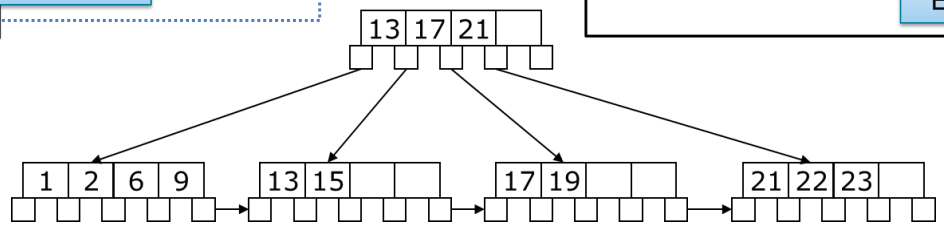


1. Füge nacheinander ein: 14, 29, 4, 24
2. Lösche nacheinander: 6, 13, 15

B+-Bäume: Übungsaufgabe – Übersicht

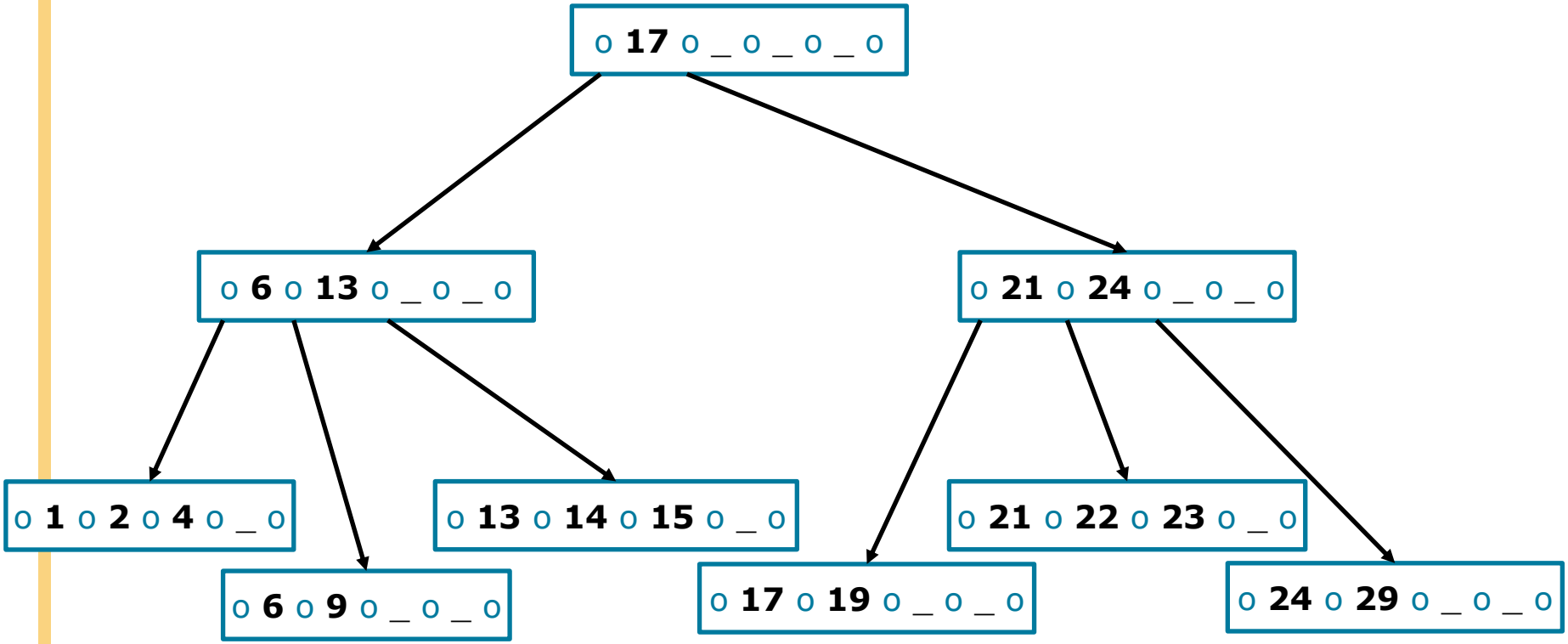






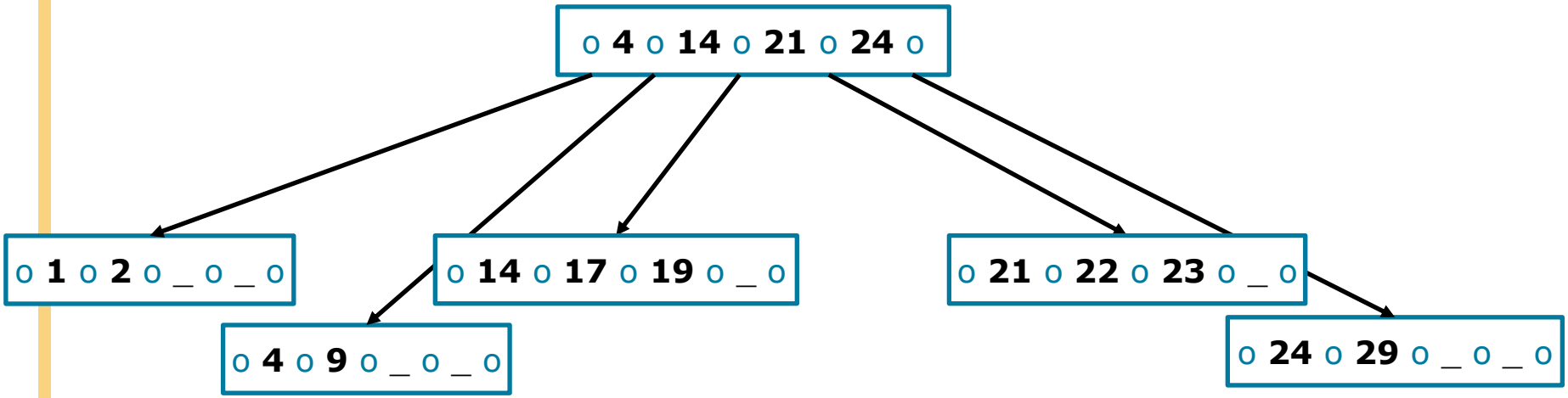
1. Füge nacheinander ein: 14, 29, 4, 24
2. Lösche nacheinander: 6, 13, 15

B⁺-Bäume: Übungsaufgabe – Lösung 1



1. Füge nacheinander ein: 14, 29, 4, 24
2. Lösche nacheinander: 6, 13, 15

B+-Bäume:
Übungsaufgabe – Lösung 2



1. Füge nacheinander ein: 14, 29, 4, 24
2. Lösche nacheinander: 6, 13, 15