Fachhochschule
Nordwestschweiz

04 Bäume - B-Bäume

Algorithmen und Datenstrukturen 2

Rechenspiele

Beispiel

Schw. Bevölkerung: ~ 8’000’000 Personen

Schüssel: 64 Byte

Daten pro Person: 2048 Byte

Daten Total: ~ 16 Gigabyte

Höhe vollständiger* Binärbaum: log_2(8’000’000) = 23

Höhe AVL-Baum: 1.1*log_2(8’000’000) = 26

Höhe entspricht durchschnittlicher Anzahl Speicherzugriffe bei einer Suchanfrage.

Binäre (balanciete) Suchbäume sind *nicht effizient* bei externem Speicher!

Mehrwegbaum

- Ein Baumknoten entspricht einem Diskblock.
- Jeder Knoten kann bis zu s Elemente speichern.
- Jeder Knoten hat *s+1* Nachfolger.
- Die Höhe *h* bei *N* gespeicherten Elementen ist *h = log_{s+1}(N+1).*

```
class Node<K> {
    int m;
    K[] keys = (K[])new Object[CAPACITY];
    int[] data = new int[CAPACITY];
    int[] successor = new int[CAPACITY + 1];
}
```

Mehrwegbaum

Aufgabe

Veranschlagen wir jeweils 4 Byte pro Referenz auf einen (Nachfolge-)Knoten, sowie 4 Byte als Referenz auf die zu einem Schlüssel gehörenden Daten.
Wie viele 64-Byte Schlüssel passen dann in einen Knoten mit 4096 Bytes (häufige Block-Grösse) und wie hoch wird ein Mehrwegbaum dann für 8·10^6 Elemente mindestens?

(Tipp: Berechnen Sie zuerst wie viele Elemente in einen Knoten passen.)

Mehrwegbaum

Lösung

Für die Java-Node Spezifikation im Skript:

- int m: 4 Byte
- K[] keys: CAPACYTY * 64 Byte
- int[] data: CAPACITY * 4 Byte
- int[] succ: (CAPACITY + 1) * 4 Byte

Damit können wir die maximale CAPACITY berechnen:

8 Byte + CAPACITY * 72 Byte

→ (4096-8)/72 = 56.6

→ CAPACITY ist maximal 56.

Nun berechnen wir die Höhe mit der Formel:

$Höhe = \lceil \log_{57} 8 \cdot 10^6 + 1 \rceil = 4$

im Idealfall, wenn der Baum vollständig und ausgefüllt wäre.

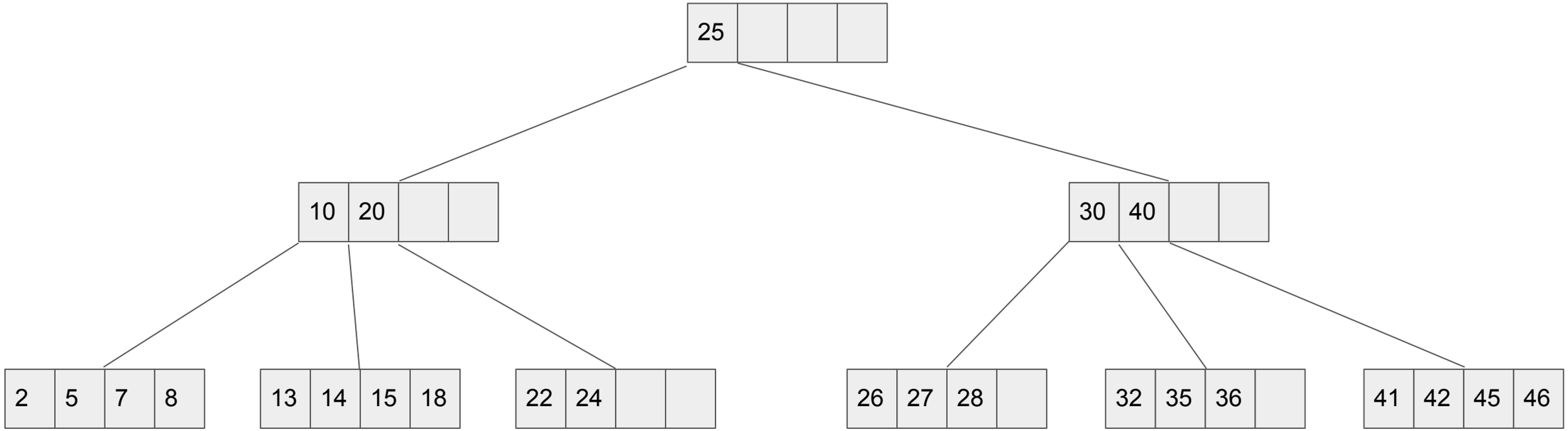
Definition: B-Baum [Bayer / McCreight 1970]

***Achtung**: Für B-Bäume wird der Begriff* **Ordnung** *anders definiert, als für allgemeine Bäume!*

Ein B-Baum der Ordnung *n* ist ein Mehrwegsuchbaum vom Grad *2n + 1* mit den Struktur-Eigenschaften:

1. Jeder Knoten enthält höchstens *2n* Elemente.
2. Jeder Knoten ausser der Wurzel enthält mindestens *n* Elemente (ist zur Hälfte gefüllt).
3. Jeder Knoten, der nicht Blattknoten ist, hat *m+1* Nachfolger, wobei *m* die Anzahl der Elemente ist.
4. Alle Blattknoten liegen auf derselben Stufe.
5. Ein Knoten enthält
   - *m* Schlüssel + Daten (bzw. Verweise auf einen Datensatz); die Schlüssel sind aufsteigend sortiert
   - *m +1* Referenzen auf Nachfolger.

B-Baum der Ordnung 2



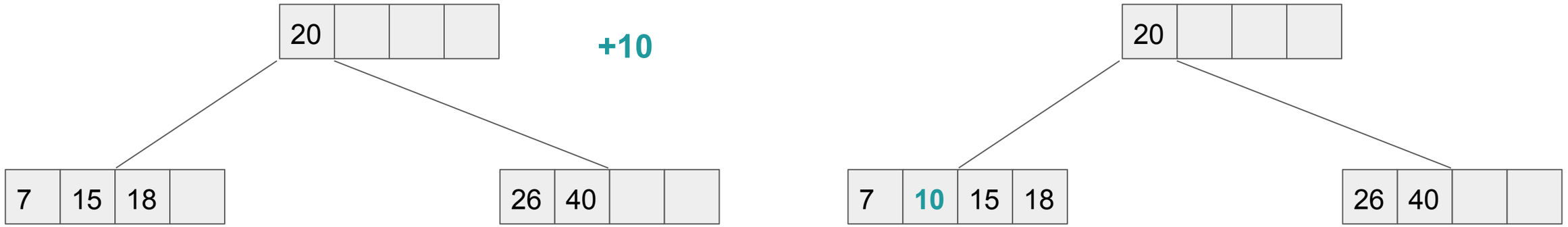
```
class Node<K> {
   int m;
   K[] keys = (K[])new Object[CAPACITY];
   int[] data = new int[CAPACITY];
   int[] successor = new int[CAPACITY + 1];
}
```

Suchen im B-Baum

```
E find(Node<K> node, K key) {
    int i = 0;
    while (i < node.m && key.compareTo(node.keys[i]) > 0) {
        i++;
    }
    if (i < node.m && key.equals(node.keys[i])) {
        return dataBlock(node.data[i]);
    }
    if (node.isLeaf()) {
        return null;
    }
    Node<K> child = diskRead(node.successor[i]);
    return find(child, key);
}
```

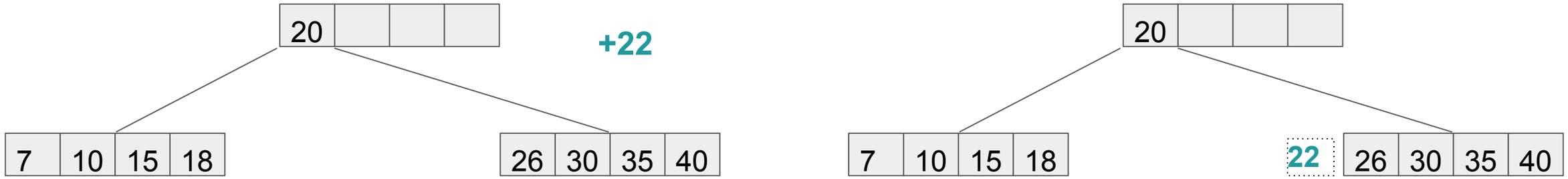
Einfügen in den B-Baum - IMMER in ein Blatt

Fall 1: Neues Element wird in Blatt mit m < 2n Elementen eingefügt

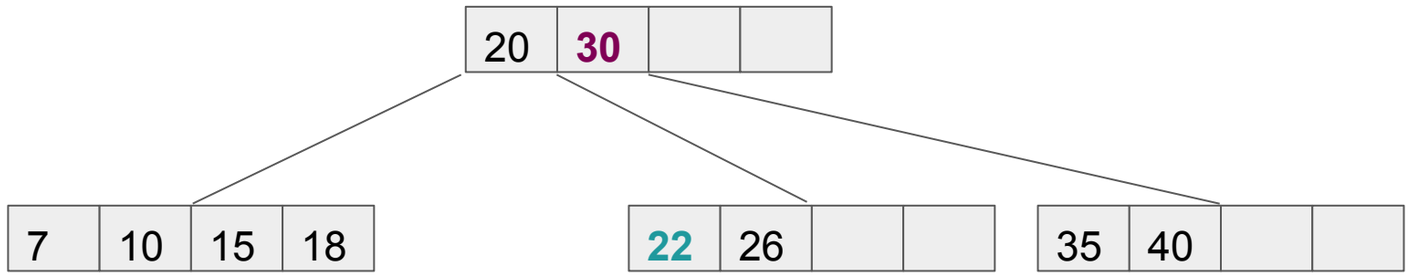


Einfügen in den B-Baum - IMMER in ein Blatt

Fall 2: Blatt ist bereits voll, wir müssen “splitten”

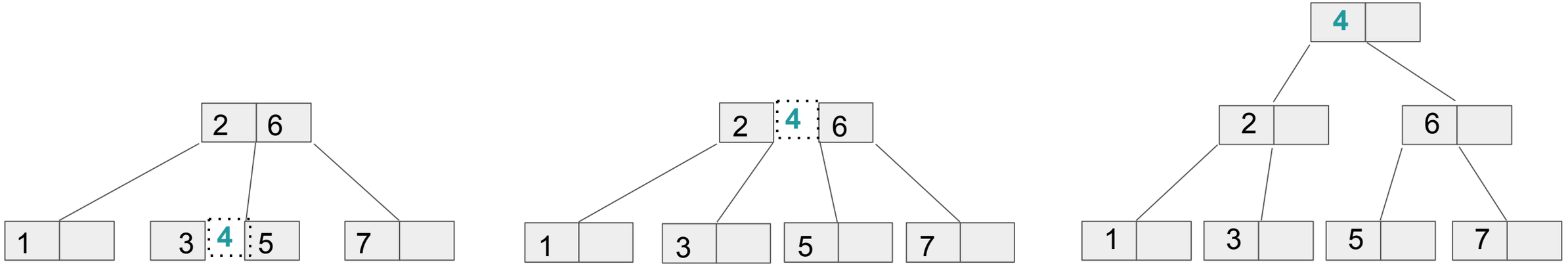


- Schlüssel 22 hat keinen Platz im Blatt / das Blatt hat nun zu viele Schlüssel
- Das Blatt wird “gesplittet”, der mittlere Schlüssel (also die 30) wird in den Vater geschoben.



Einfügen in den B-Baum - IMMER in ein Blatt

Fall 2b: Blatt ist bereits voll, wir müssen “splitten” inklusive Vorgänger



- Schlüssel 4 hat keinen Platz im Blatt / das Blatt hat nun zu viele Schlüssel
- Das Blatt wird “gesplittet” und der mittlere Schlüssel (also die 4) wird in den Vater geschoben.
- Der Vaterknoten ist bereits voll, es wird “gesplittet”. In unserem Beispiel entsteht eine neue Wurzel.
- Das «Splitten» kann also auf dem Pfad vom Blatt bis zur Wurzel propagieren.

B-Baum

Aufgabe

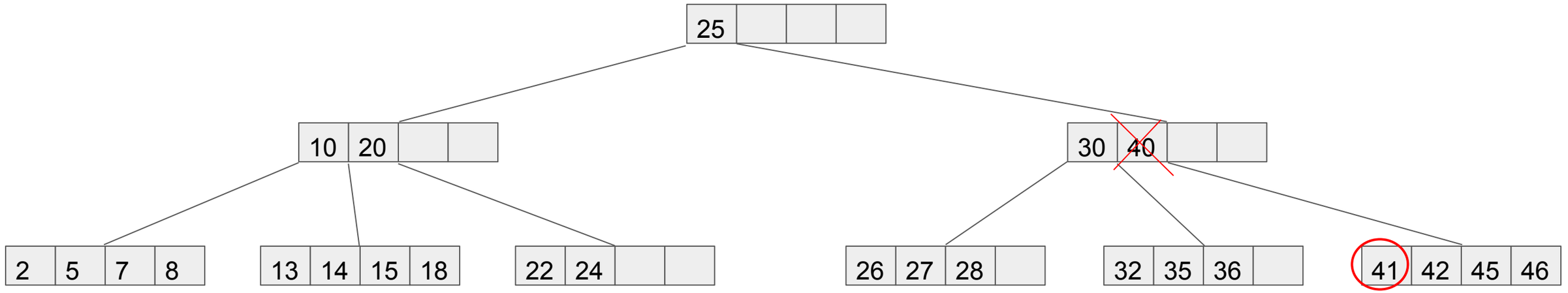
Lösen Sie die Aufgabe 1a auf dem Arbeitsblatt

Löschen aus B-Baum

Fall 1: Element befindet sich in einem Blatt

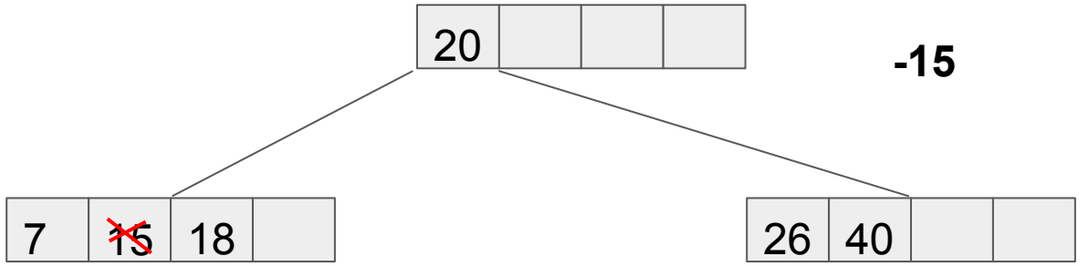
Fall 2: Element befindet sich in einem inneren Knoten:

- Ersetzen durch symmetrischen Nachfolger (wie im Binärbaum) und
- Nachfolger Löschen. Dieser befindet sich IMMER in einem Blatt **->** Fall 1.

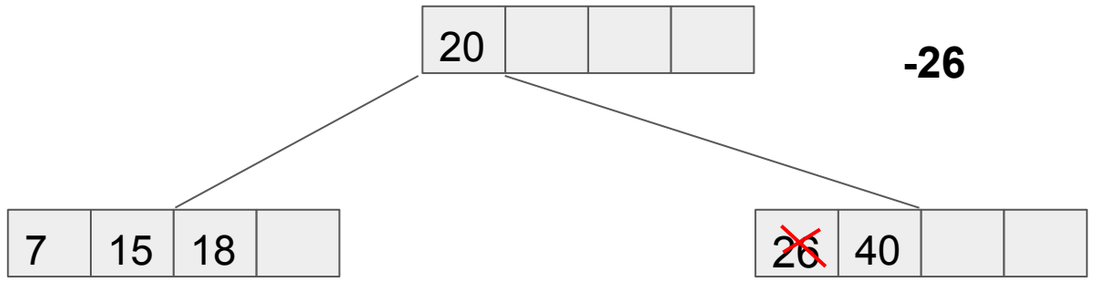


Blattlöschen aus B-Baum

Fall a: Blatt hat m > n Elemente:

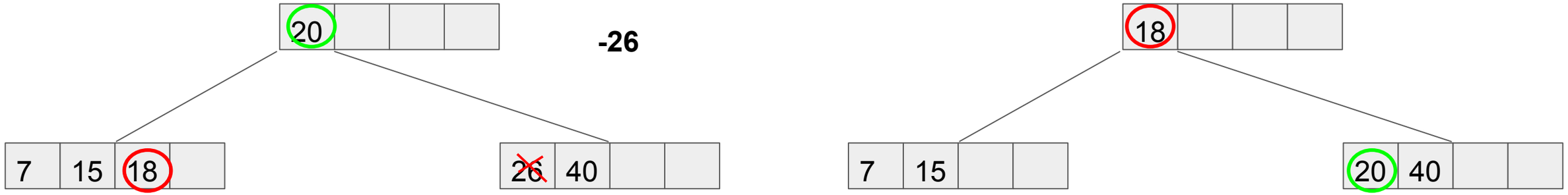


Fall b: Blatt hat genau n Elemente:



- Blatt ist nicht mehr halbvoll.
- B-Baum Bedingungen wieder herstellen durch «*Ausgleichen*».
- Zwei Strategien zum Ausgleichen: «*Ausleihen*» und «*Zusammenlegen*».

Blattlöschen aus B-Baum - Ausleihen



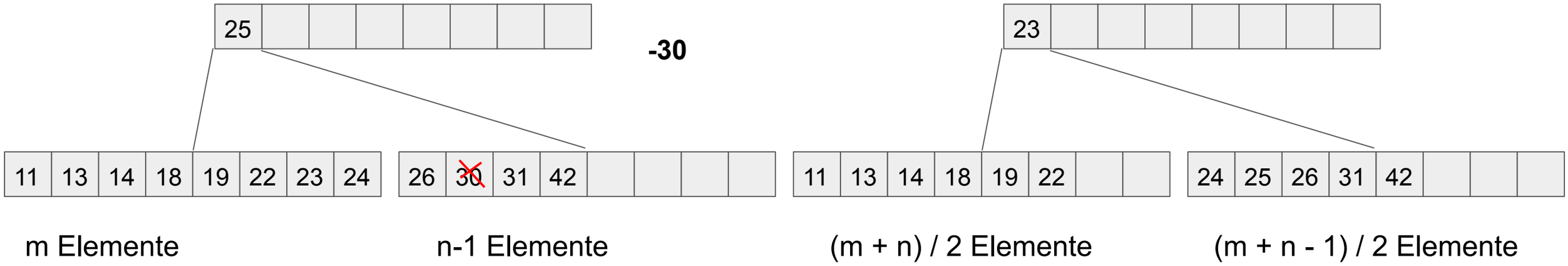
- Das rechte Blatt hat zu wenig Elemente. Das linke Blatt ist mehr als zur Hälfte voll.

- Das rechte Blatt «leiht» sich ein Element von weiter links.
- Das Element mit dem *grössten* Schlüssel vom linken Blatt (18) wandert in den Vaterknoten und ein Element des Vaterknotens (20) in das rechte Blatt.

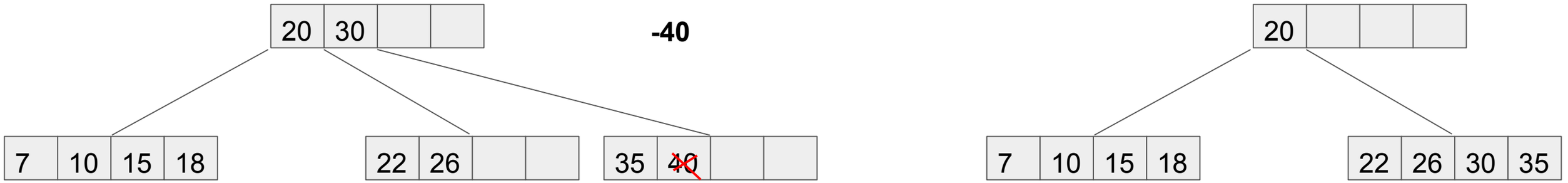
- Beim Ausleihen von links, wandert das Element mit dem *kleinsten* Schlüssel aus dem Nachbarblatt.

Blattlöschen aus B-Baum - Ausleihen - Variante

- Da die Nachbarseite sowieso in den Hauptspeicher geladen werden muss, kann man die Situation ausnützen und die Elemente gleichmässig aufteilen:
  - neu linker Knoten: (*m* + *n*) / 2
  - neu rechter Knoten: (*m* + *n* - 1) / 2

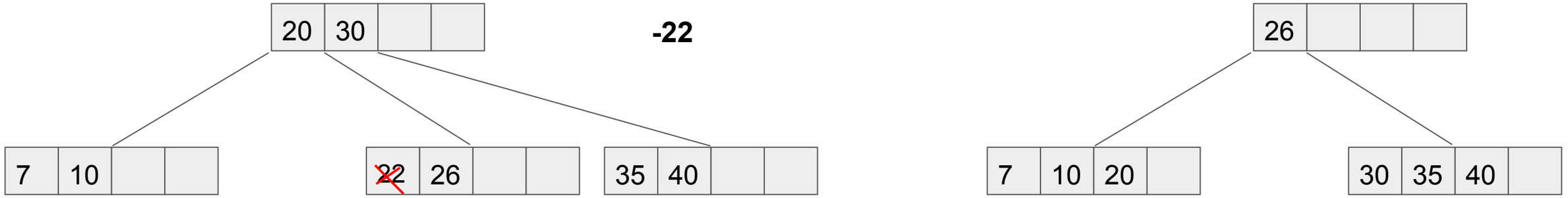


Blattlöschen aus B-Baum - Zusammenlegen



- Kein Nachbarblatt ist mehr als zur Hälfte voll, d.h. “Ausleihen” ist nicht möglich.
- Das Blatt wird deshalb mit einem Nachbarblatt zusammengelegt.
- Dabei wandert auch ein Element des Vaterknotens (hier die 30) ins neue Blatt.

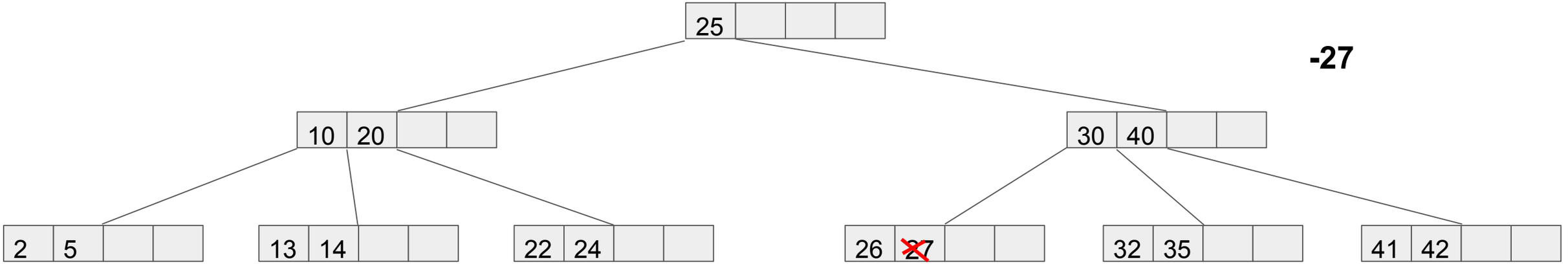
Blattlöschen aus B-Baum - Zusammenlegen - Variante

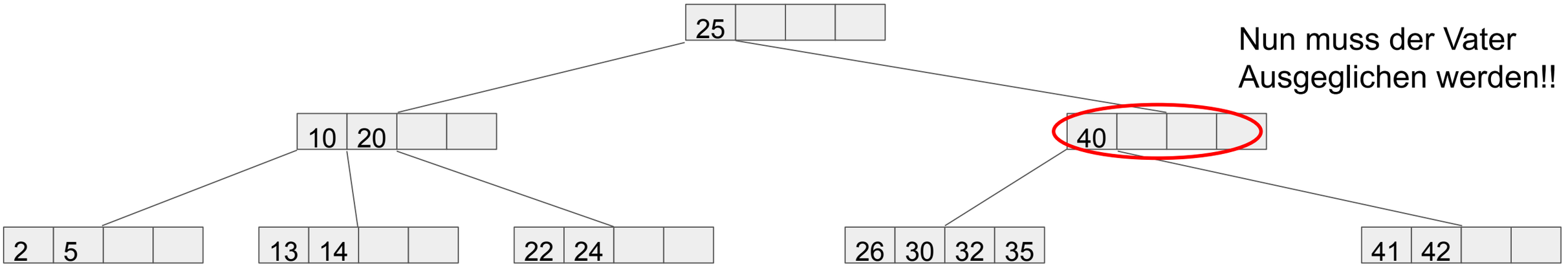


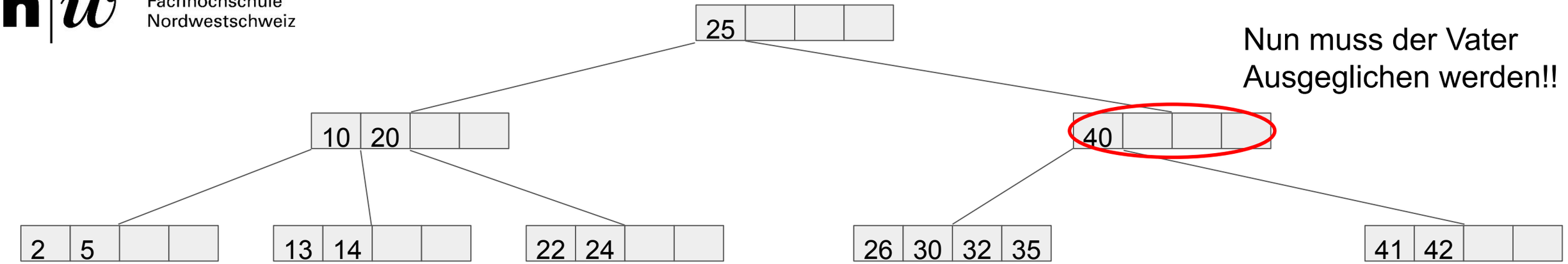
- Kein Nachbarblatt ist mehr als zur Hälfte voll, d.h. “Ausleihen” ist nicht möglich.
- *Zusammenlegen* von drei benachbarten Seiten und Bildung von zwei neuen Seiten daraus.
- Die entstehenden Seiten sind nicht ganz gefüllt aber mehr als halbvoll.
- Vorteil: das nächste Einfügen oder Löschen in einem der drei Blätter ist einfach.

Blattlöschen aus B-Baum - Zusammenlegen

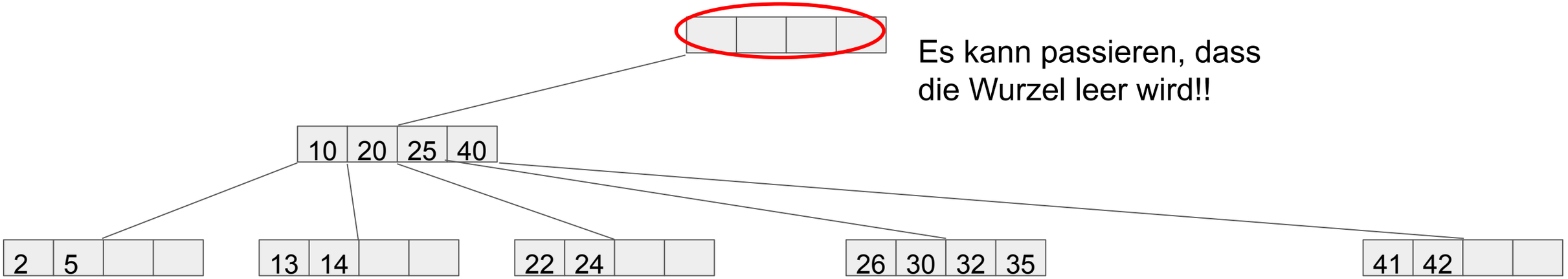
Fall: Vater hat nach dem Verschmelzen zweier Kinder zu wenig Elemente



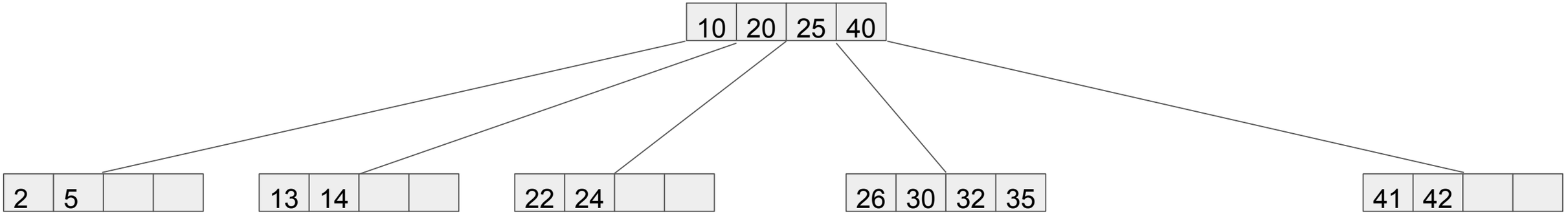




Nun muss der Vater
Ausgeglichen werden!!



Es kann passieren, dass
die Wurzel leer wird!!



B-Baum

Aufgabe

Lösen Sie die Aufgabe 1b auf dem Arbeitsblatt

Laufzeiten im B-Baum (Ordnung n, Anzahl Elemente N)

Suchen:

- Pfad von Wurzel zu Blatt: O(log_n N)

Einfügen:

- Suchen O(log_n N),
- Einfügen häufig konstant, manchmal bis zu O(log_n N)

Löschen:

- Suchen O(log_n N),
- Löschen häufig konstant, manchmal bis zu O(log_n N)

B-Baum

Lösen Sie die Aufgaben 1-3 auf dem Arbeitsblatt.

Lösen Sie die Probeprüfung:

- 90 min, schriftlich
- Kapitel 01-04 (inklusive B-Bäume)