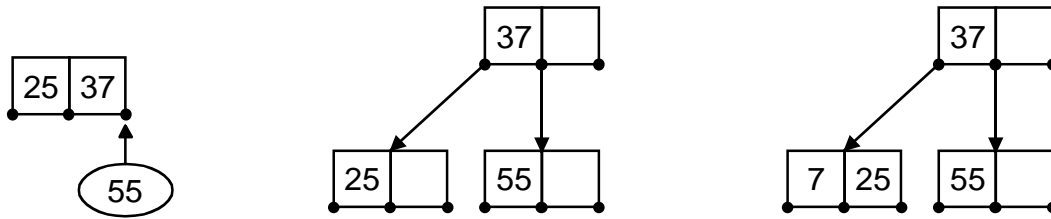
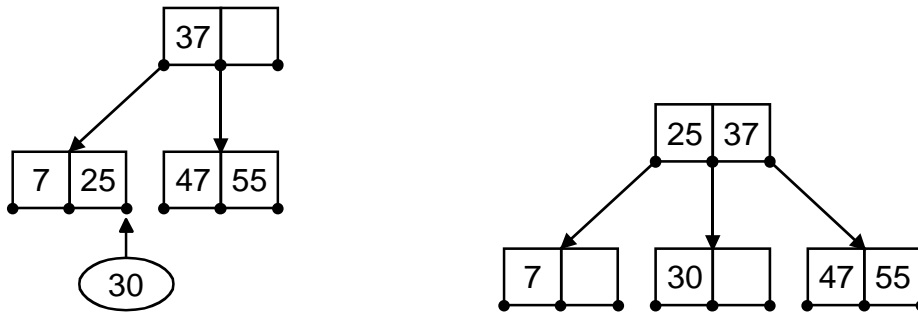


VL08, Lösung 1

a) Einfügen von 25, 37 (links), 55 (Mitte) und 7 (rechts):

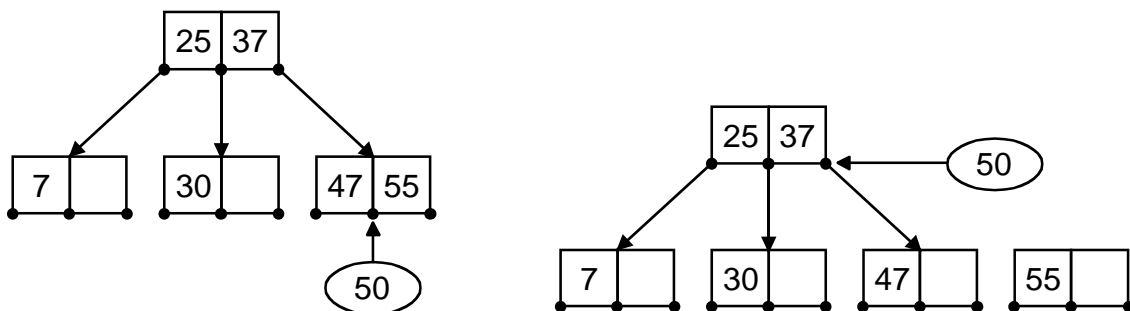


b) Einfügen von 47 (links), und 30 (rechts):

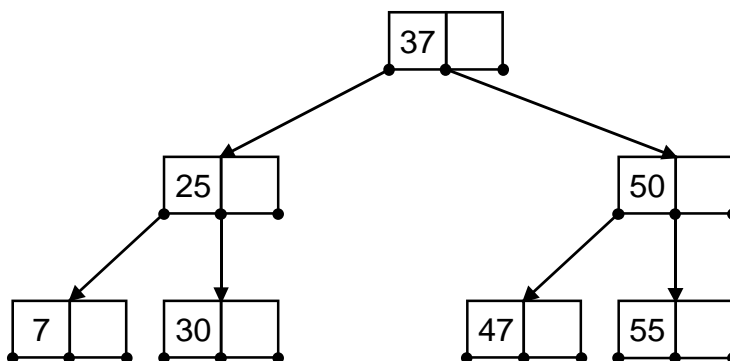


c) Einfügen von 50:

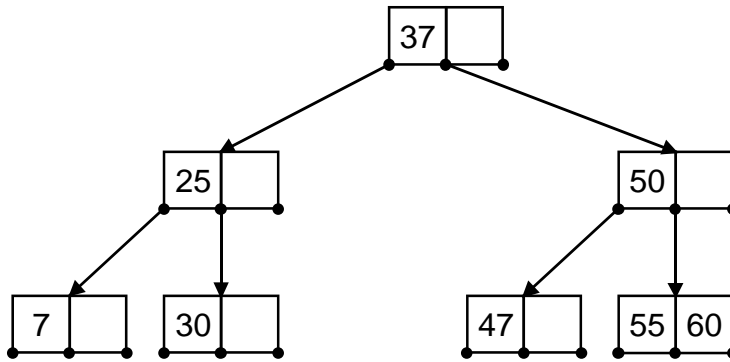
Zwischenzustand:



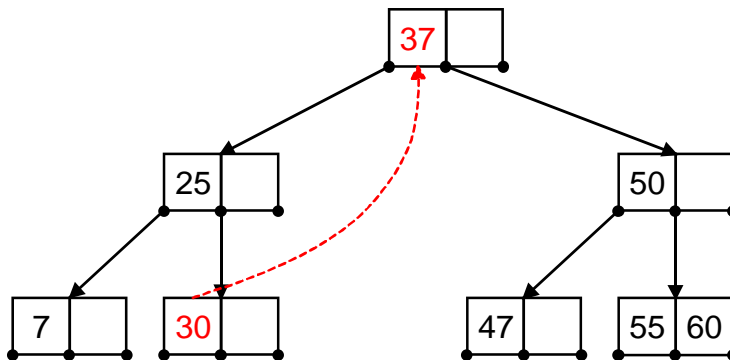
Endzustand:



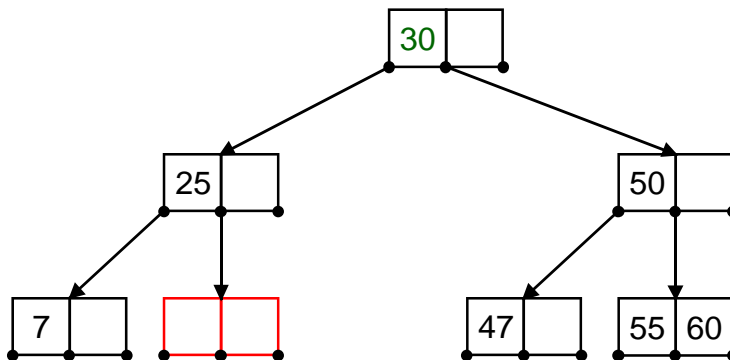
d) Einfügen von 60:



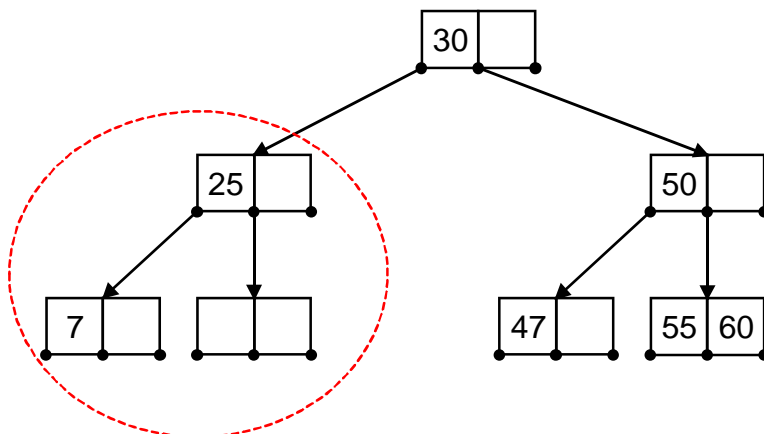
e) Löschen von 37:



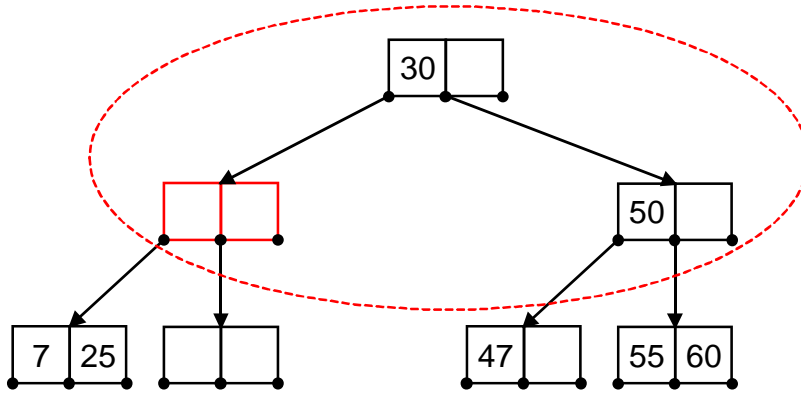
Zwischenzustand mit Unterlauf in Blatt:



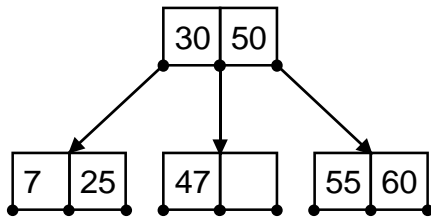
Verschmelzen mit Schlüssel des Elternknotens und linkem Nachbarn:



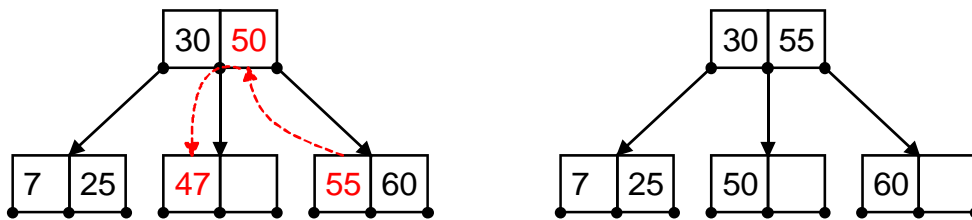
Zwischenzustand mit Unterlauf in innerem Knoten und Verschmelzen:



Endzustand:



- f) Löschen von 47, Beseitigung des Unterlaufs durch „Ausleihen“ des Schlüssels vom rechten Nachbarn:



VL08, Lösung 2

```
private void traversieren(BKnoten<T> knoten)
{
    // Leerer Teilbaum
    if (knoten == null)
        return;

    // Teilbaum ganz links traversieren
    traversieren(knoten.kinder[0]);

    for (int a=0; a < knoten.elemente.length; a++)
    {
        // Element ausgeben
        System.out.print(knoten.elemente[a] + " ");

        // Teilbaum rechts vom Element traversieren
        traversieren(knoten.kinder[a + 1]);
    }
}
```

```
private boolean suchen(final T daten, BKnoten<T> knoten)
{
    // Leerer Teilbaum
    if (knoten == null)
        return false;

    for (int a=0; a < knoten.elemente.length; a++)
    {
        // Gesuchtes Element mit Element im Knoten vergleichen
        final int cmp = daten.compareTo(knoten.elemente[a]);

        // Das Element wurde gefunden!
        if (cmp == 0)
            return true;

        // Das gesuchte Element ist kleiner als das Vergleichselement:
        // wir suchen im Teilbaum links vom Vergleichselement weiter.
        if (cmp < 0)
            return suchen(daten, knoten.kinder[a]);
    }

    // Im Teilbaum ganz rechts weitersuchen
    return suchen(daten, knoten.kinder[knoten.elemente.length]);
}
```