Grundzüge der Informatik 1

Vorlesung 17 - flipped classroom

Datenstrukturen

- **Aufgabe 1**
- Entwickeln Sie eine Datenstruktur, die die folgenden Operationen unterstützt
- Einfügen(x) in O(n) Zeit: Eine reelle Zahl x wird in die Datenstruktur eingefügt
- Löschen(x) in O(n) Zeit: Eine reelle Zahl x wird aus der Datenstruktur gelöscht
- Intervall(x,y) in O(log n) Zeit: Gibt die Anzahl der Zahlen im Intervall [x,y] zurück (für x<y)
- Ihre Datenstruktur soll O(n) Speicher benutzen, wobei n die aktuelle Anzahl an Zahlen in der Datenstruktur bezeichnet. Beschreiben Sie Ihre Datenstruktur und geben Sie Pseudocode für die Operationen Einfügen, Löschen und Intervall an.

Datenstrukturen

Beschreibung der Datenstruktur

- Wir nutzen als Datenstruktur ein sortiertes Feld. Beim Einfügen und Löschen wird das Feld neu aufgebaut.
- Idee für Intervall: Suche nach x und y in der Datenstruktur mit Hilfe von BinärerSuche und nutze die Differenz der Indizes

```
BinäreSuche(A,x,p,r)                 \* Finde Zahl x in sortiertem Feld A[p..r]
if p=r then return p                 \* sofern vorhanden
else                                 \* Ausgabe: Index der gesuchten Zahl
    q = ⌊(p+r)/2⌋
    if x ≤ A[q] then return BinäreSuche(A,x,p,q)
    else return BinäreSuche(A,x,q+1,r)
```

Datenstrukturen

Beschreibung der Datenstruktur

- Wir nutzen als Datenstruktur ein sortiertes Feld. Beim Einfügen und Löschen wird das Feld neu aufgebaut.
- Idee für Intervall: Suche nach x und y in der Datenstruktur mit Hilfe von BinärerSuche und nutze die Differenz der Indizes
- Seien i und j die Indizes die BinäreSuche(A,x,1,n) bzw. BinäreSuche(A,y,1,n) zurückgibt
- Ist A[i] = x und A[j] = y, dann ist Intervall(x,y) = i-j+1
- Ist A[i] >x und A[j] = y, dann ist Intervall(x,y) = i-j+1
- Ist A[i] = x und A[j]>y, dann ist Intervall(x,y) = i-j
- Ist A[i] >x und A[j]>y, dann ist Intervall(x,y) = i-j

Datenstrukturen

Intervall(A,x,y)

1. n = lenght[A]
2. i = BinäreSuche(A,x,1,n)
3. j = BinäreSuche(A,y,1,n)
4. Anz = i-j
5. **if** A[j] = y **then** Anz = Anz +1
6. **return** Anz

Datenstrukturen

Einfügen(A,x)

1. n = length[A]
2. B = **new array**[1..n+1]
3. i=1
4. **while** A[i] <x **do**
5.   B[i] = A[i]; i=i+1
6. B[i] = x; i=i+1
7. **while** i$\leq$ n+1**do**
8.   B[i] = A[i-1]; i=i+1
9. **delete** A
10. A=B

Datenstrukturen

Löschen(A,x) * Annahme: x ist in A vorhanden

```
n = length[A]
B = new array[1..n-1]
i=1
while A[i] <x do
   B[i] = A[i]; i=i+1
while i≤ n-1do
   B[i] = A[i+1]; i=i+1
delete A
A=B
```

Datenstrukturen

Rot-Schwarz-Bäume

- Balancierter Suchbaum
- Nach Einfügen/Löschen wird die Struktur des Suchbaums so modifiziert, dass eine Höhe von O(log n) garantiert wird
- Rebalancierung nach Einfügen/Löschen wird in O(log n) Zeit möglich sein
- Damit sind Operationen Suchen, Einfügen und Löschen in O(log n) Zeit möglich

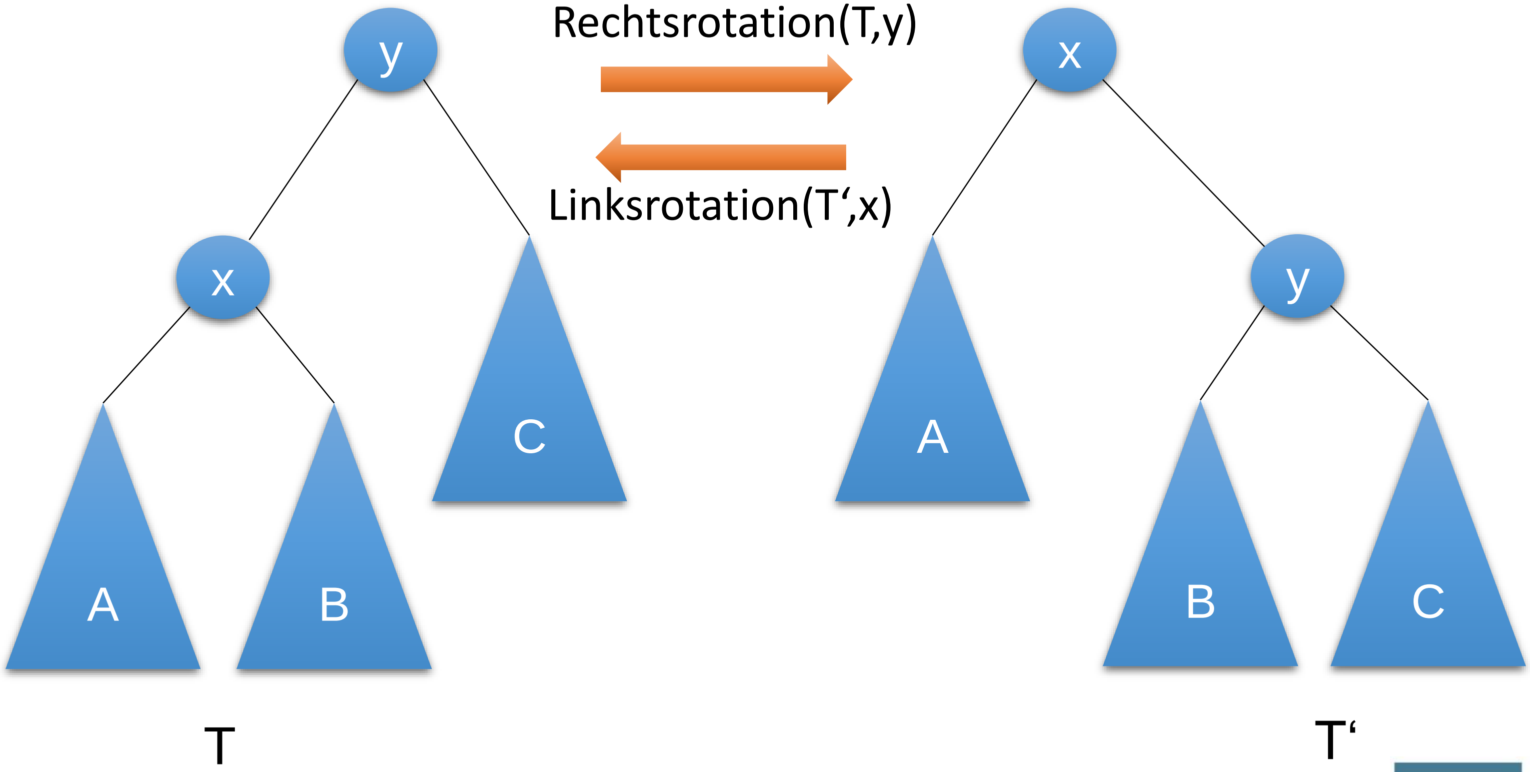
Datenstrukturen

Die Rot-Schwarz-Eigenschaften

- Jeder Knoten ist rot oder schwarz
- Die Wurzel ist schwarz
- Jedes Blatt ist schwarz
- Wenn ein Knoten rot ist, dann sind seine Kinder schwarz
- Für jeden Knoten v haben alle Pfade vom Knoten zu den Blättern im Unterbaum mit Wurzel v dieselbe Anzahl schwarzer Knoten

Datenstrukturen

Rotationen



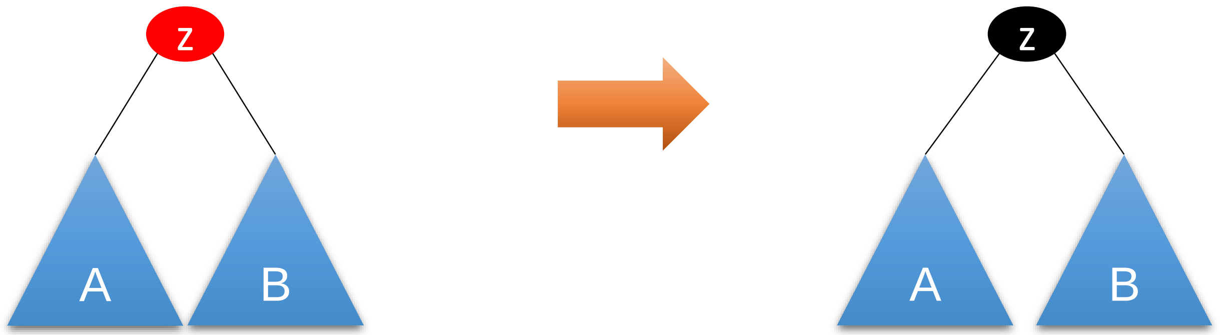
Datenstrukturen

Überblick: Wiederherstellen der Rot-Schwarz-Eigenschaften

- Starte mit eingefügtem Knoten z
- Stelle die Eigenschaft lokal wieder her, so dass sie nur von einem Knoten verletzt werden kann, der näher an der Wurzel ist
- Bei der Wurzel angekommen wird diese schwarz gefärbt

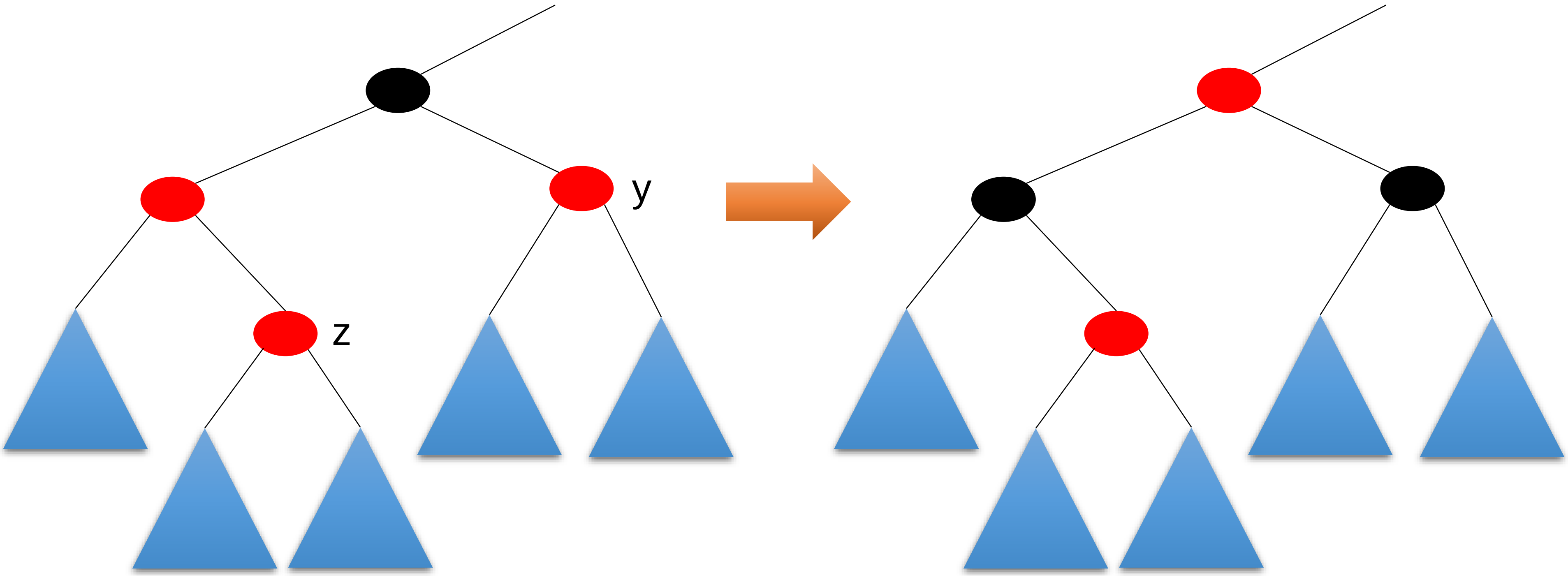
Datenstrukturen

Fall (1) (z ist die Wurzel)



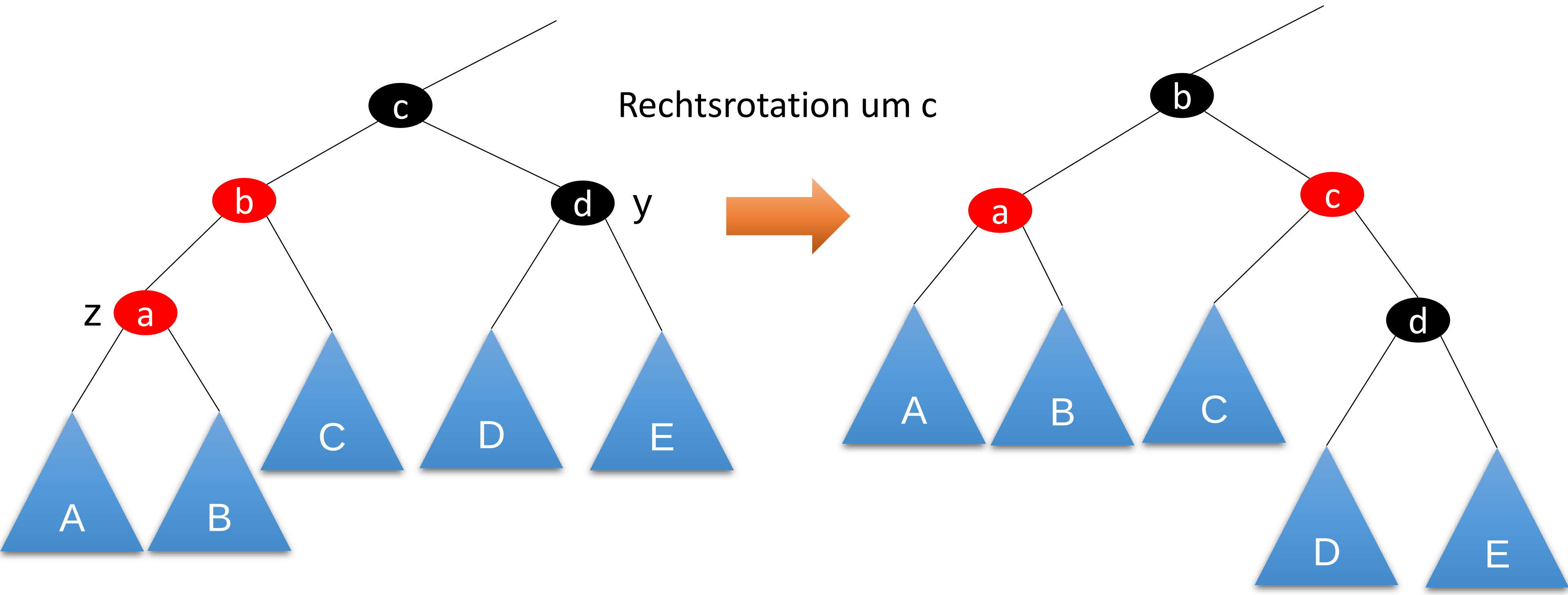
Datenstrukturen

Fall (2) (Onkel von z ist rot)



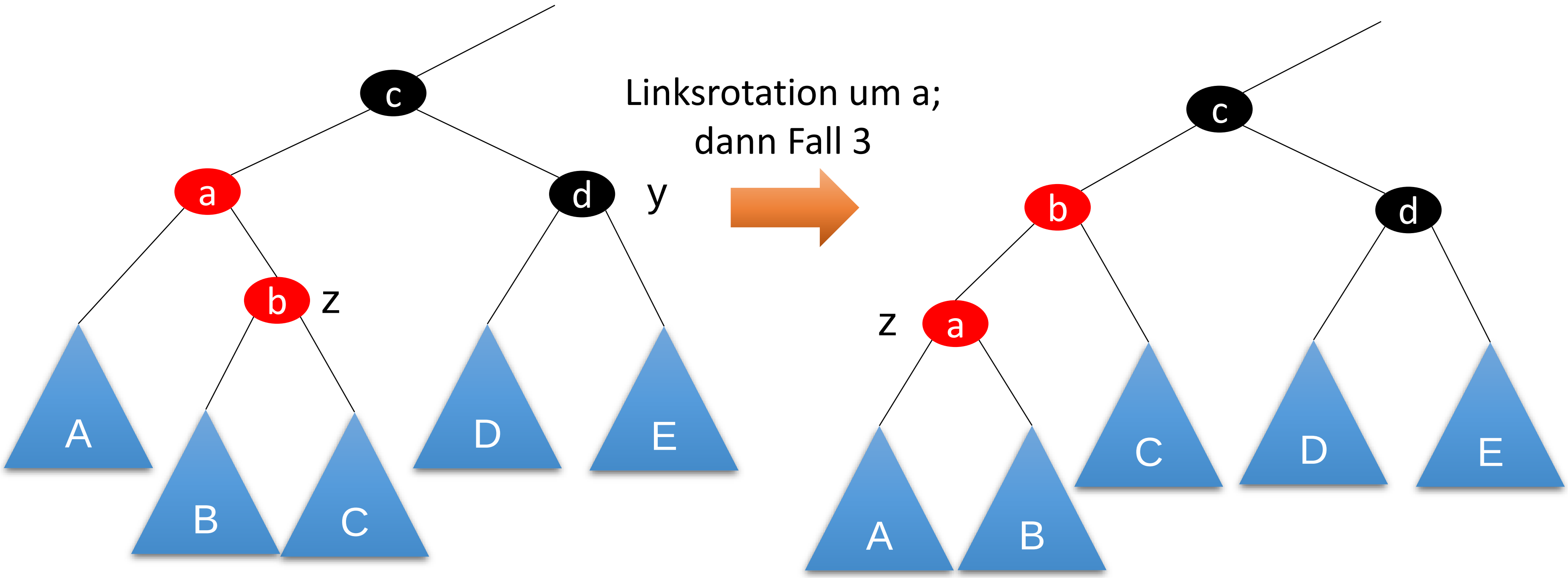
Datenstrukturen

Fall (3) (Onkel von z ist schwarz, z ist linkes Kind und parent(z) ist linkes Kind)



Datenstrukturen

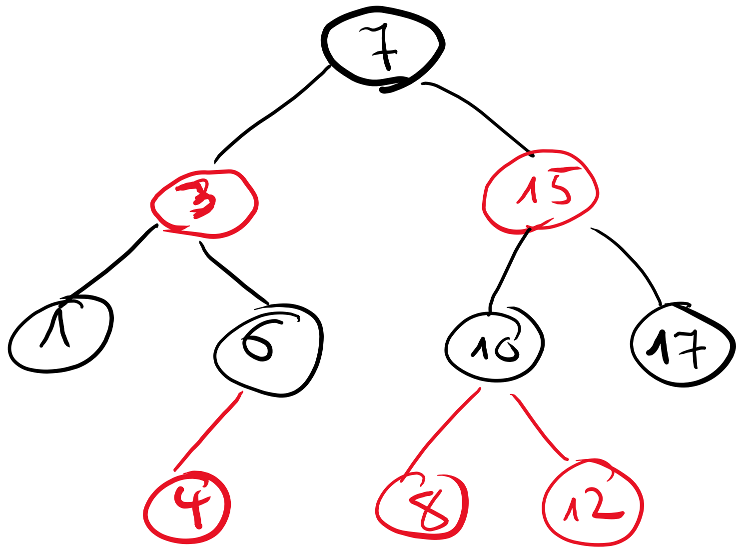
Fall (4) (Onkel von z ist schwarz, z ist rechtes Kind und parent(z) ist linkes Kind)



Datenstrukturen

Aufgabe 2

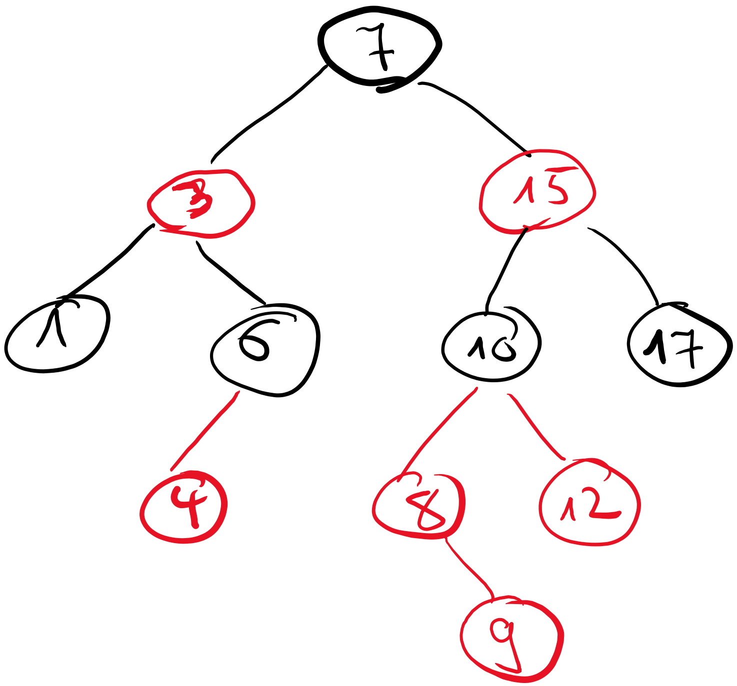
- Fügen Sie die Zahl 9 in den unten stehenden Rot-Schwarz-Baum ein



Datenstrukturen

Aufgabe 2

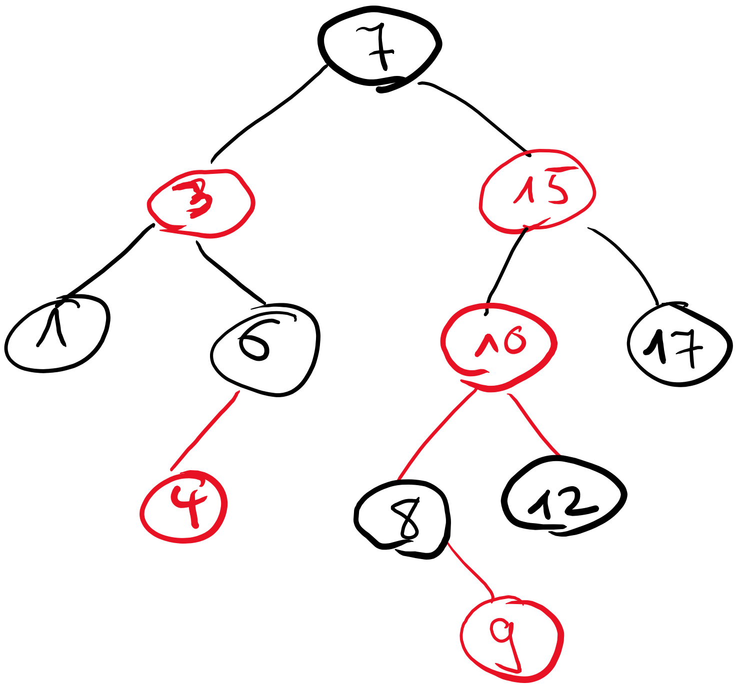
- Fügen Sie die Zahl 9 in den unten stehenden Rot-Schwarz-Baum ein



Datenstrukturen

Aufgabe 2

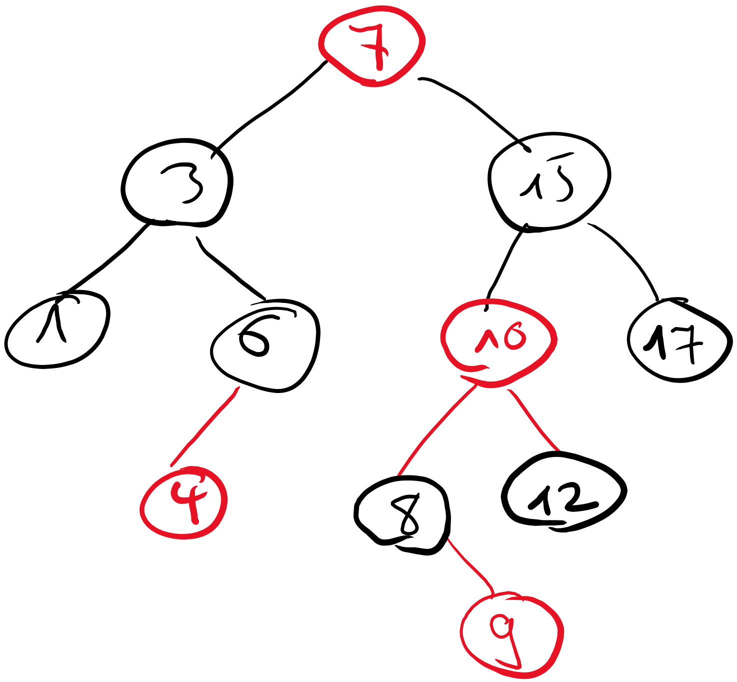
- Fügen Sie die Zahl 9 in den unten stehenden Rot-Schwarz-Baum ein



Datenstrukturen

Aufgabe 2

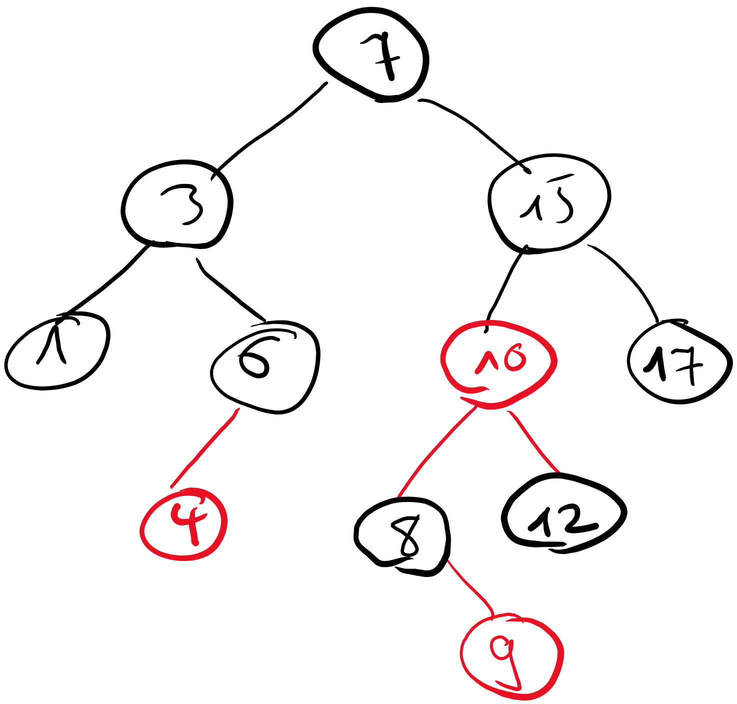
- Fügen Sie die Zahl 9 in den unten stehenden Rot-Schwarz-Baum ein



Datenstrukturen

Aufgabe 2

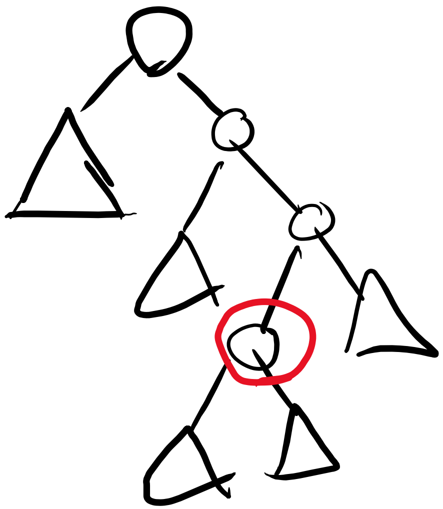
- Fügen Sie die Zahl 9 in den unten stehenden Rot-Schwarz-Baum ein



Datenstrukturen

Aufgabe 3

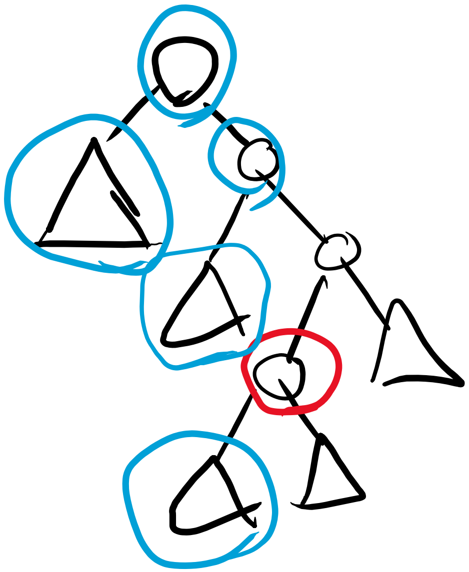
- Betrachten Sie folgenden Suchbaum. Welche Knoten bzw. Teilbäume sind kleiner als die Zahl, die am rot umkreisten Knoten steht? Sie können annehmen, dass keine Zahl im Suchbaum mehrfach vorkommt



Datenstrukturen

Aufgabe 3

- Betrachten Sie folgenden Suchbaum. Welche Knoten bzw. Teilbäume sind kleiner als die Zahl, die am rot umkreisten Knoten steht? Sie können annehmen, dass keine Zahl im Suchbaum mehrfach vorkommt



Datenstrukturen

Aufgabe 4

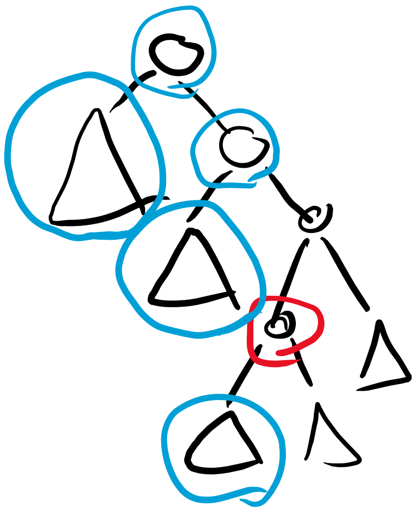
- Welches zusätzliche Attribut sollte in den Knoten eines Suchbaums vorhanden sein, damit man die Operation
- AnzahlKleinerAls(x,k), die die Anzahl der Zahlen im Suchbaum mit Wurzel x angibt, die kleiner als k sind, in O(h) Zeit bestimmen kann, wobei h die Höhe des Suchbaums ist?
- Geben Sie Pseudocode für AnzahlKleinerAls(k) an, unter der Annahme, dass das Attribut vorhanden ist.

Universität zu Köln

Datenstukturen

Idee

- Wenn wir für jeden Unterbaum seine Größe kennen würden, dann wäre der Rang gerade die Summe dieser Größen plus 1 plus die Anzahl Zahlen auf dem Suchpfad, die kleiner sind als x
- Wir benötigen also das Attribut Größe



Datenstrukturen

AnzahlKleinerAls(x,k)

```
if x=NIL or (k=key[x] and left[x]=NIL) then
   return 0
else
   if k=key[x] then return Größe[left[x]]
if k<key[x] then
   return AnzahlKleinerAls(left[x],k)
else
   if left[x]=NIL then
      return 1+AnzahlKleinerAls(right[x],k)
   else return 1+ Größe[left[x]] + AnzahlKleinerAls(right[x],k)
```

Datenstrukturen

Aufgabe 5

- Modifizieren Sie den Rot-Schwarz-Baum so, dass an den Knoten auch das Attribut Größe (Anzahl der Knoten im Unterbaum) aus der letzten Übung aufrecht erhalten werden kann
- Identifizieren Sie dazu zunächst die Schritte, in denen der Pseudocode geändert werden muss
- Geben Sie die Modifikationen der Prozeduren für Rotationen und Einfügen im Pseudocode an

Datenstrukturen

Aufgabe 5

- Modifizieren Sie den Rot-Schwarz-Baum so, dass an den Knoten auch das Attribut Größe (Anzahl der Knoten im Unterbaum) aus der letzten Übung aufrecht erhalten werden kann
- Identifizieren Sie dazu zunächst die Schritte, in denen der Pseudocode geändert werden muss

(a) Rotationen

(b) Einfügen

(c) Löschen

- Geben Sie die Modifikationen der Prozeduren für Rotationen und Einfügen im Pseudocode an

Datenstrukturen

Rotationen