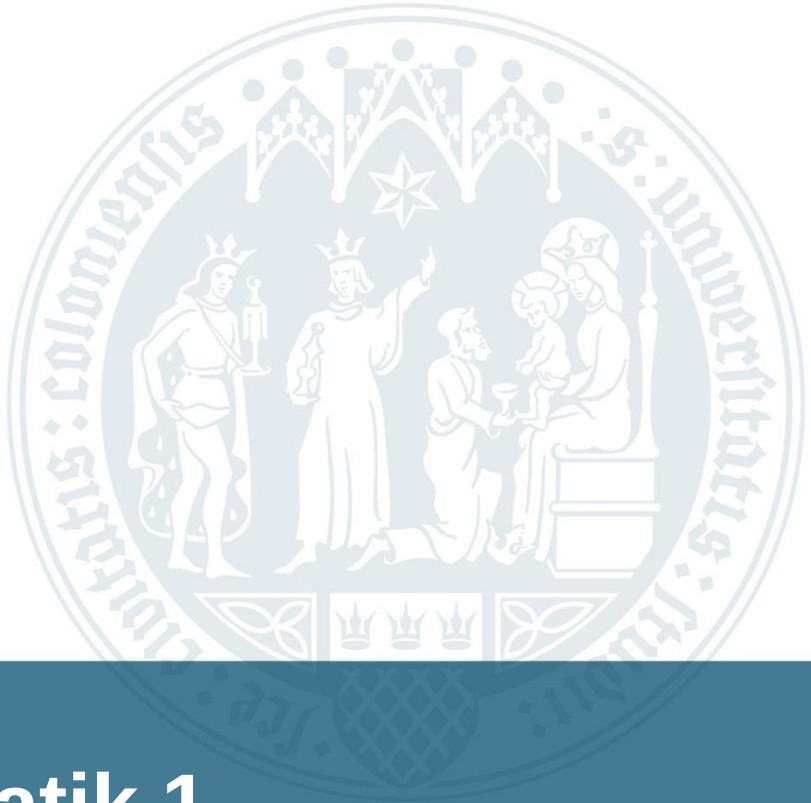
Universität zu Köln



Grundzüge der Informatik 1

Vorlesung 16 - flipped classroom

Gierige Algorithmen

Erinnerung

- Bei der Vorlesungsplanung müssen ALLE Vorlesungen auf Hörsäle abgebildet werden, so dass sich keine zwei Vorlesungen überschneiden. Wir gehen davon aus, dass jede Vorlesung in jedem Hörsaal abgehalten werden kann. Gesucht ist eine Zuordnung der Vorlesungen zu den Hörsälen, die die Anzahl der genutzten Hörsäle minimiert.

Gierige Algorithmen

Hörsäle(S,E,n)

1. new array Belegung[1..n]
2. maxBedarf = BerechneBedarf(S,E,n)
3. Sortiere S nach Startzeiten
4. **for** i=1 **to** n **do**
5. Wähle einen Hörsaal h aus {1,…,maxBedarf}, der zum Zeitpunkt S[i] nicht belegt ist
6. Belegung[i] = h
7. **return** Belegung

Gierige Algorithmen

Aufgabe 1

- Kann man den Algorithmus Hörsäle in O(n log n) Laufzeit implementieren?

Gierige Algorithmen

Implementierung

- Die Frage ist, wie man Zeile 5 implementieren kann.
- Wir werden ein Feld Frei[1..maxBedarf] verwenden, in dem die freien Hörsäle gespeichert sind und eine Variable x, die die Anzahl der freien Hörsäle abspeichert
- Wir nehmen an, die Hörsäle sind von 1 bis maxBedarf durchnummeriert
- Wenn es x freie Hörsäle gibt, so sind deren Nummern in Frei[i..x] gespeichert
- Wird nun ein Hörsaal für Vorlesung i gesucht, so wird diese in Frei[x] durchgeführt, Frei[x] wird auf 0 gesetzt und x um 1 verringert
- Zum Endzeitpunkt einer Vorlesung wird x um 1 erhöht und der Hörsaal in Frei[x] gespeichert (dazu wird der Hörsaal mit der Vorlesung abgespeichert)

Gierige Algorithmen

Verbund Data:
Nummer
Zeitpunkt
StartEnd

Hörsäle(S,E,n)

1. Belegung = new array [1..n]
2. maxBedarf = BerechneBedarf(S,E,n)
3. Frei = new array [1..maxBedarf]; x = maxBedarf
4. **for** i=1 **to** maxBedarf **do** Frei[i] =i
5. Sei F[1..2n] ein Feld über Verbundtyp Data
6. **for** i=1 **to** n **do**
7. F[i] = (i, S[i], 0); F[n+i] = (i, E[i], 1)
8. Sortiere F nach Zeitpunkten
9. **for** i=1 **to** 2n **do**
10. **if** F[i].StartEnd = 0 **then** Belegung[F[i].Nummer] = Frei[x]; Frei[x]=0; x=x-1
11. **if** F[i].StartEnd = 1 **then** x=x+1; Frei[x] = Belegung[F[i].Nummer]
12. **return** Belegung

Datenstrukturen

Aufgabe 2

- In einen anfangs leeren binären Suchbaum werden die Zahlen 3,7,1,15,8,19,18 in dieser Reihenfolge eingefügt
- Zeichnen Sie den Baum nach jeder Einfügeoperation

Datenstrukturen

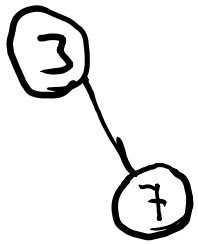
Aufgabe 2

- In einen anfangs leeren binären Suchbaum werden die Zahlen 3,7,1,15,8,19,18 in dieser Reihenfolge eingefügt
- Zeichnen Sie den Baum nach jeder Einfügeoperation

Datenstrukturen

Aufgabe 2

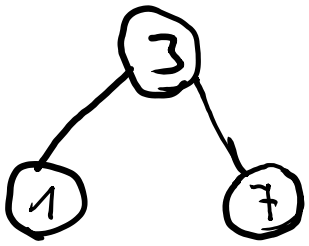
- In einen anfangs leeren binären Suchbaum werden die Zahlen 3,7,1,15,8,19,18 in dieser Reihenfolge eingefügt
- Zeichnen Sie den Baum nach jeder Einfügeoperation



Datenstrukturen

Aufgabe 2

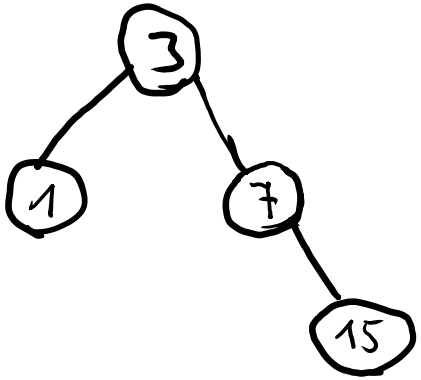
- In einen anfangs leeren binären Suchbaum werden die Zahlen 3,7,1,15,8,19,18 in dieser Reihenfolge eingefügt
- Zeichnen Sie den Baum nach jeder Einfügeoperation



Datenstrukturen

Aufgabe 2

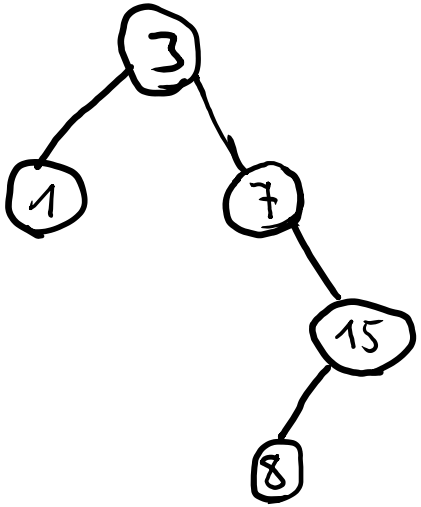
- In einen anfangs leeren binären Suchbaum werden die Zahlen 3,7,1,15,8,19,18 in dieser Reihenfolge eingefügt
- Zeichnen Sie den Baum nach jeder Einfügeoperation



Datenstrukturen

Aufgabe 2

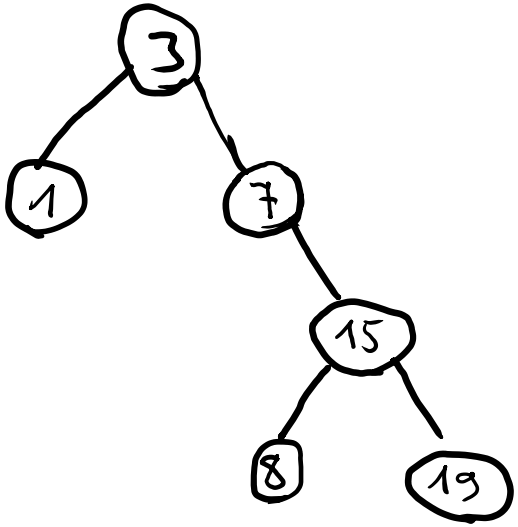
- In einen anfangs leeren binären Suchbaum werden die Zahlen 3,7,1,15,8,19,18 in dieser Reihenfolge eingefügt
- Zeichnen Sie den Baum nach jeder Einfügeoperation



Datenstrukturen

Aufgabe 2

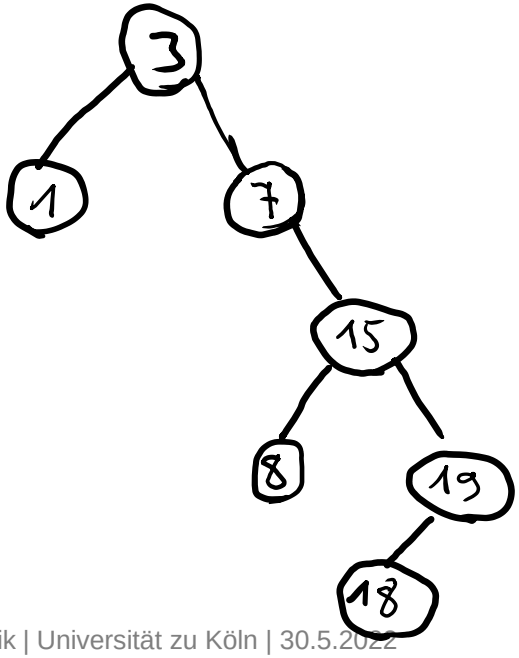
- In einen anfangs leeren binären Suchbaum werden die Zahlen 3,7,1,15,8,19,18 in dieser Reihenfolge eingefügt
- Zeichnen Sie den Baum nach jeder Einfügeoperation



Datenstrukturen

Aufgabe 2

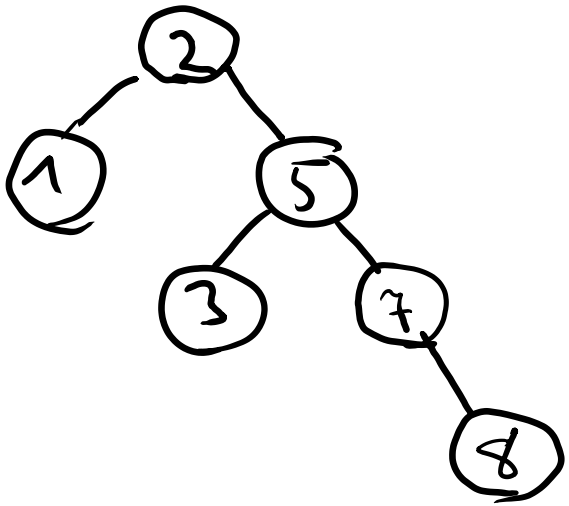
- In einen anfangs leeren binären Suchbaum werden die Zahlen 3,7,1,15,8,19,18 in dieser Reihenfolge eingefügt
- Zeichnen Sie den Baum nach jeder Einfügeoperation



Datenstrukturen

Aufgabe 3

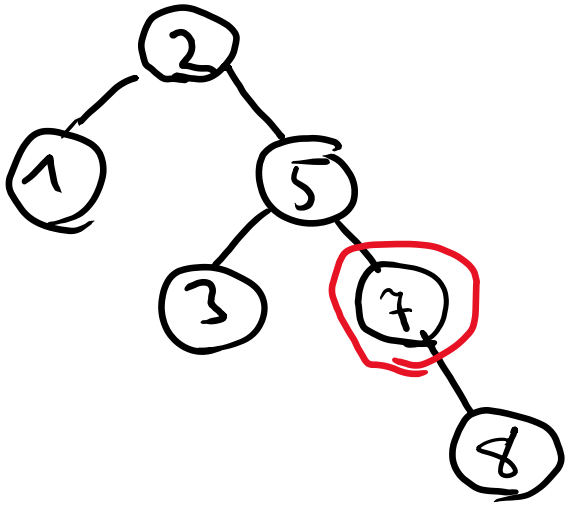
- Aus unten stehendem binären Suchbaum sollen die Zahlen 5, 7 und 8 in dieser Reihenfolge entfernt werden
- Zeichnen Sie den Baum nach jeder Löschoperation



Datenstrukturen

Aufgabe 3

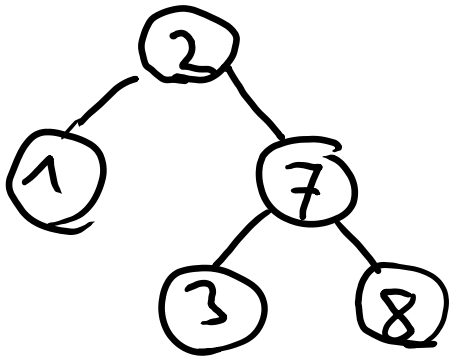
- Aus unten stehendem binären Suchbaum sollen die Zahlen 5, 7 und 8 in dieser Reihenfolge entfernt werden
- Zeichnen Sie den Baum nach jeder Löschoperation



Datenstrukturen

Aufgabe 3

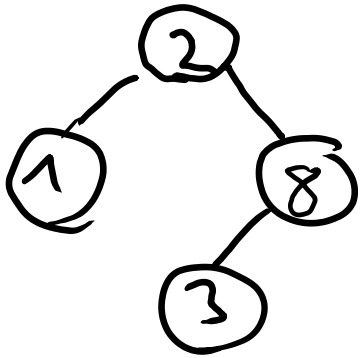
- Aus unten stehendem binären Suchbaum sollen die Zahlen 5, 7 und 8 in dieser Reihenfolge entfernt werden
- Zeichnen Sie den Baum nach jeder Löschoperation



Datenstrukturen

Aufgabe 3

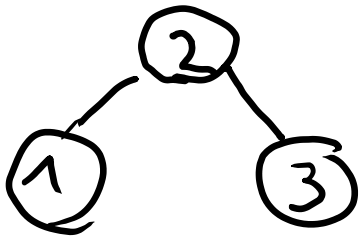
- Aus unten stehendem binären Suchbaum sollen die Zahlen 5, 7 und 8 in dieser Reihenfolge entfernt werden
- Zeichnen Sie den Baum nach jeder Löschoperation



Datenstrukturen

Aufgabe 3

- Aus unten stehendem binären Suchbaum sollen die Zahlen 5, 7 und 8 in dieser Reihenfolge entfernt werden
- Zeichnen Sie den Baum nach jeder Löschoperation



Datenstrukturen

- **Aufgabe 4**
- 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]
1.  if p=r then return p                \* sofern vorhanden
2.  else                                \* Ausgabe: Index der gesuchten Zahl
3.      q = ⌊(p+r)/2⌋
4.      if x ≤ A[q] then return BinäreSuche(A,x,p,q)
5.      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≤ 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

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. **while** i≤ n-1**do**
7. B[i] = A[i+1]; i=i+1
8. **delete** A
9. A=B