2 Geordnete Binärbäume

= binäre Suchbäume

= BST (binary search tree)

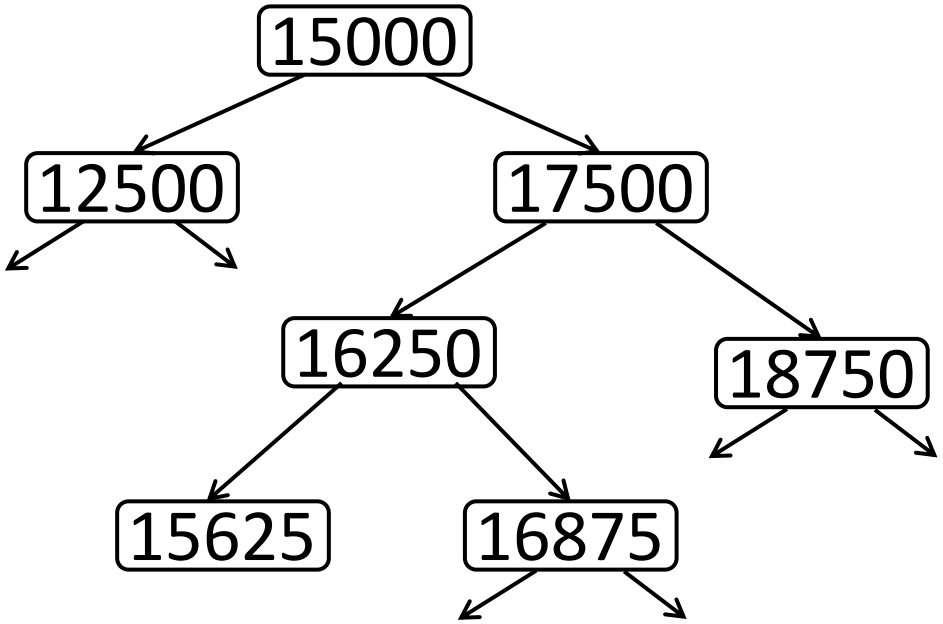
S. 62 / Einführende Aufgabe

Denke dir eine Zahl zwischen 10000 und 20000.
Lass sie von deinem Nachbarn durch Stellen von Ja-Nein-Fragen erraten.

Stelle die Situation (zumindest ansatzweise) in einem Baumdiagramm dar.

Wie viele Fragen sind maximal nötig?

Suche 16753:



Mit jeder Frage halbiert sich die Intervalllänge:

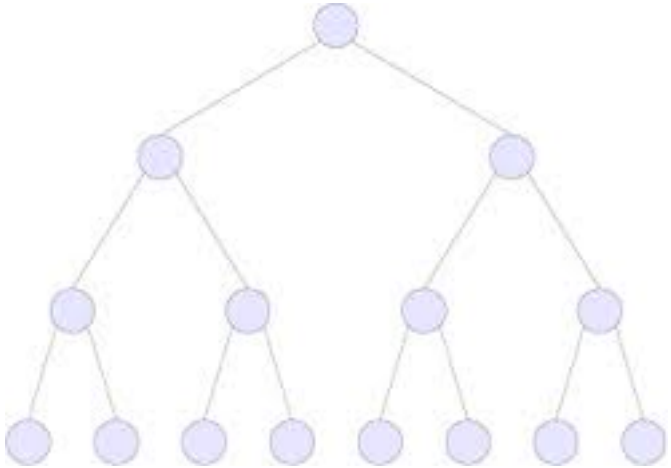
Anzahl Fragen	0	1	2	3	4	5	6
Intervalllänge	10000	5000	2500	1250	625	313	157

7	8	9	10	11	12	13	14
79	40	20	10	5	3	2	1

Oder: $2^{13} < 10000 < 2^{14}$

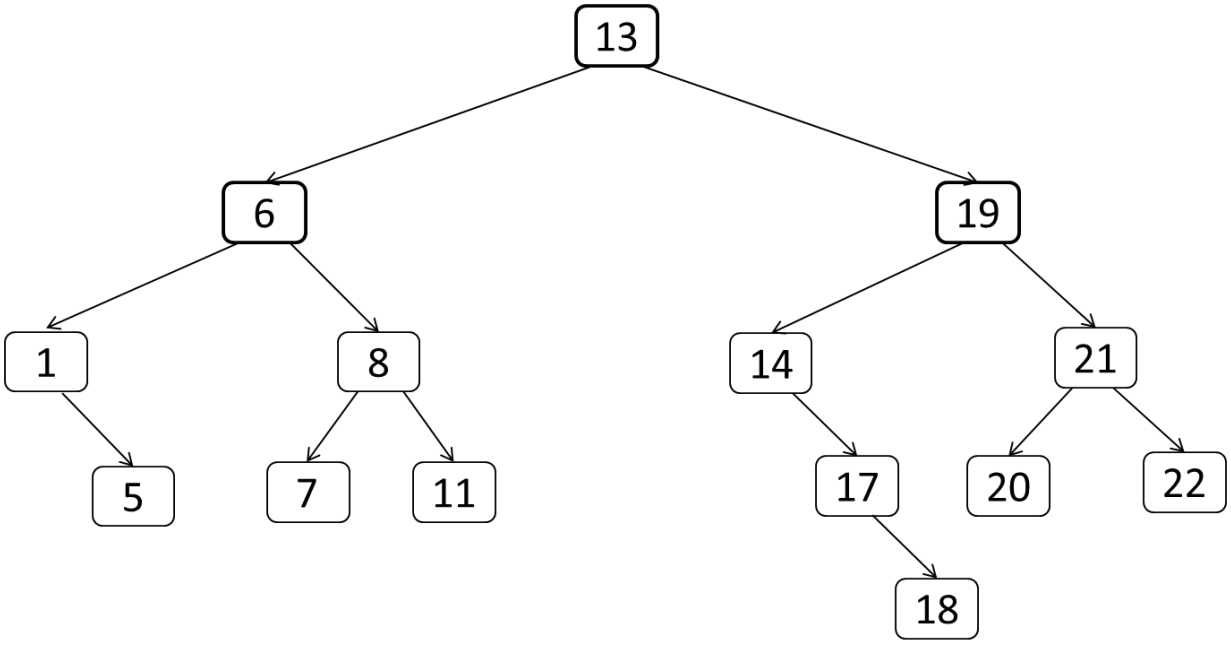
Wie viele Datenknoten bringt man unter in einem Binärbaum mit Ebenenzahl

1:	1	$=2^1 -1$
2:	1+2=3	$=2^2 -1$
3:	1+2+4=7	$=2^3 -1$
4:	1+2+4+8=15	$=2^4 -1$
n:	$1+2+4+\ldots+2^{n-1}$	$=2^n -1$



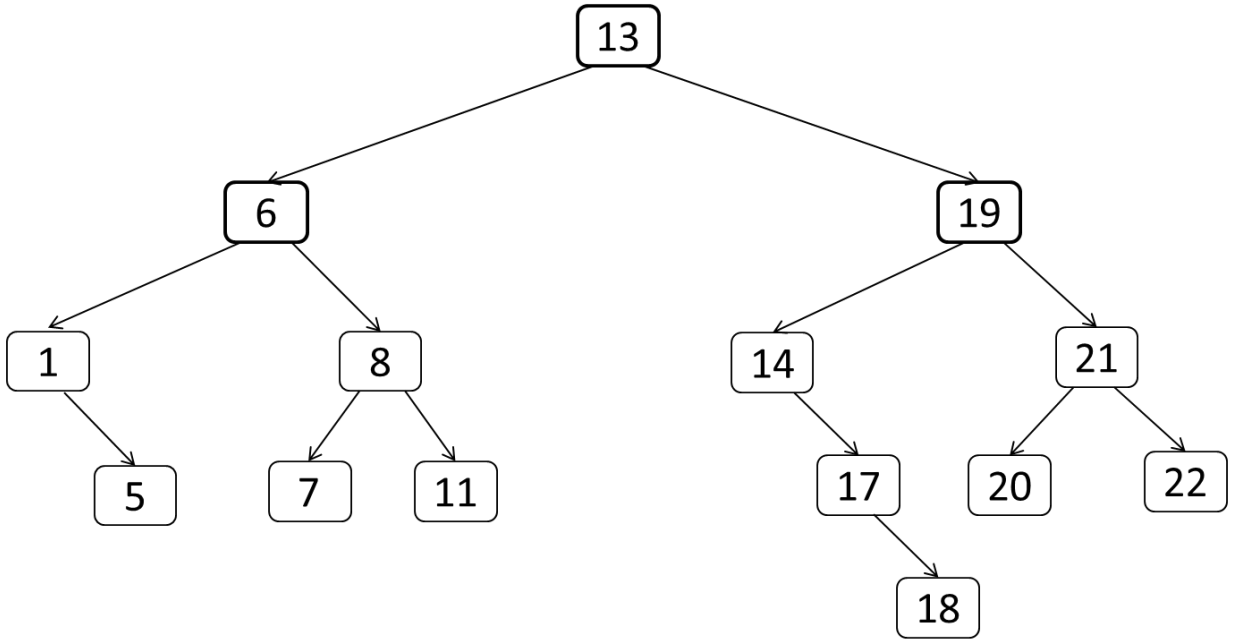
Geordneter Binärbaum:

Ein Binärbaum heißt geordnet, wenn die Knoten des linken Teilbaums eines Knotens nur kleinere Schlüssel und die Knoten des rechten Teilbaums eines Knotens nur größere Schlüssel als der Knoten selbst besitzen.



Suche nach einem Datenelement mit einem bestimmten Schlüsselwert:

Suche 17



Strategie:

Vergleiche den Schlüsselwert s_{such} des gesuchten Elements mit dem der Wurzel s_w:

- $s_w = s_{such}$

  Wurzel enthält das gesuchte Datenelement und gibt es zurück.

- $s_w < s_{such}$

  Suche im rechten Teilbaum weiter.

- $s_w > s_{such}$

  Suche im linken Teilbaum weiter.

S. 65/1 Kundenverwaltung

Öffne S65-A1-Vorlage. Lies dazu S. 62 (Mitte).

- Ergänze in den Klassen BST und Datenknoten Attribute und Konstruktor.
- Betrachte die in der Testklasse vorgegebene Baumstruktur. Wer ist Wurzel, wer ist Blatt?
- Implementiere in der Klasse Datenknoten die Methode datenGeben(). Welche Traversierung sorgt für eine entsprechend des Schlüsselwerts geordnete Ausgabe? Ergänze dazu nötige Methoden.
- Zeichne die Baumstruktur auf.
- Implementiere eine Methode
  public Kunde kundeSuchen(int nr) in der Klasse Kundenverwaltung und die dazu nötigen Methoden in den anderen Klassen.

In BST:

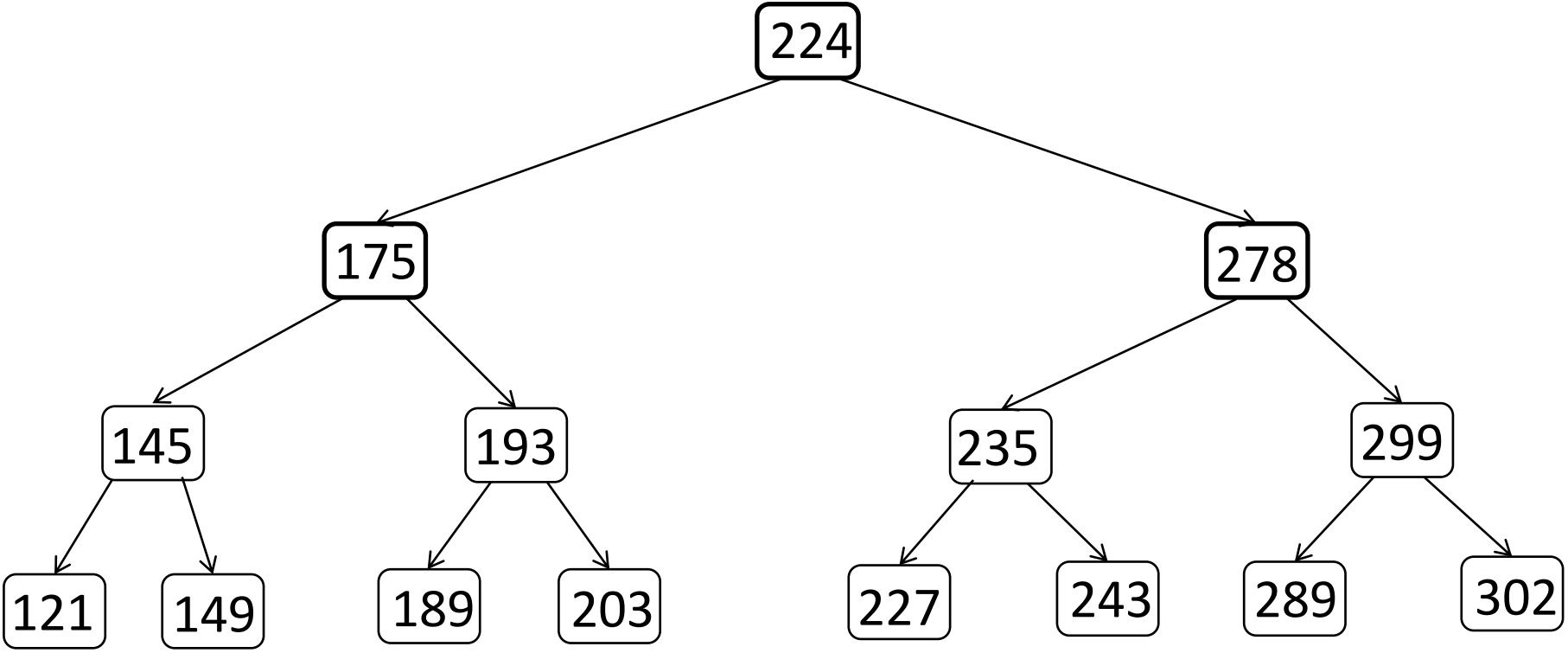
```
private Baumelement wurzel;

public BST(){
    wurzel = new Abschluss();
}
```

In Datenknoten:

```
private Baumelement naechsterLinks, naechsterRechts;
private Datenelement inhalt;

public Datenknoten(Baumelement nL, Baumelement nR,
                                    Datenelement inh){
    naechsterLinks = nL;
    naechsterRechts = nR;
    inhalt = inh;
}
```



121: Prinz
145: Freud
149: Bohr
175: Meitner
189: Zuse
193: Mandelbrot
203: Neumann

224: Sierpinski
227: Dijkstra
235: Hoare
243: Ulam
278: Chomsky
289: Turing
299: Landau
302: Moore

In Datenknoten:

```
//Inorder

 public String alleDatenGeben(){
        return naechsterLinks.alleDatenGeben()
                + inhalt.datenGeben() + "\n"
                + naechsterRechts.alleDatenGeben();
   }
```

In Datenelement:

```
public abstract String alleDatenGeben();
```

In Kunde:

```
 public String datenGeben(){
       return kundennr +": "+ name +", " +vorname;
    }
```

In Kundenverwaltung:

```
public Kunde kundeSuchen(int nr){
      return (Kunde) bst.inhaltSuchen(new Kunde(nr, "", ""));
   }
```

In BST:

```
  public Datenelement inhaltSuchen(Datenelement de){
        return wurzel.inhaltSuchen(de);
    }
```

In Baumelement:

```
public abstract Datenelement inhaltSuchen(Datenelement de);
```

In *Datenelement*:

```
public abstract boolean istGleich(Datenelement de);

public abstract boolean istKleiner(Datenelement de);
```

In Kunde:

```
public boolean istGleich(Datenelement de){
    return kundennr == ((Kunde) de).kundennr;
}

public boolean istKleiner(Datenelement de){
    return kundennr < ((Kunde) de).kundennr;
}
```

In Datenknoten:

```
public Datenelement inhaltSuchen(Datenelement de){
        if (inhalt.istGleich(de)){
            return inhalt;
       }
       else {
           if (inhalt.istKleiner(de)){
               return naechsterRechts.inhaltSuchen(de);
            }
            else {
                return naechsterLinks.inhaltSuchen(de);
            }
        }
    }
```

In Abschluss:

```
public Datenelement inhaltSuchen(Datenelement de){
    return null;
  }
```