Was ist die größte Zahl in einem Array?

```
int a[ ];
```

```
public int maximum() {
   int n = a.length;
   if(n == 0) {
      return 0;
   }
   int max = a[0];
   for(int i = 1; i < n; i++) {
      if(a[i] > max) {
         max = a[i];
      }
   }
   return max;
}
```

Einfacher Sortieralgorithmus

Wir sortieren ein Array $a[\,]$.

```
public void sort() {
  int n = a.length; // Länge von a
  for(int j = 0; j < n; j++) {
    for(int k = 0; k < n - 1; k++) {
      if(a[k] > a[k + 1]) { // Falsche Reihenfolge
        // Vertausche a[k] mit a[k+1]
        int temp = a[k];
        a[k] = a[k + 1];
        a[k + 1] = temp;
      }
    }
  }
}
```

Neues Kapitel

Überblick

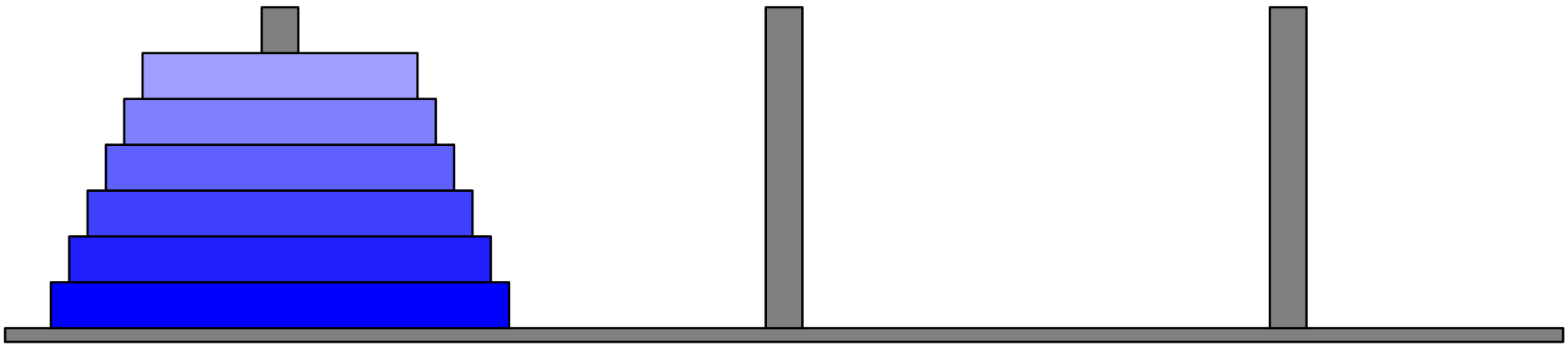
2. Rekursion
   - Rekursive Methoden
   - Backtracking
   - Memorization
   - Einfache rekursive Datenstrukturen
   - Bäume
   - Aufzählen, Untermengen, Permutationen, Bitmengen

Fibonacci-Zahlen

- $F_0 = 0$,
- $F_1 = 1$,
- $F_n = F_{n-1} + F_{n-2}$ falls $n > 1$.

```
int fibo(int n) {
  if(n == 0) {
    return 0;
  }
  else if(n == 1) {
    return 1;
  }
  else {
    return fibo(n - 1) + fibo(n - 2);
  }
}
```

Die Türme von Hanoi



Bewege die Scheiben von der linken Stange auf die rechte Stange.

- Nur je eine Scheibe darf bewegt werden.
- Eine Scheibe darf nicht auf einer kleineren liegen.

Die Türme von Hanoi

Bewege *n* Scheiben von *sourcePeg* nach *destPeg*.

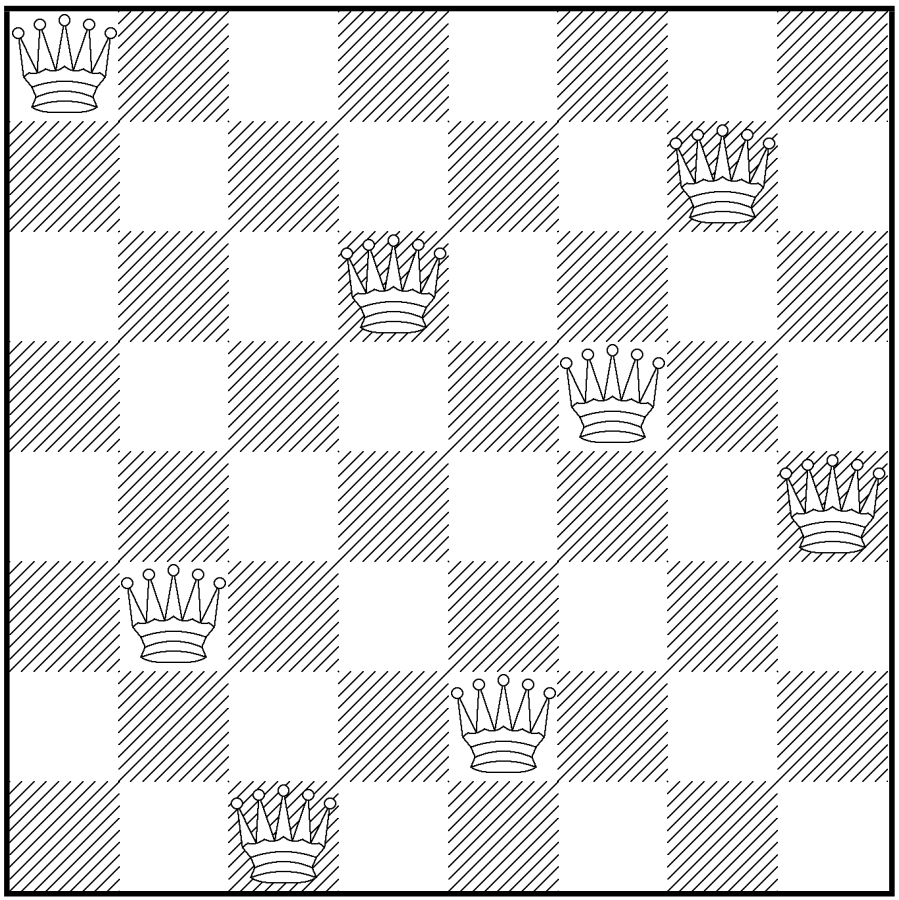
Benutze *thirdPeg* als Ablage.

```
static String move(int n, int sourcePeg, int destPeg, int thirdPeg) {
  if(n == 0) {
    return "";
  }
  String phase1 = move(n - 1, sourcePeg, thirdPeg, destPeg);
  String phase2 = "  " + sourcePeg + "->" + destPeg;
  String phase3 = move(n - 1, thirdPeg, destPeg, sourcePeg);
  return phase1 + phase2 + phase3;
}
```

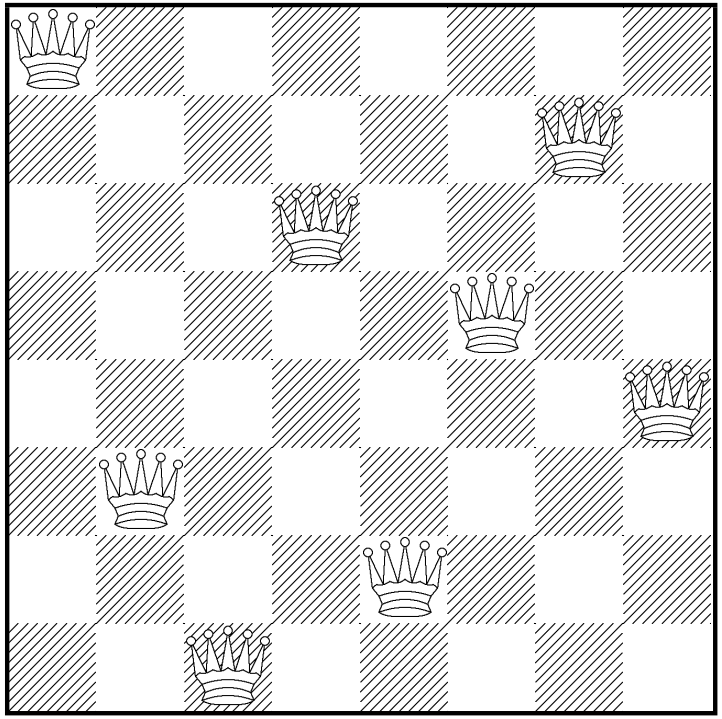
Überblick

2. Rekursion
   - Rekursive Methoden
   - Backtracking
   - Memorization
   - Einfache rekursive Datenstrukturen
   - Bäume
   - Aufzählen, Untermengen, Permutationen, Bitmengen

Das Acht-Damen-Problem



Backtracking



Wir verwenden *Backtracking*:

Setze Damen in Zeilen von links nach rechts.

Wenn nächstes Setzen unmöglich, nimm letzten Zug zurück und versuche nächsten.

```
void setzeDamenAbZeile(int y) {
  if(y >= n) {
    geloest = true;
    return;
  }
  for(int x = 0; x < n; x++) {
    if(safePosition(y, x)) {
      brett[y][x] = true;
      setzeDamenAbZeile(y + 1);
      if(geloest) {
        return;
      }
      brett[y][x] = false;
    }
  }
}
```