Informatik II

Übung 5

Giuseppe Accaputo, Felix Friedrich, Patrick Gruntz, Tobias Klenze, Max Rosmannek, David Sidler, Thilo Weghorn

FS 2017

Heutiges Programm

1. Zur Money Aufgabe
2. Java Exceptions
3. Sortieren

Macht der Rekursion

Wie viele Möglichkeiten der Aufteilung von n in gültige Teilbeträge $1000, 200, 100, 50, 20, 10, 5, 2, 1$.

Macht der Rekursion

Wie viele Möglichkeiten der Aufteilung von n in gültige Teilbeträge $1000, 200, 100, 50, 20, 10, 5, 2, 1$.

- 1. Idee: Wenn ich das Problem lösen kann für alle $k < n$, dann betrachte Unterteilungen $|n-1, 1|, |n-2, 2|, |n-3, 3|, \ldots, |n/2, n/2|$ und summiere alle Lösungen (allenfalls plus der Lösung mit gültigem Teilbetrag n)

Macht der Rekursion

Wie viele Möglichkeiten der Aufteilung von n in gültige Teilbeträge $1000, 200, 100, 50, 20, 10, 5, 2, 1$.

- 1. Idee: Wenn ich das Problem lösen kann für alle $k < n$, dann betrachte Unterteilungen $|n-1, 1|, |n-2, 2|, |n-3, 3|, \ldots, |n/2, n/2|$ und summiere alle Lösungen (allenfalls plus der Lösung mit gültigem Teilbetrag n)
- Das scheitert wegen der unkontrollierbaren Permutationen. Lösung $(2+2+1+2+2+1)$ entspricht der Lösung $(2+2)+(1+1+2+2)$.

Macht der Rekursion

Wie viele Möglichkeiten der Aufteilung von n in gültige Teilbeträge $1000, 200, 100, 50, 20, 10, 5, 2, 1$.

- 2. Idee: Wenn ich das Problem lösen kann für alle $k < n$, dann betrachte Unterteilungen der Form $k, n - k$, wobei k ein gültiger Teilbetrag sein muss.

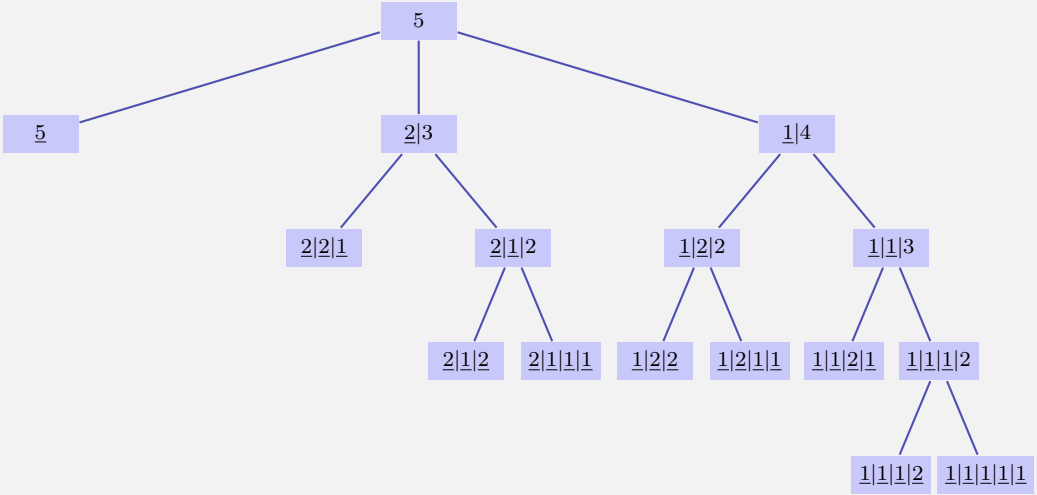
Macht der Rekursion

Wie viele Möglichkeiten der Aufteilung von n in gültige Teilbeträge $1000, 200, 100, 50, 20, 10, 5, 2, 1$.

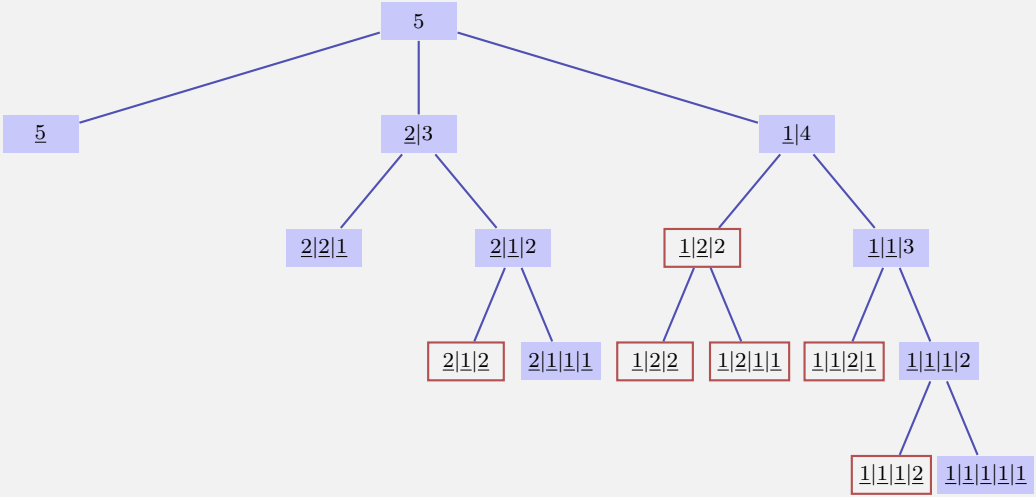
- 2. Idee: Wenn ich das Problem lösen kann für alle $k < n$, dann betrachte Unterteilungen der Form $k, n - k$, wobei k ein gültiger Teilbetrag sein muss.
- Das scheitert zunächst auch wegen zu vieler Permutationen. Lösung $(2 + (2 + 1))$ entspricht der Lösung $(1 + (2 + 2))$.

Macht der Rekursion

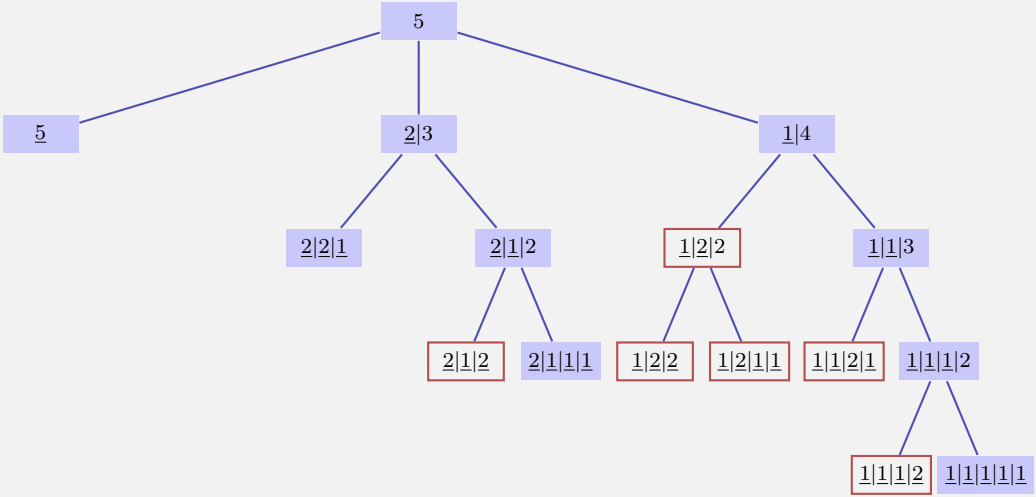
Welchen Fehler machen wir eigentlich... Wir probieren mal aus:



Erkenntnis



Erkenntnis



Erlaube nur z.B. absteigende Folgen von Beträgen.

2. Java Exceptions

Beispiel einer Systemausnahme

```
1  import java.util.Scanner;
2  class ReadTest {
3      public static void main(String[] args){
4          int i = readInt("Zahl");
5      }
6      private static int readInt(String prompt){
7          System.out.print(prompt + ": ");
8          Scanner input = new Scanner(System.in);
9          return input.nextInt();
10     }
11 }
```

Eingabe: `Zahl: asdf`

Exceptions fangen

```
boolean done = false;
while (!done){
  try{
    int kinder = readInt("Kinder: ", 1);
    int kekse = readInt("Kekse: ", kinder);
    done = true;
  }
  catch(java.util.InputMismatchException ex){
    System.out.println("Your input was invalid. Try to use numbers nex
    // try to catch up with the input by ignoring last input
    String ignore = input.next();
  }
}
```

Exceptions werfen

```
public class Main {
    static void f(int i){
        if (i==0) { throw new Error(); }
        f(i-1);
    }

    public static void main(String[] args) {
        f(3);
    }
}
```

Exceptions werfen

```
public class Main {
    static void f(int i){
        if (i==0) { throw new Error(); }
        f(i−1);
    }

    public static void main(String[] args) {
        f(3);
    }
}
```

```
Exception in thread "main" java.lang.Error
at Main.f(Main.java:8)
at Main.f(Main.java:9)
at Main.f(Main.java:9)
at Main.f(Main.java:9)
at Main.main(Main.java:13)
```

Arten von Ausnahmen

Systemausnahmen
(runtime exceptions)

- Können überall auftreten
- *Können* behandelt werden
- Ursache: Bug im Programm

Benutzerausnahmen
(checked exceptions)

- Müssen deklariert werden
- *Müssen* behandelt werden
- Ursache: Unwahrscheinliches, aber prinzipiell mögliches Ereignis

Arten von Ausnahmen

Systemausnahmen
(runtime exceptions)

- Können überall auftreten
- *Können* behandelt werden
- Ursache: Bug im Programm

Benutzerausnahmen
(checked exceptions)

- Müssen deklariert werden
- *Müssen* behandelt werden
- Ursache: Unwahrscheinliches, aber prinzipiell mögliches Ereignis

User Exceptions

```
public class Main {

  static void f(int i){
    System.out.println("Exception wird geworfen.");
    throw new Exception();
  }


  public static void main(String[] args) {
    f(3);
  }
}
```

User Exceptions

```
public class Main {

  static void f(int i){
    System.out.println("Exc
    throw new Exception();
  }

  public static void main(String[] args) {
    f(3);
  }
}
```

Compiler:
./Root/Main.java:7: error: unreported exception Exception; must be caught or declared to be thrown throw new Exception();

User Exceptions

```
public class Main {

  static void f(int i) throws Exception{
    System.out.println("Exception wird geworfen.");
    throw new Exception();
  }

  public static void main(String[] args) {
    f(3);
  }
}
```

User Exceptions

```
public class Main {

  static void f(int i) throws Exception{
    System.out.println("Exception wird geworfen.");
    throw new Exception();
  }

  public static void main(String[] args) {
    f(3);
  }
}
```

Compiler:
./Root/Main.java:7: error: unreported exception Exception; must be caught or declared to be thrown f(3);

User Exceptions

```
public class Main {
  static void f(int i) throws Exception{
    System.out.println("Exception wird geworfen.");
    throw new Exception();
  }

  public static void main(String[] args) {
    try{
      f(3);
    }
    catch (Exception e){
      System.out.println("Exception wurde gefangen.");
    }
  }
}
```

User Exceptions

```
public class Main {
  static void f(int i) throws Exception{
    System.out.println("Exception wird geworfen.");
    throw new Exception();
  }

  public static void main(String[] args) {
    try{
      f(3);
    }
    catch (Exception e){
      System.out.println("Exception wurde gefangen.");
    }
  }
}
```

```
Exception wird geworfen.
Exception wurde gefangen.
```

Sortieralgorithmen

Geben Sie bei nachfolgenden Folien an

- Anzahl Vergleichsoperationen im besten und schlechtesten Fall
- Anzahl Swaps (Speicherverschiebungen) im besten und schlechtesten Fall
- (Wie) kann die Implementation jeweils verbessert werden?

Geben Sie Ihre Antworten in Landau-Notation an und begründen Sie jeweils. Fragen dieser Art müssten Sie in einer Prüfung beantworten können.

Bubble-Sort

```
void bubbleSort(int[] A, int l, int r) {
   for(int i=r; i>l; i--)
      for(int j=l; j<i; j++)
         if(A[j] > A[j+1])
            swap(A, j, j+1);
}
```

Insertion-Sort

```
void insertionSort(int[] A, int l, int r) {
   for(int i=l; i<=r; i++)
      for(int j=i−1; j>=l && A[j] > A[j+1]; j−−)
         swap(A, j, j+1);
}
```

Selection-Sort

```
void selectionSort(int[] A, int l, int r) {
   for(int i=l; i<r; i++) {
      int minJ = i;
      for(int j=i+1; j<=r; j++){
         if(A[j] < A[minJ])
            minJ = j;
      }
      if(minJ != i)
        swap(A, i, minJ);
   }
}
```

Quick-Sort

```
void quicksort(int[] A, int l, int r) {
   if(l<r){
      int i=l+1, j=r;
      do{
         while(i<j && A[i] <= A[l])
            i++;
         while(i<=j && A[j] >= A[l])
            j--;
         if(i<j) swap(A, i, j);
      } while(i<j);
      swap(A, l, j);
      quicksort(A, l, j-1);
      quicksort(A, j+1, r);
   }
}
```

Fragen oder Anregungen?