



### Aufgabe 1

2 + 4 = 6T

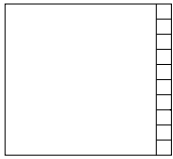
#### Nicht optimaler Greedy

- Geben Sie (informell) einen Greedy-Algorithmus an, welcher zu einem gegebenen Rechteck eine Zerlegung in möglichst wenige Quadrate berechnet.
- Zeigen Sie anhand eines Beispiels, daß der Algorithmus nicht immer eine optimale Zerlegung findet.
- Geben Sie eine Folge von Rechtecken an, in deren Zerlegung nur Quadrate unterschiedlicher Seitenlängen vorkommen.
- Geben Sie für beliebige Rechtecke aus Teilaufgabe 3 eine geschlossene Form für die (absteigende) Folge der Seitenlängen der in der Zerlegung vorkommenden Quadrate an.
- Begründen Sie, daß der Greedy-Algorithmus für Rechtecke aus Teilaufgabe 3 optimal ist.

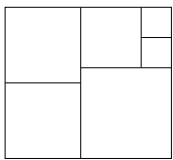
#### Lösung zu Aufgabe 1

- Wähle als nächste Seitenlänge für ein Quadrat jeweils die minimal zur Verfügung stehende Seitenlänge des noch nicht zerlegten Teils des Rechtecks.
- Ein Rechteck mit den Maßen  $10 \times 11$  cm:

Raffke: 1 mal  $10 \times 10$   
 10 mal  $1 \times 1$



Verbesserung: 1 mal  $6 \times 6$   
 2 mal  $5 \times 5$   
 1 mal  $4 \times 4$   
 2 mal  $2 \times 2$



- Wähle die Fibonacci-Folge, wobei  $F_0$  und  $F_1$  nicht verwendet werden, da die Seitenlängen teilerfremd sein müssen:

$$r_0 = F_2 \times F_3, \quad r_i = F_{i+2} \times F_{i+3}$$

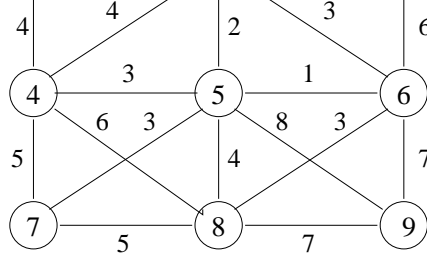
- ?
- Im Falle dessen, daß die Seitenlängen zwei aufeinander folgende Fibonacci-Zahlen sind, gibt es keine andere Zerlegung. Folglich muß diese optimal sein

### Aufgabe 2

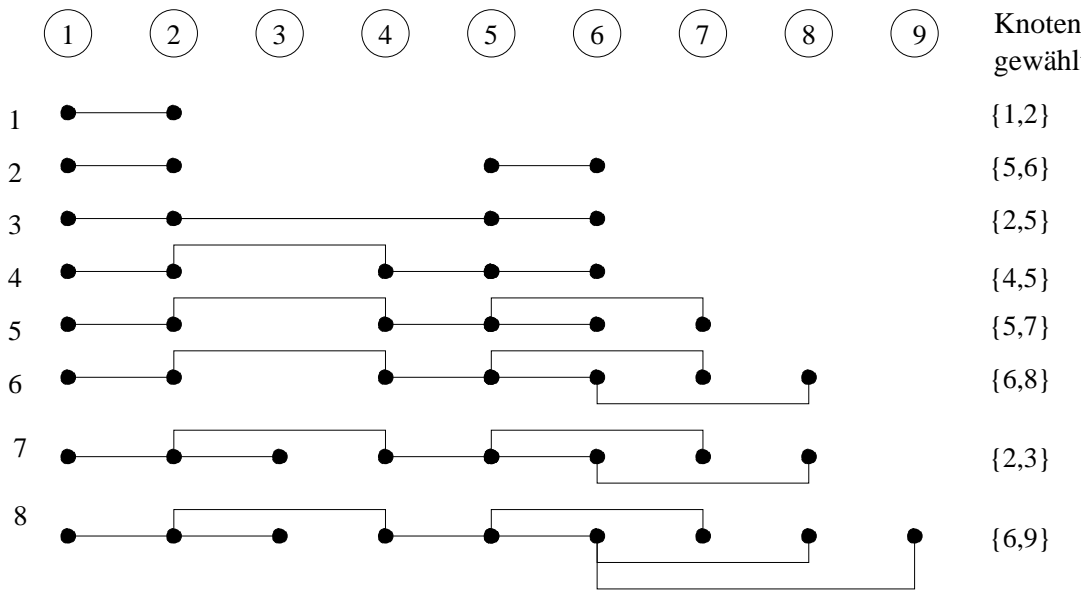
?

#### Algorithmus von Kruskal

Bestimmen Sie mit dem Algorithmus von Kruskal den minimal spannenden Baum des folgenden Graphen, und geben Sie die Kosten des minimal spannenden Baumes an. Geben Sie Zwischenschritte an!



**Lösung zu Aufgabe 2**



Kosten des Baumes:  $1+1+2+3+3+3+5+7=25$

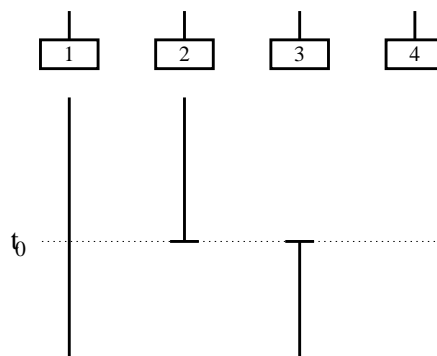
**Aufgabe 3**

?

Und weil's so schön war ...

Können Sie die Aufgabe 3 von Blatt 2 auch für 4 Schalter lösen?

**Lösung zu Aufgabe 3**



Zu Beginn werden die ersten beiden Schalter betätigt. Zum Zeitpunkt  $t_0$  wird der zweite wieder ausgeschaltet und dafür der dritte wieder eingeschaltet. Dann rennt man so schnell wie möglich in den Keller.

Brennt das Licht aber die Lampe ist kalt, ist es Schalter 3, brennt das Licht und die Lampe ist warm, ist es Schalter 1. Ist das Licht aus und die Lampe warm, ist es Schalter 2. Ist das Licht aus und die Lampe kalt, so muß es Schalter 4 sein.

**Aufgabe 4**

?

Huffman-Codierung, Datentyp Baum

```
data Baum a = Blatt a | Baum a :^: Baum a deriving (Eq,Read,Show)
```

Die Klasse Functor a definiert eine Funktion fmap, die analog zu map auf jedes Element des Datentyps a eine Funktion anwendet. Implementieren Sie eine Instanz für Baum a.

- b) Ausgehend vom Datentyp F2 wird mit  
`type Bin = [F2]`

eine einfach Darstellung von Binärworten realisiert. Implementieren Sie die folgenden Funktionen:

```
char2Bin :: Char → Bin — z.B. char2Bin 'a' ergibt [I,0,0,0,0,0,I,0]
```

```
bin2Char :: Bin → Char
```

```
bin2String :: Bin → String
```

```
string2Bin :: String → Bin
```

Vergessen Sie dabei die führenden Nullen nicht! Ein Aufruf von `string2Bin.bin2String` soll die identische Abbildung ergeben.

- c) Nun zu Huffman: Die Funktionen

```
huffmanBaum :: String → Baum Char
```

```
encodeHuffman :: String → Baum Char → String
```

```
decodeHuffman :: Eq a ⇒ String → Baum Char → String
```

sollen die übergebenen Zeichenketten Huffman-codieren bzw. den zugehörigen Huffmanbaum berechnen.

*Hinweis:* Die Module `Data.List` und `Data.Char` enthalten einige für die Teilaufgaben nützliche Funktionen:  
`ord`, `chr`, `nub`, `sortBy`, `insertBy`, `compare`

Eine effizientere `encodeHuffman` Funktion können Sie z.B. durch Verwendung eines Arrays erreichen. Dann sollten Sie auch noch das Modul `Data.Array` einbinden.