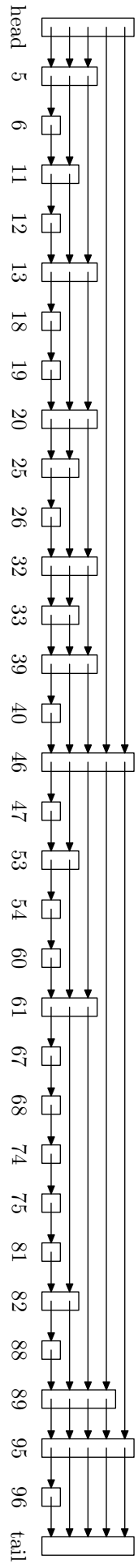
RWTH Aachen **SoSe 2007**
Lehrgebiet Theoretische Informatik Blatt 7
Kneis—Mölle—Richter—Rossmanith—Thiemann 21.5.2007

Übungen zur Vorlesung Datenstrukturen und Algorithmen

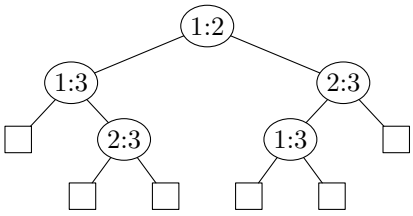
T16

Gegeben sei die rechts abgebildete Skip-List. Welche Knoten werden bei einer Suche nach 54 beziehungsweise 80 berührt?



T17

Gibt es einen vergleichsbasierten Sortieralgorithmus, der folgenden Vergleichsbaum besitzt?



T18

Analysieren Sie die Anzahl der Vergleiche (von Elementen) im Erwartungswert für den Algorithmus Mergesort, wenn die Eingabe aus $n = 2^k$ verschiedenen Schlüsseln besteht und jede ihrer Permutationen gleich wahrscheinlich ist.

```
procedure Mergesort(l,r):
  if l ≥ r then return fi;
  m ← ⌈(r+l)/2⌉;
  Mergesort(l,m−1);
  Mergesort(m,r);
  i ← l; j ← m; k ← l;
  while k ≤ r do
    if a[i] ≤ a[j] and i<m or j>r
    then b[k] ← a[i]; k ← k+1; i ← i+1
    else b[k] ← a[j]; k ← k+1; j ← j+1 fi
  od;
  for i=l ,..., r do a[k] ← b[k] od
```

H14 (5 Punkte)

Sei $\mathcal{F}$ die Menge aller Funktionen $U \to \{1, \ldots, m\}$.

Beweisen oder widerlegen Sie: $\mathcal{F}$ ist eine universelle Familie von Hashfunktionen.

Falls Ihre Antwort *ja* ist, sollte man diese Familie dann praktisch verwenden? Falls die Anwort *nein* ist, kann man es einfach reparieren?

H15 (10+10 Punkte)

Implementieren Sie eine Klasse *Bitset* für durch Bitmaps repräsentierte Mengen, die Funktionen für die Berechnung aller Teilmengen enthält:

Bitset(int i)	Konstruktor mit Menge
void start()	initialisiere den Teilmengeniterator
int subset()	liefere die aktuelle Teilmenge
boolean more()	teste, ob eine weitere Iteration möglich ist
void step()	gehe zur nächsten Teilmenge über

Insbesondere soll Ihre Klasse mit dem unten dargestellten Programm korrekt funktionieren, dessen Ausgabe rechts vom Quelltext aufgelistet ist. Selbstverständlich soll die Korrektheit nicht nur für 51, sondern für alle *int*-Werte erzielt werden.

```
class Bitset {
  ...
}

public class H15 {
  public static void main(String args[]) {
    Bitset s = new Bitset(51);
    for(s.start(); s.more(); s.step()) {
      for(int i=31; i ≥ 0; i--)
        System.out.print((s.subset()>>i)&1);
      System.out.println();
    }
  }
}
```

```
00000000000000000000000000000000
00000000000000000000000000000001
00000000000000000000000000000010
00000000000000000000000000000011
00000000000000000000000000010000
00000000000000000000000000010001
00000000000000000000000000010010
00000000000000000000000000010011
00000000000000000000000000100000
00000000000000000000000000100001
00000000000000000000000000100010
00000000000000000000000000100011
00000000000000000000000000110000
00000000000000000000000000110001
00000000000000000000000000110010
00000000000000000000000000110011
```

Zur Erläuterung: Die 51, binär 110011, repräsentiert eine vierelementige Menge, die das erste, zweite, fünfte und sechste Element des Universums enthält. Natürlich hat diese Menge genau $2^4 = 16$ Teilmengen, deren binäre Repräsentationen gerade die Ausgabe des Programmes bilden.

Zur Analyse wollen wir annehmen, daß ein *int* nicht zwangsweise auf 32 Bit beschränkt ist, sondern die variable Länge w hat. Implementieren Sie Ihre Klasse so, daß die folgenden amortisierten Laufzeiten eingehalten werden:

Bitset(int i)	$O(w)$
void start()	$O(w)$
int subset()	$O(1)$
boolean more()	$O(1)$
void step()	$O(1)$

Beweisen Sie diese Schranken!