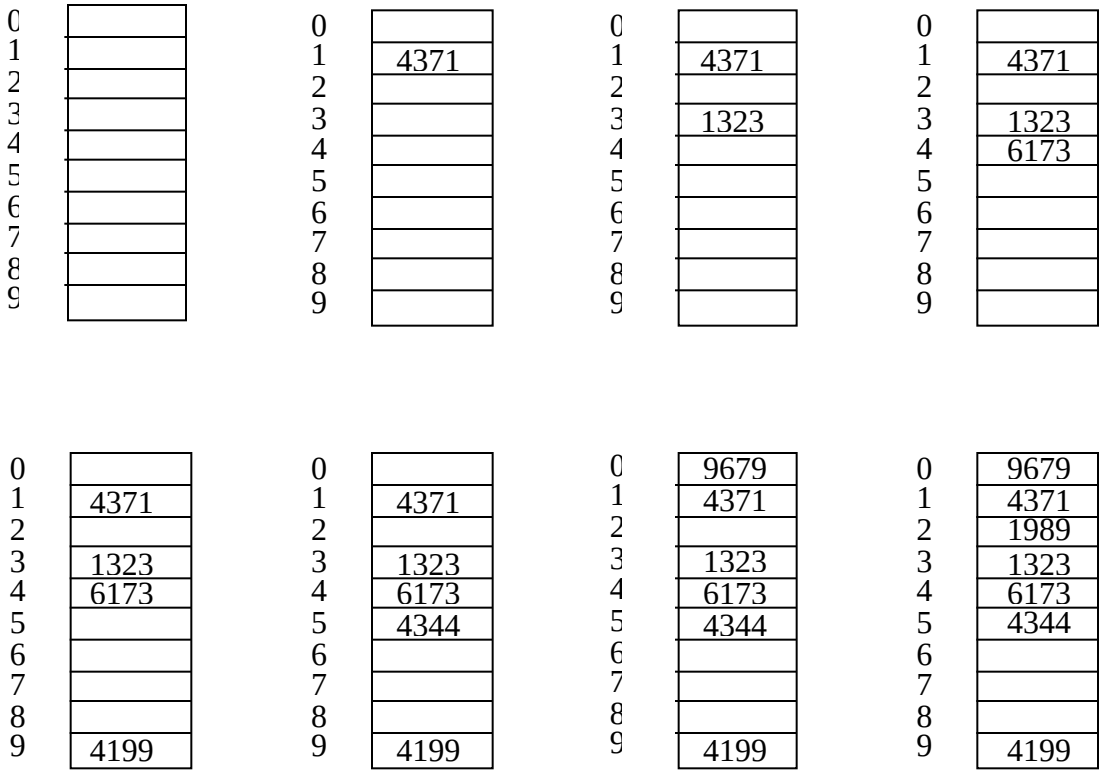
Opgaveløsninger
(sæt 12)

Opgave 46: 20.5 (1 point)

(a) Lineær prøvning

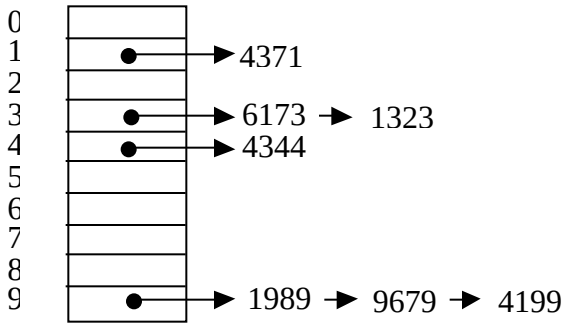


(b) Kvadratisk prøvning

0	9679
1	4371
2	
3	1323
4	6173
5	4344
6	
7	
8	1989
9	4199

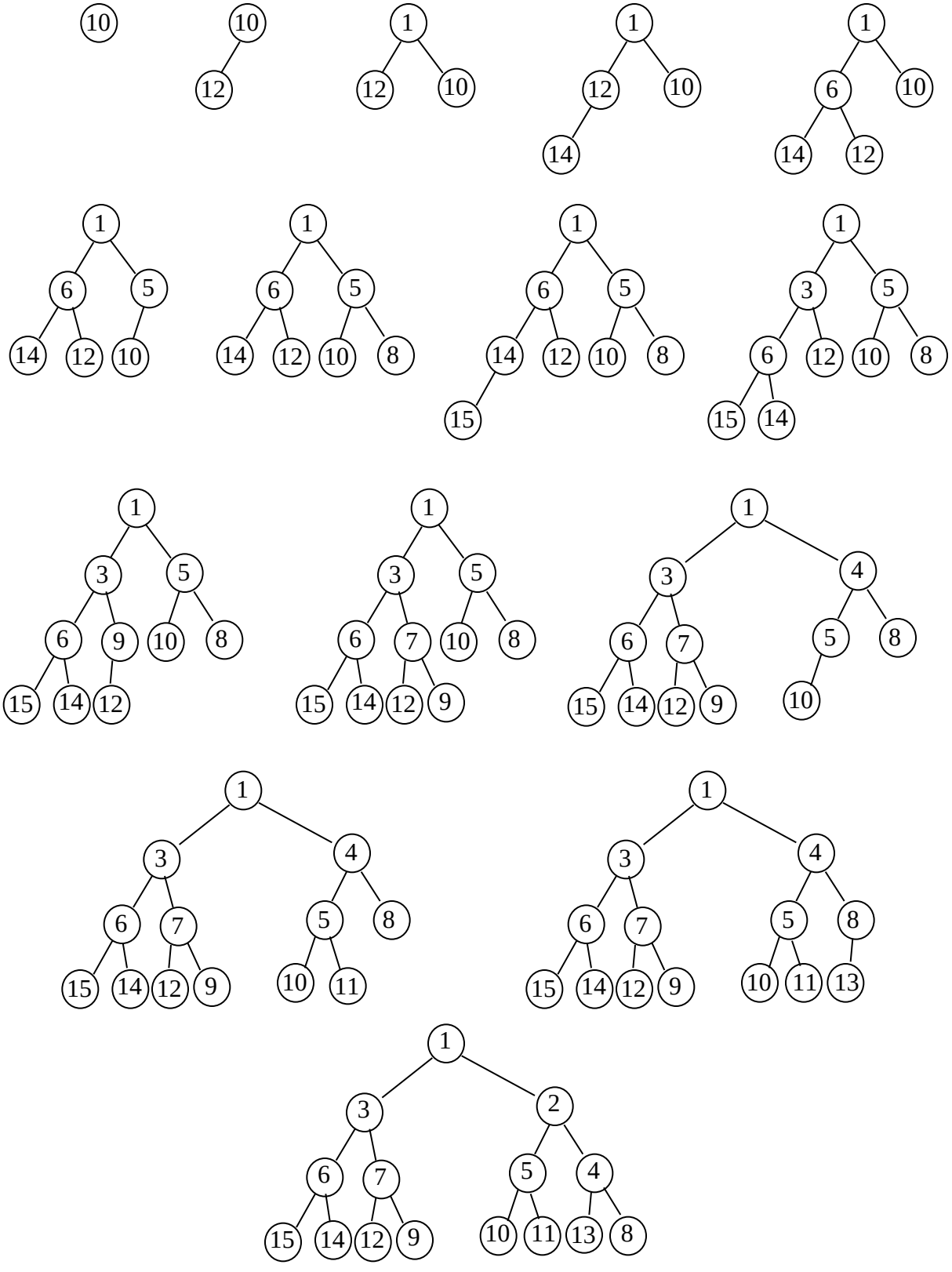
Kun den sidste indsættelse giver en afvigelse i forhold til lineær prøvning.

(c) Separat kædning

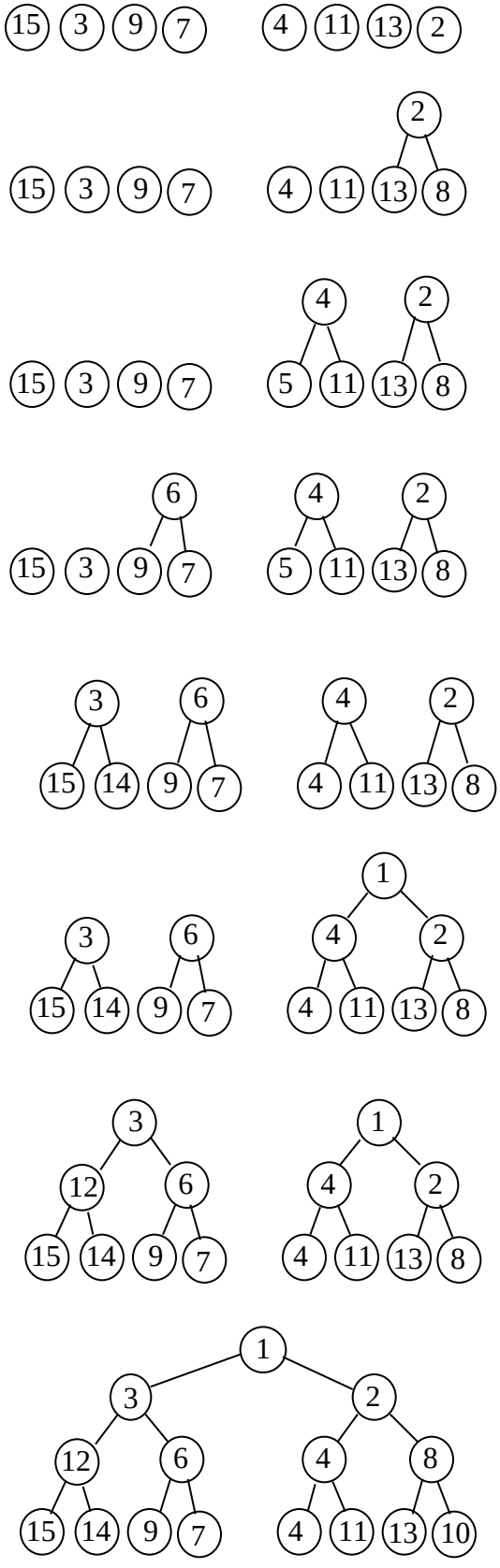


Opgave 47: 21.3 (2 point)

a.



b.


15 3 9 7
4 11 13 2
15 3 9 7
2
4 11 13 8
15 3 9 7
4
2
5 11 13 8
6
15 3 9 7
4
2
5 11 13 8
3
6
15 14 9 7
4
2
4 11 13 8
3
6
15 14 9 7
1
4
2
4 11 13 8
3
12
6
15 14 9 7
1
4
2
4 11 13 8
1
3
2
12
6
4
8
15 14 9 7
4 11 13 10

Opgave 48: 21.7 (1 point)

Nej.

Antag at alle elementer er ens. Algoritmen opbygger først en hob, hvorefter elementer udtages et for et og placeres bagest i arrayet. Opbygningen af hoben ændrer ikke på elementernes rækkefølge. Det gør derimod den efterfølgende del af algoritmen, idet den vender deres rækkefølge.

Opgave 49: 21.22 (1 point, ikke-obligatorisk)

En simpel løsning er at sørge for at antallet bliver Fibonaccital ved blot at tilføje et passende antal ekstra kunstige blokke til indbåndene. Hver sådan blok indeholder en kunstig post, hvis nøgle er større end alle de reelle posters. Når sorteringen er slut, vil de kunstige poster ligge til sidst på resultatbåndet.

Opgave 50: 21.23 (1 point, ikke-obligatorisk)

```
  static void percDown(Comparable[] a, int index, int size) {
      int child;
      Comparable tmp = a[index];

      for ( ; 2 * index + 1 < size; i = child) {
          child = 2 * index + 1;
          if (child != size && a[child].compareTo(a[child + 1]) < 0)
              child++;
          if (tmp.compareTo(a[child]) < 0)
              a[index] = a[child];
          else
              break;
      }
      a[index] = tmp;
  }
```