R-2.10

Give an example of a worst-case sequence with n elements for insertion-sort, and show that insertion-sort runs in $\Omega(n^2)$ time on such a sequence.

The worst-case sequence with n elements for insertion-sort is a sequence that has the reserved order. For example, there is a sequence like $\mathrm{n}, \mathrm{n}-1, \dots, \mathrm{i}, \dots, 2, 1$. With this sequence, each element i will be moved to the first place with $(\mathrm{n}-\mathrm{i})$ steps. As a result, for n elements, there will be $\sum_{i=1}^{n}(n-i) = \frac{1}{2}n(n-1)$ steps, which means $\Omega(n^2)$ time.

R-2.13

Suppose a binary tree T is implemented using a vector S, as described in Section 2.3.4. If n items are stored in S in sorted order, starting with index 1, is the tree T a heap?

Yes, tree T is a heap.

Because at the same depth, the nodes is ordered from the min to the max, and the first node at this tier is the smallest one and the last node at this tier is the biggest one. And the last node's value at this tier must be smaller than the first node's value at the child's tier. As a result, each of the children's value is bigger than each of their parents' value. Thus, this tree is a heap.

R-2.19

Draw the 11-item hash table resulting from hashing the keys 12, 44, 13, 88, 23, 94, 11, 39, 20 ,16, and 5, using the hash function $\mathrm{h(i)} = (2\mathrm{i}+5) \bmod 11$ and assuming collisions are handled by chaining.

$\mathrm{h}(12) = (2 \times 12 + 5) \bmod 11 = 7$ $\quad \mathrm{h}(44) = (2 \times 44 + 5) \bmod 11 = 5$

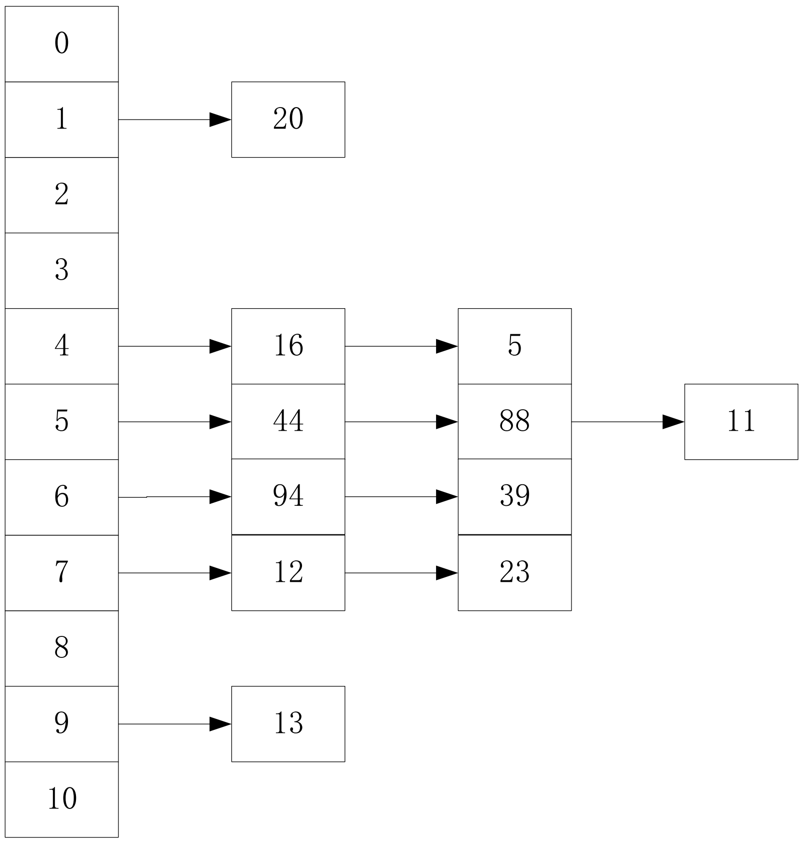
$\mathrm{h}(13) = (2 \times 13 + 5) \bmod 11 = 9$ $\quad \mathrm{h}(88) = (2 \times 88 + 5) \bmod 11 = 5$

$\mathrm{h}(23) = (2 \times 23 + 5) \bmod 11 = 7$ $\quad \mathrm{h}(94) = (2 \times 94 + 5) \bmod 11 = 6$

$\mathrm{h}(11) = (2 \times 11 + 5) \bmod 11 = 5$ $\quad \mathrm{h}(39) = (2 \times 39 + 5) \bmod 11 = 6$

$\mathrm{h}(20) = (2 \times 20 + 5) \bmod 11 = 1$ $\quad \mathrm{h}(16) = (2 \times 16 + 5) \bmod 11 = 4$

$\mathrm{h}(5) = (2 \times 5 + 5) \bmod 11 = 4$



R-2.22

What is the result of Exercise R-2.19 assuming collisions are handled by double hashing using a secondary hash function $h'(k) = 7 - (k \bmod 7)$?

$h'(12) = 7 - (12 \bmod 7) = 2$ $h'(44) = 7 - (44 \bmod 7) = 5$

$h'(13) = 7 - (13 \bmod 7) = 1$ $h'(88) = 7 - (88 \bmod 7) = 3$

$h'(23) = 7 - (23 \bmod 7) = 5$ $h'(94) = 7 - (94 \bmod 7) = 4$

$h'(11) = 7 - (11 \bmod 7) = 3$ $h'(39) = 7 - (39 \bmod 7) = 3$

$h'(20) = 7 - (20 \bmod 7) = 1$ $h'(16) = 7 - (16 \bmod 7) = 5$

$h'(5) = 7 - (5 \bmod 7) = 2$

k	h(k)	h'(k)	probes			
12	7	2	7			

44	5	5	5			
13	9	1	9			
88	5	3	5	8		
23	7	5	7	1		
94	6	4	6			
11	5	3	5	8	0	
39	6	3	6	9	1	4
20	1	1	1	2		
16	4	5	4	9	3	
5	4	2	4	6	8	10

11	23	20	16	39	44	94	12	88	13	5

C-2.10

Given an $O(n)$ time algorithm for computing the depth of all the nodes of a tree T, where n is the number of nodes of T.

We can develop a recursion method to print the depth of each node. When the current node is the root of the tree, the depth is 0. Next we can print its children's depth which is 1. Then we can print their children's depth which is 2. … With the recursion method, we can print the depth of the whole nodes.

```
algorithm function(v)
 if (v == T.root) then
  print(0)
 else
  for each v' which is a child of v do
   print(d+1)
   function(v')
```

C-2.31

Develop an algorithm that computes the kth smallest element of a set of n distinct integers in $\mathrm{O}(\mathrm{n} + \mathrm{k} \log n)$ time.

We can build a heap, which could take $\mathrm{O}(\mathrm{n})$ time. Then we can call removeMin k times, which will take $\mathrm{O}(\mathrm{k} \log n)$ time.

First the procedure removeMin will delete the root, which is the smallest node in the heap. Nest this procedure will delete the last node and move it to the first node. Then this will do down-heap bubbling, which will take $\mathrm{O}(\log n)$ time at the worst case ($\log n$ is the height of the heap).