**Exercise 6.1**

1. **What is the minimum and maximum number of elements in a heap of height h?**

**Ans)**

Maximum number of elements: 2h+1 – 1

Minimum number of elements: 2h

1. **Show that an n element heap has height lgn.**

**Ans)**

Each node in a heap at at most 2 children. The root is a single node with 2 children. Sum of all nodes:

1 + 2 + 22 + 23 + … + 2h = n

* 2h+1 – 1 = n
* 2h+1 = n+1

Taking log on both sides:  
 h = log(n+1) -1

In worst case scenario, number of nodes = 2h

n = 2h

* h = lgn

Therefore, we can say,

2h < n < 2h+1 -1 < 2h+1

* h <= lgn < h+1
* h = floor(lgn)

1. **Show that in any subtree of a max-heap, the root of the subtree contains the largest value occurring anywhere in that subtree.**

**Ans)**

In a max-heap,

PARENT(i) > i > LEFT(i)

and

PARENT(i) > i > RIGHT(i)

If root is not the largest value, this property will be violated as at some node, LEFT(i) > i or RIGHT(i) > i

1. **Where in a max-heap might the smallest element reside, assuming that all elements are distinct?**

**Ans)** In the leaves of the heap. In the array, anywhere between A[ceil[n/2]] and A[A.heap-size]

1. **At which levels in a max heap might the kth largest element reside, for 2<= k <= floor(n/2) assuming all elements are distinct?**

**Ans)**

Anywhere except leaves and root of the heap. So, all levels except level 0 and level floor(lg(n))