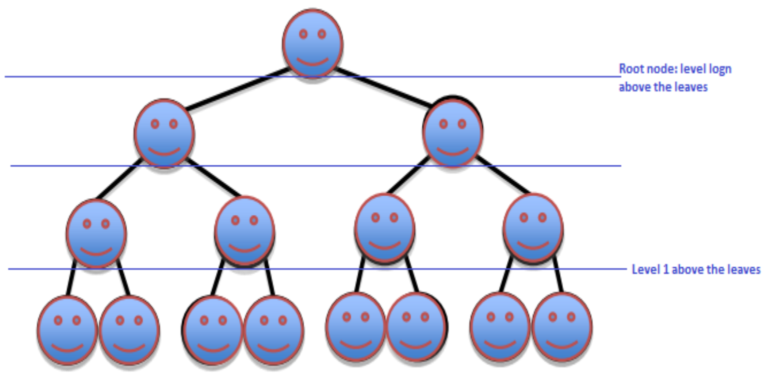
Convert A[1..n] to max heap:

```
void buildMaxHeap(A, n){
        for i=n/2 downto 1
                maxHeapify(A, i);
}
```



Time complexity estimation for ***buildMaxHeap***

Observe that, ***maxHeapify*** takes

- O(1) for time for nodes at level 1 above the leaves. There are $\frac{n}{4}$ nodes at level 1.
- O(2) for time for nodes at level 2 above the leaves. There are $\frac{n}{8}$ nodes at level 2.
- …
- O(L) for time for nodes at level L above the leaves. There are $\frac{n}{2^{L+1}}$ nodes at level L.
- The root has only 1 node and it is logn level above the leaves.

Therefore, total amount of work for the **for** *loop*:

$$T(n) = \frac{n}{4}*1*c + \frac{n}{8}*2*c + \frac{n}{16}*3*c + \ldots + 1*\log n*c$$

Set $\frac{n}{4} = 2^k$, we get

$$T(n) = c * 2^k * \left(\frac{1}{2^0} + \frac{2}{2^1} + \frac{3}{2^2} + \cdots + \frac{k+1}{2^k}\right)$$

$$\Rightarrow T(n) = c * 2^k * \sum_{i=0}^{k} \frac{i+1}{2^i}$$

Where $\sum_{i=0}^{k} \frac{i+1}{2^i}$ is a convergent series and it's bounded by a constant.

Therefore, ***buildMaxHeap*** takes **O(n)**