Initializing the Heap: `HeapSorter()`

```
public class HeapSorter extends ASorter
{
  public HeapSorter(int[] A, int lo, int hi)
  {
    for (int cur = (hi - lo + 1) / 2; cur >= lo; cur--) {
      Heapifier.Singleton.siftDown(A, lo, cur, hi);
    }
  }
  . . .
```

Analysis of `HeapSorter()`'s running time

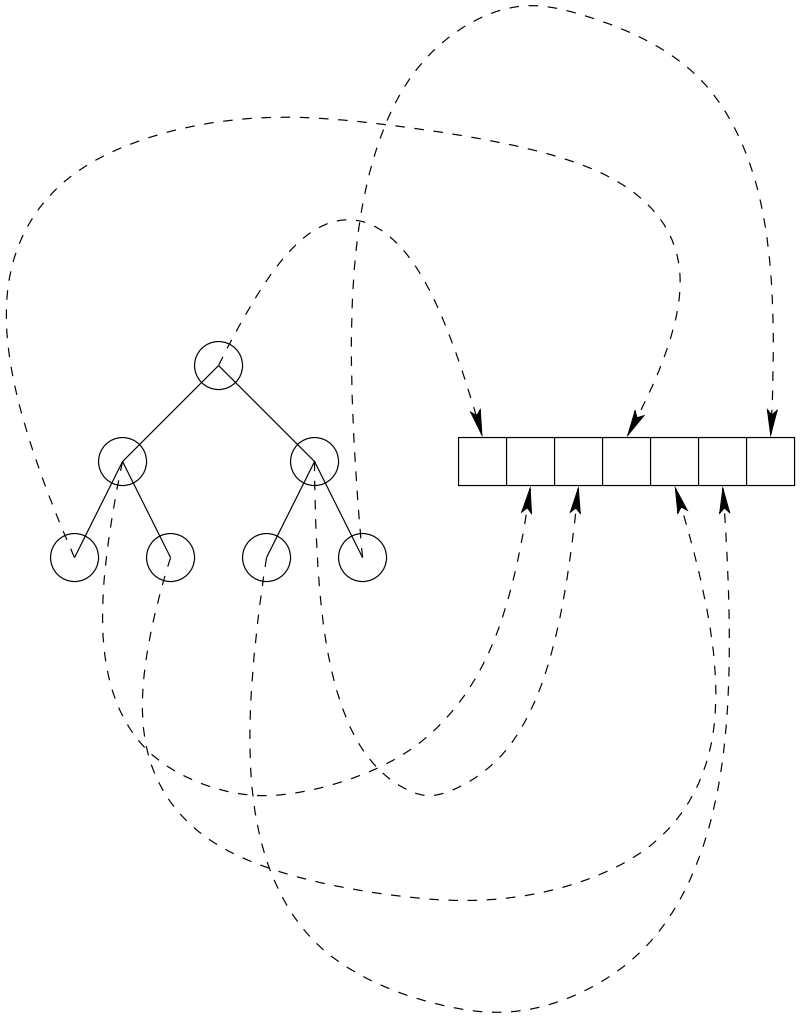
- We can derive a tighter bound than $O(n\ log\ n)$ by observing that the time for `siftDown()` to run at a node varies with the height of the node in the tree, and the heights of most nodes are small.

- The tighter analysis relies on the property that in an n-element heap there are at most $\lceil n/2^{h+1} \rceil$ nodes of height h.

- The time required by `siftDown()` when called on a node of height h is $O(h)$, so we can express the total cost of `HeapSorter()` as

$$\sum_{h=0}^{\lfloor log\ n \rfloor} \lceil \frac{n}{2^{h+1}} \rceil O(h) = O(n \sum_{h=0}^{\lfloor log\ n \rfloor} \frac{h}{2^h}). \quad (1)$$

Analysis of `HeapSorter()`'s running time (cont.)



Analysis of `HeapSorter()`'s running time (cont.)

The last summation can be evaluated by differentiating and multiplying by x both sides of the infinite geometric series (for $|x| < 1$)

$$\sum_{k=0}^{\infty} x^k = \frac{1}{1-x}, \tag{2}$$

to obtain

$$\sum_{k=0}^{\infty} k x^k = \frac{x}{(1-x)^2} \tag{3}$$

in which $x = 1/2$ is substituted to yield

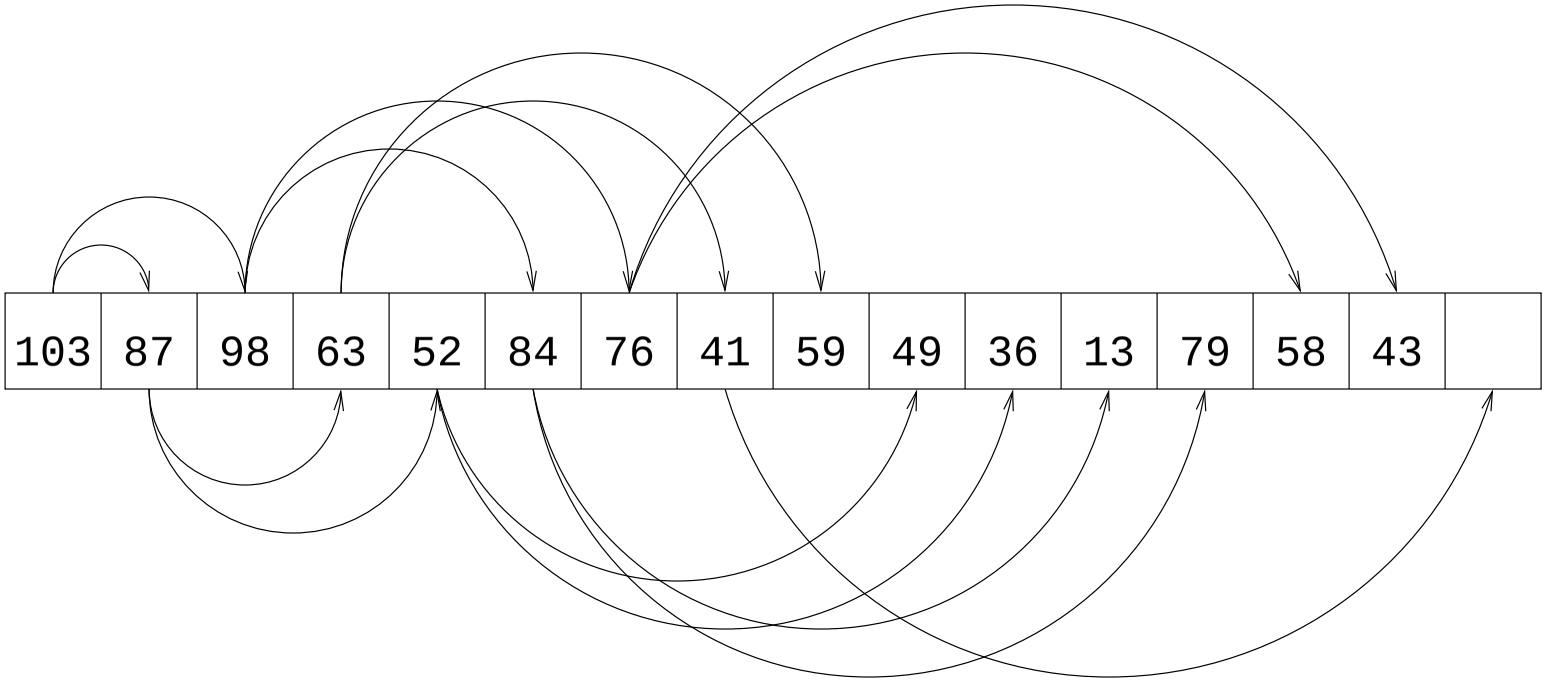
$$\sum_{h=0}^{\infty} \frac{h}{2^h} = \frac{1/2}{(1-1/2)^2} = 2. \tag{4}$$

Analysis of `HeapSorter()`'s running time (cont.)

Thus, the running time of the `HeapSorter()` constructor can be bounded as

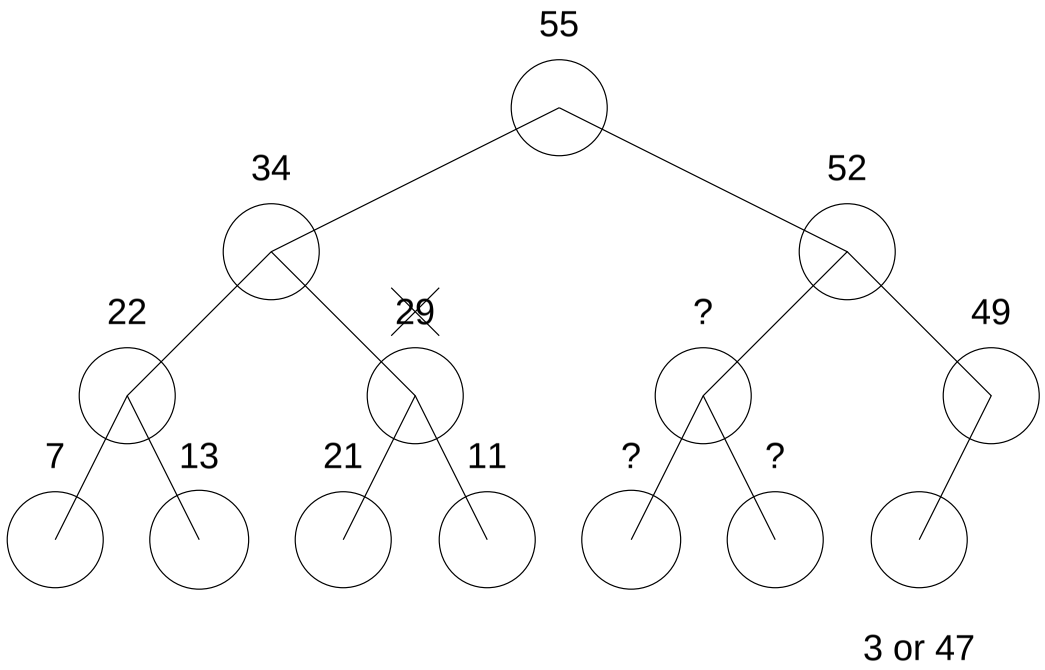
$$O(n \sum_{h=0}^{\lfloor log\ n \rfloor} \frac{h}{2^h}) = O(n \sum_{h=0}^{\infty} \frac{h}{2^h}) = O(2n) = O(n). \quad (5)$$

Example of `siftUp()`



Removal Of A Non-Root Node From A Heap

- Consider removing the node with the key 29 from the following heap.



IPriorityQueue

```
package queues;

import java.util.Enumeration;

import ordered.IOrdered;

public interface IPriorityQueue {

    public void enqueue(IOrdered data);

    public IOrdered dequeue();

    public Enumeration enumeration();
}
```