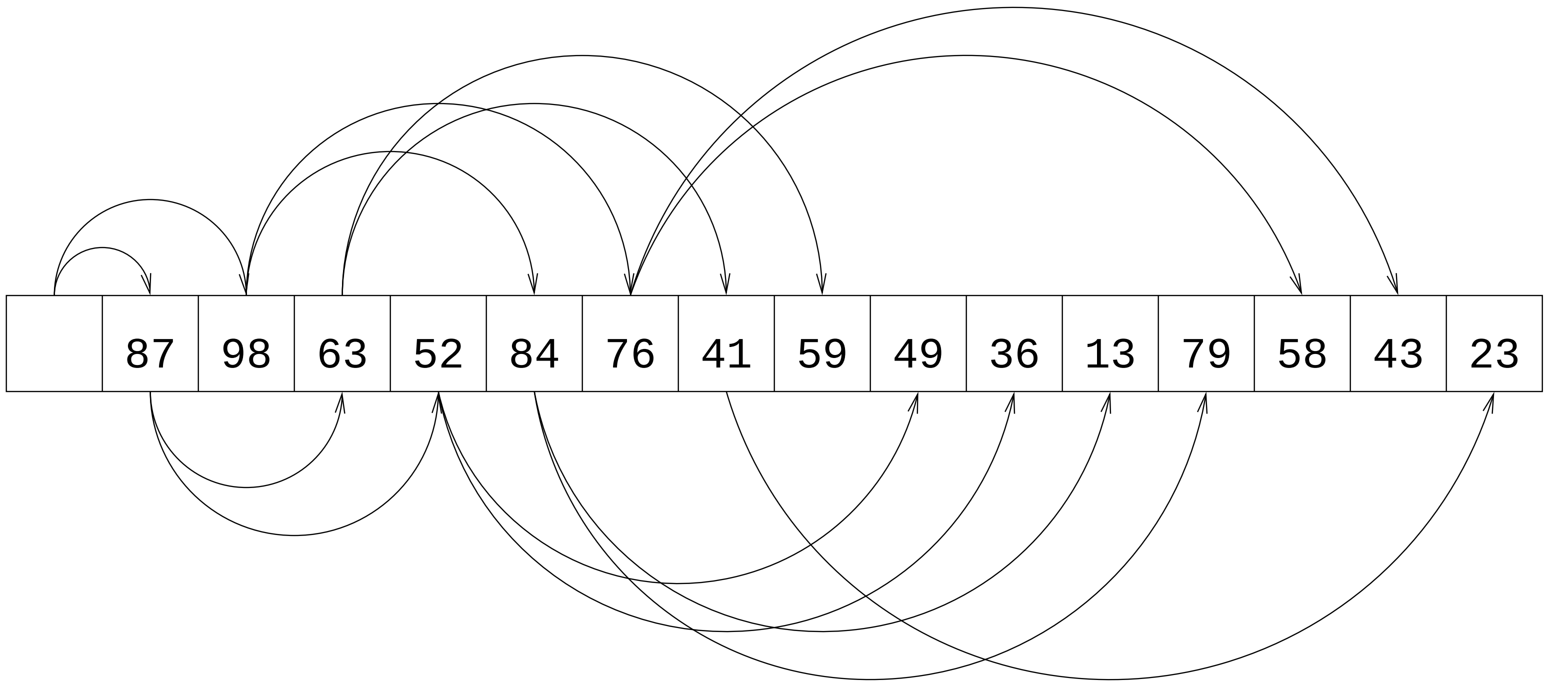
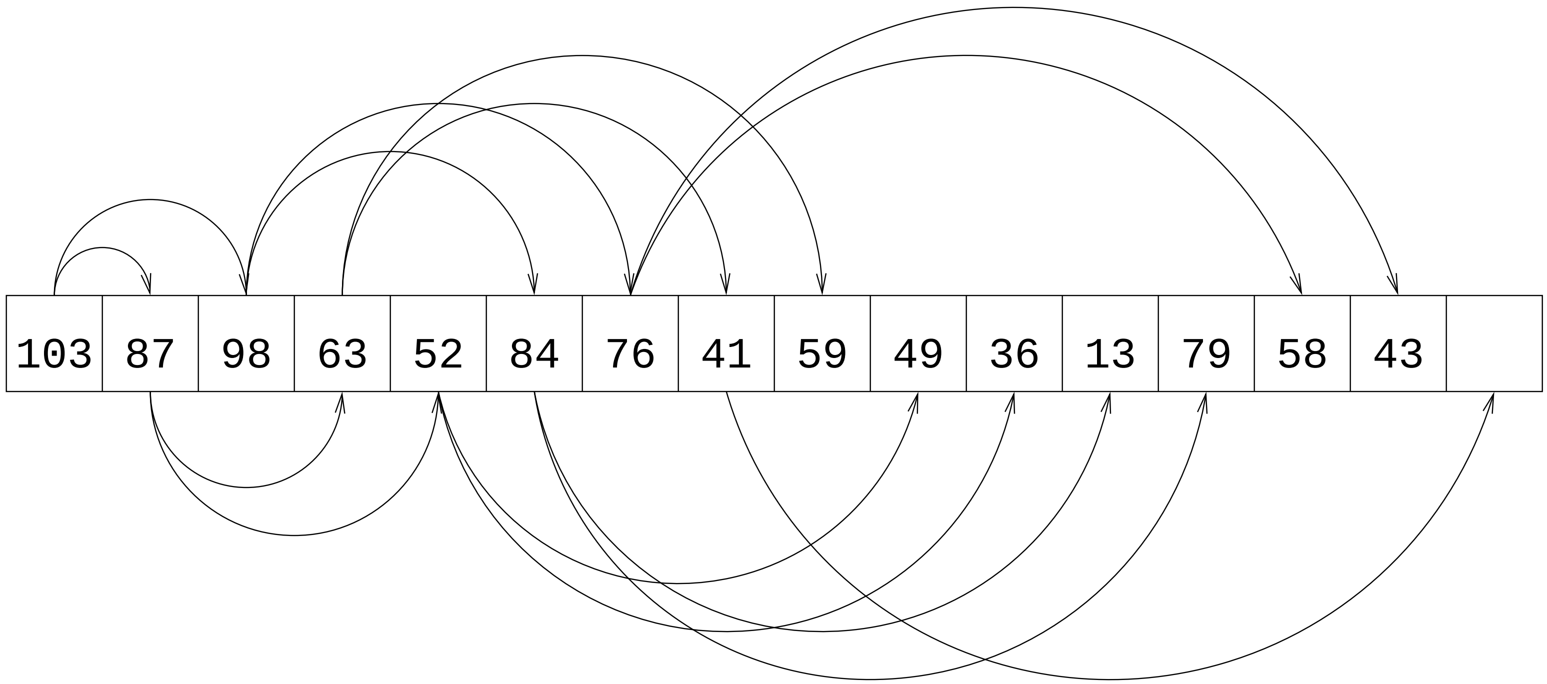
Overview

- Examples of `siftDown()` and `siftUp()`
- Analysis of `HeapSorter()`'s running time
- Quicksort

Example of `siftDown()`



Example of `siftUp()`



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

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

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

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

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

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

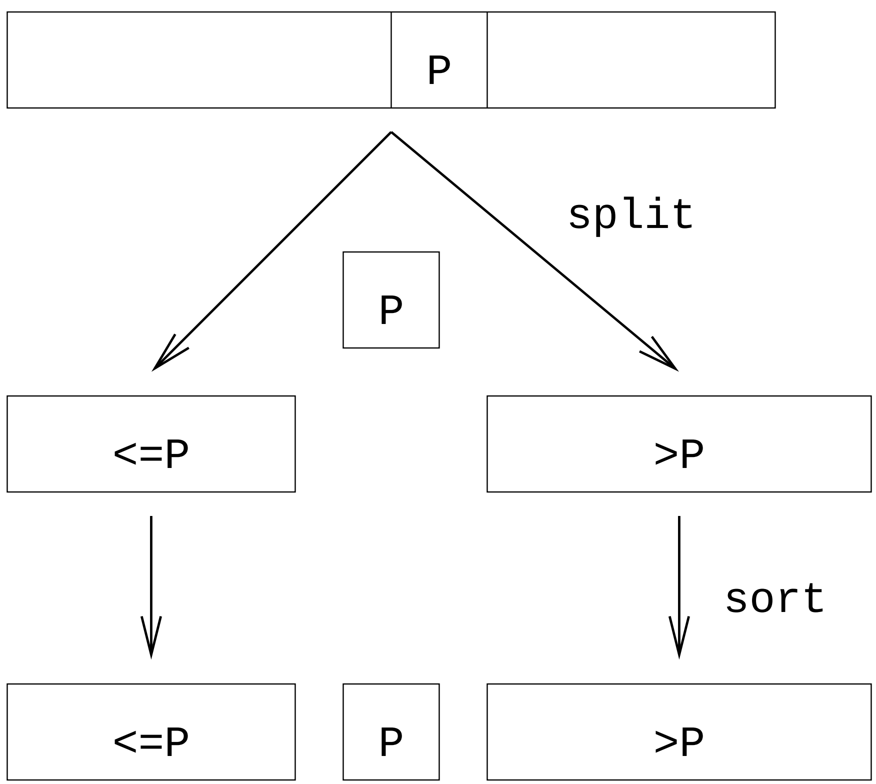
Analysis of `HeapSorter()`'s running time (cont.)

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

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

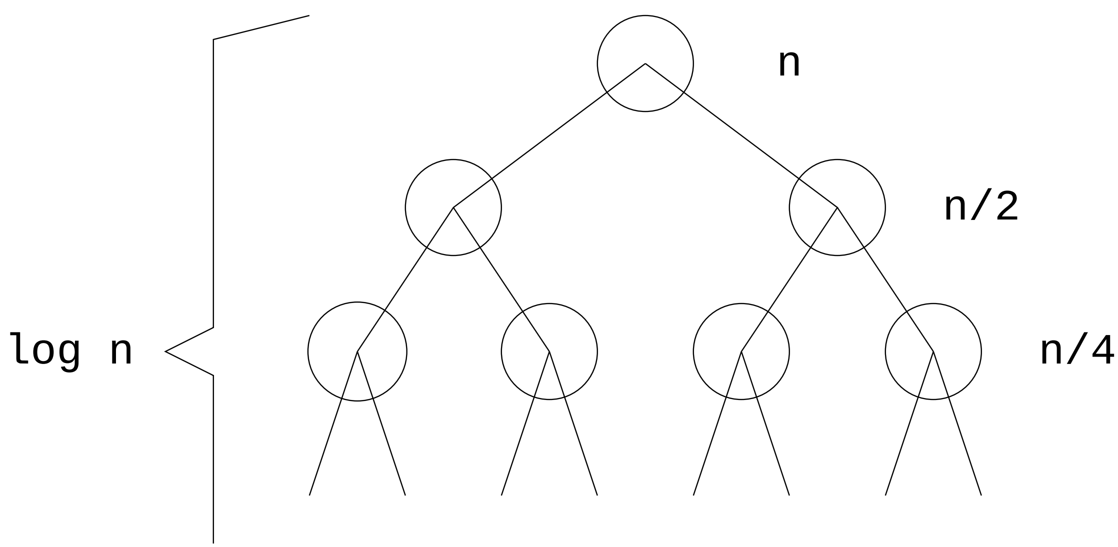
Quick Sort

- Quick Sort is a *hard-split, easy-join* method.
- The following diagram illustrate one step.



Quick Sort

- If the pivot chosen by `split()` divides the array into two (almost) equal-sized parts, each element is `split()` $log\ n$ times.



- Thus, in the expected case, Quick Sort takes $O(n\ log\ n)$ steps.