Problem: Implement quick sort.

Constraints

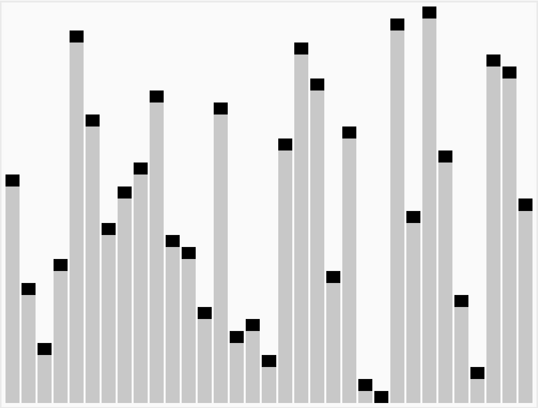
- Is a naive solution sufficient (ie not in-place)?
    - Yes
- Are duplicates allowed?
    - Yes
- Can we assume the input is valid?
    - No
- Can we assume this fits memory?
    - Yes

Test Cases

- None -> Exception
- Empty input -> []
- One element -> [element]
- Two or more elements

Algorithm

Wikipedia's animation:



- Set pivot to the middle element in the data
- For each element:
    - If current element is the pivot, continue
    - If the element is less than the pivot, add to left array
    - Else, add to right array
- Recursively apply quicksort to the left array
- Recursively apply quicksort to the right array
- Merge the left array + pivot + right array

Complexity:

- **Time: O(n log(n)) average, best, O(n^2) worst**
- **Space: O(n)**

Misc:

- **More sophisticated implementations are in-place, although they still take up recursion depth space**
- **Most implementations are not stable**

See Quicksort on wikipedia:

Typically, quicksort is significantly faster in practice than other Θ(nlogn) algorithms, because its inner loop can be efficiently implemented on most architectures [presumably because it has good cache locality], and in most real-world data, it is possible to make design choices which minimize the probability of requiring quadratic time.

See: Quicksort vs merge sort