Sorting

QUICK SORT

SORT COMPARISON

Quick sort (Hoare sort, 1959)

Divide-and-conquer approach

Comparative sort

Time complexity

- Worst case $O(n^2)$
- Best case O(n log(n))
- Average case T(n log(n))

Space complexity

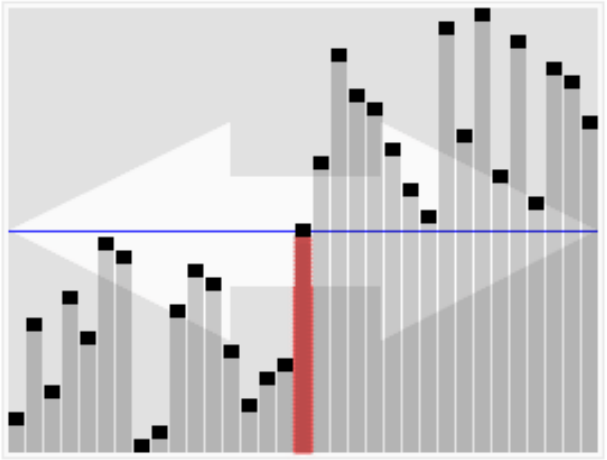
- Average case O(n log(n))
- Worst case O(n)

Sort principle

1) **Split** array into two parts where right parts **dominates** on left.

∃(*pivot* from i..j) ∀(*m* from i..j)

- if (*pivot* < m) then A[*pivot*] < A[m]
- if (*pivot* >= m) then A[*pivot*] >= A[m]



2) Sort both left and right parts recursively

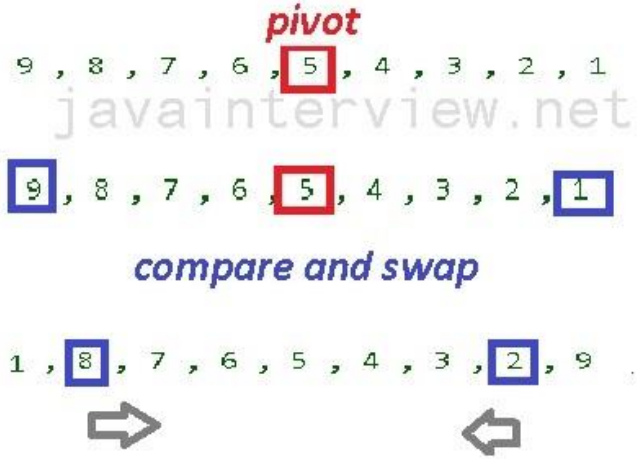
```
QSort (array, low, high) {
        mid = split(array, low, high);
        QSort(array, low, mid-1);
        QSort(array, mid+1, high);
}
```

Split algorithm

```
qsort(int a[], int left, int right) {
          int l      = left;              // set left index
          int r      = right;             // set right index
          int foo = 0;
          int pivot = a[(l + r) / 2];     // pivot selection is not determined
          while (l <= r) {
                    while ((a[l] < pivot) && (l <= right)) l++; // moving while ok
                    while ((a[r] > pivot) && (r >= left)) r--; // moving while ok

                    if (l <= r) {
                              foo = a[l]; a[l] = a[r]; a[r] = foo; // swap
                              l++;
                              r--;
                    }
          }

          if (r > left) qsort(a, left, r);
          if (l < right) qsort(a, l, right);
}
```



Drawbacks and best cases

Quick sort:

- Takes O(n) comparisons for array of equal elements
  (with 3 part splitting)
- Cache-friendly
- Parallelizable
- Works for linked lists (initially – streamers)
- Worst case n^2 for sorted array
  - if each split produces *1* and *n-1* element arrays
    (when selecting 1st element as a pivot in a sorted array)
- Unstable sort

Merge sort:

- Highly parallelizable for large amounts of data
- Worst case is still n log(n)
- Stable sort
- Takes n additional space always

Heap sort:

- O(1) additional memory
- Worst case is still n log(n)
- Cache-unfriendly (jumps over the array)
- Unstable sort