Animation Demos

http://www-cse.uta.edu/~holder/courses/cse2320/lectures/applets/sort1/heapsort.html

http://cg.scs.carleton.ca/~morin/misc/sortalg/

Order Statistics

- Maximum, Minimum n-1 comparisons

7	3	1	9	4	8	2	5	0	6

- MinMax
  - 2(n-1) comparisons
  - 3n/2 comparisons
- Max and 2ndMax
  - (n-1) + (n-2) comparisons
  - ???

k-Selection; Median

- Select the k-th smallest item in list
- Naïve Solution
  - Sort;
  - pick the k-th smallest item in sorted list.

  O(n log n) time complexity
- Randomized solution: Average case O(n)
- Improved Solution: worst case O(n)

```
QuickSort(A, p, r)
   if (p < r) then
      q = Partition(A, p, r)
      QuickSort(A, p, q)
      QuickSort(A, q+1, r)

Partition(A, p, r)
  x = A[r]
  i = p-1
  for j = p to r-1 do
     if (A[j] <= x) then
        i++
        SWAP(A[i], A[j])
  SWAP(A[i+1], A[r])
  return i+1
```


Partition Procedure Revisited

- The Partition code can be rewritten so that it accepts another parameter, namely, the pivot value. Let's call this new variation as PivotPartition.
- This change does not affect its time complexity.
- RandomizedPartition as used in RandomizedSelect picks the pivot uniformly at random from among the elements in the list to be partitioned.

Randomized Selection

```
RandomizedSelect(A, p, r, i)
   if (p = r) then
      return A[p]
   q = RandomizedPartition(A, p, r)
   k = q – p + 1
   if (i = k)
      return A[q]
  else if (i < k)
      return RandomizedSelect(A, p, q-1, i)
   else
      return RandomizedSelect(A, q+1, r, i-k)
```

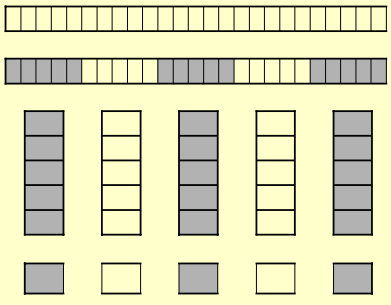
```
QuickSort(A, p, r)
   if (p < r) then
      q = Partition(A, p, r)
      QuickSort(A, p, q)
      QuickSort(A, q+1, r)

Partition(A, p, r)
  x = A[r]
  i = p-1
  for j = p to r-1 do
     if (A[j] <= x) then
        i++
        SWAP(A[i], A[j])
  SWAP(A[i+1], A[r])
  return i+1
```


Randomized Selection: Rewritten

```
RandomizedSelect(A, p, r, i)
   if (p = r) then
      return A[p]
   Pivot = A[random(p,r)]
   q = PivotPartition(A, p, r, Pivot)
   k = q – p + 1
   if (i = k)
      return A[q]
   else if (i < k)
      return RandomizedSelect(A, p, q-1, i)
   else
      return RandomizedSelect(A, q+1, r, i-k)
```

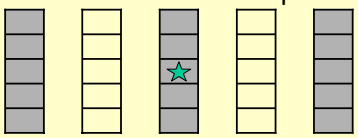

k-Selection & Median: Improved Algorithm

- Start with initial array



k-Selection & Median: Improved Algorithm(Cont'd)

- Use median of medians as pivot



- $T(n) < O(n) + T(n/5) + T(3n/4)$

Improved Selection

```
ImprovedSelect(A, p, r, i)
   if (p = r) then
      return A[p]
   else N = r – p + 1
   Partition A[p..r] into subsets of 5 elements and collect
   all the medians of the subsets in B[1..⌈(N/5)⌉].
   Pivot = ImprovedSelect (B, 1, ⌈N/5⌉, ⌈N/10⌉)
   q = PivotPartition (A, p, r, Pivot)
   k = q – p + 1
   if (i = k)
      return A[q]
   else if (i < k)
      return ImprovedSelect(A, p, q-1, i)
   else
      return ImprovedSelect(A, q+1, r, i-k)
```