Tue 02/09/16

Find Kth largest element

Problem : Given an integer input array A, find the value of the element that would be in slot i if we sorted A.

```
findRankKthElement(A,k) \\\
  Sort(A)               \\\ nlogn
  return A[k]           \\\
```

An O(n) strategy

```
find Rank(A,k)
  t = partition(A[0, ..., A.length-2], A[A.length-1])
  SWAP(A[t]), A[A.length])
  if k == t then return A[t]
  else if k < t then return findRankKthElement(A[0, .. , t-1], k)
  else return findRankKthElement(A[t+1, .. , A.length-1], k-t-1)
```

Time taken :

```
T(1) = 1
T(n) = 3n + 3 + T(n-1)
    ~= 3n^2

Good luck case:
T(1) = 1
T(n) = 3n + 3 + T(n/2)
T(n/2) = 3(n/2) + 3 + T(n/4)
T(n/4) = 3(n/4) + 3 + T(n/8)

T(n) = (3n+3) + 3(n/2 + 3) + T(n/4)
     = (3n+3) + 3(n/2 + 3) + 3(n/4 + 3) + T(n/8)
    => 3i + 3n(1 + 1/2 + 1/4 + ... 1/(2^(i-1))) + T(n/2^i)
    => 3i + 3(2) + T(n/2^i)
```

Solve for i such that T(n/2^i) = T(1)

i.e $n/^i = 1 \Rightarrow i = \log_2 n$

```
T(n) = 3log(2)(n) + 6n + T(1)
    ~= 6n
```

An O(n) strategy assuming partition always gives a good split

```
findRankKthElement(A, k)
  let m = arrayOfMediansOfGroupsOf5(A)
  let x = findRankKthElement(m, m.length-2)
  t = partition(A, x)
  if k<t return findRankKthElement(A[0, .. , t-1], k)
  else return findRankKthElement(A[t+1, .. , A.length-1], k-t)
```

Claim: The rank of x is always in the middle 3rd of the array

Proof:

```
Rank(x) >= 3n/10. The following element of A must always be smaller than
x:
    a: half of m
    b: 2 bonus elements per small elements in m
    c:
```