Divide and Conquer: Linear-Time Median

Median Problem

1. Given an unsorted list $A = [a_1, \ldots, a_n]$ of n numbers
   - Goal: Find the median of A
     - Median: $\text{ceil}(n/2)^{\text{th}}$ smallest
   - For odd $n = 2l + 1$
     - Median is the $(l+1)^{\text{st}}$ smallest
2. More general problem
   - Given an unsorted list A and integer k where 1 <= k <= n, find the k^{th} smallest element of A
   - Easy algorithm: Sort A and then output the k^{th} element
   - MergeSort takes O(nlogn) time
3. Now: O(n) time algorithm
   - Blum, Floyd, Pratt, Rivest, Tarjan from 1973

QuickSort

1. Divide and conquer: QuickSort style
   - QuickSort(A):
     - Choose a pivot P
     - Partition A into $A_{<p}$, $A_{=p}$, $A_{>p}$
     - Recursively sort $A_{<p}$ and $A_{>p}$
   - Challenge of QuickSort is choosing a good pivot
     - If we choose the largest or smallest element, one of the lists is of size n-1 and then the running time is O(n^2)
     - Good pivot is median, or something close to it
     - To find median, we only have to examine one of the sublists

Search Example

1. Example: A = [5, 2, 20, 17, 11, 13, 8, 9, 11]
   - Say p = 11
     - Less than p: [5, 2, 8, 9]
     - Equal to p: [11, 11,]
     - Greater than p: [20, 17, 13]
   - If k <= 4 then we want k^{th} smallest in $A_{<p}$
   - If 4 < k <= 6 then we output 11
   - If k > 6 then we want $(k\text{-}6)^{\text{th}}$ smallest in $A_{>p}$

QuickSelect

1. Choose a pivot p - How?
2. Partition A into $A_{<p}$, $A_{=p}$, $A_{>p}$
3. If k <= $|A_{<p}|$ then return(Select($A_{<p}$, k))
4. If $|A_{<p}| < k <= |A_{<p}| + |A_{=p}|$ then return(p)
5. If $k > |A_{<p}| + |A_{=p}|$ then return(Select($A_{>p}$, k - $|A_{<p}|$ - $|A_{=p}|$))

Simple Recurrence

1. What does the recurrence T(n) = T(n/2) + O(n) sovle to?
   - T(n) = O(log(n))
   - T(n) = O(n)
     - Correct
   - T(n) = O(nlog(n))

D&C: High-level Idea

1. Aim: O(n) running time
   - $T(n) = T(n/2) + O(n)$ is $O(n)$
     - Need: $p = \text{median}(A)$
   - Approximate median: Close to the median, but not exactly
     - Suppose we can find a pivot that lies between $n/4$ and $3n/4$
       * Not on either extreme
     - $T(n) = T(3n/4) + O(n)$, which is still $O(n)$
     - $T(n) = T(0.99n) + O(n)$, which is still $O(n)$
       * Require any constant less than 1

Goal: Good Pivot

1. Pivot p is **good** if $|A_{<p}| <= 3n/4$ and $|A_{>p}| <= 3n/4$
2. Goal: Find good pivot p in O(n) time
   - $T(n) = T(3n/4) + O(n) = O(n)$

Random Pivot

1. When in doubt, just act randomly
   - Let p be a random element of A
     - What's the probability that p is good?
       * Using the range $n/4$ to $3n/4$ as before, $P = 0.5$
   - We can spend O(n) time breaking the array into segments and determining if p is a good pivot
     - We expect to be able to find a pivot in O(n) time
   - We want an algorithm with guaranteed O(n) runtime

D&C: Recursive Pivot

1. Aim: Find a good pivot in O(n) time
   - $T(n) = T(3n/4) + O(n) = O(n)$
     - Slack: $T(0.24n)$
   - $T(n) = T(3n/4) + T(n/5) + O(n)$
     - $3/4 + 1/5 < 1 = O(n)$
   - Choose a subset S of A where $|S| < n/5$
     - Set $P = \text{Median}(S)$

Representative Sample

1. Naive selection of S
   - Let $S = [a_1, \ldots, a_{n/5}]$ = First $n/5$ elements of A
     - Set $p = \text{Median}(S)$
   - Is p_a a good pivot? No!
     - Suppose A is sorted
     - $S = n/5$ smallest elements of A
     - $p = n/10^{th}$ smallest element
     - $|A_{>p}| <= 9n/10$
       * This indicates that the subsets resulting from this pivot will be too large

Recursive Representative Sample

1. Choose S that is "representative" of A
   - Want: median(S) approximate median(A)
   - For each x in S, a few elements of A are <= x and a few are >= x

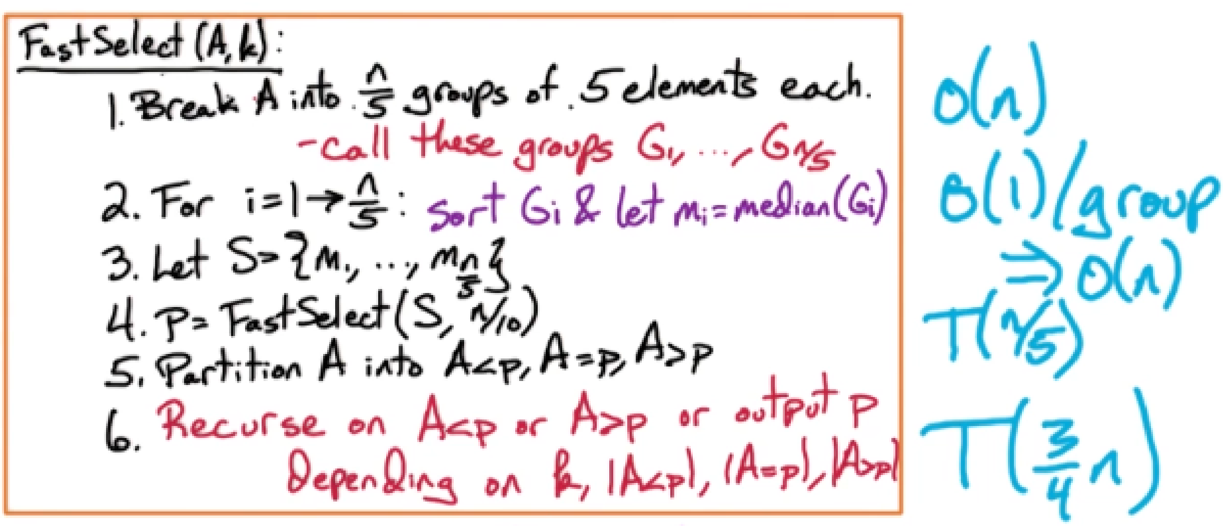
2. Break A into n/5 groups of 5 elements each
   - G = $\{x_1, x_2, x_3, x_4,$ x~5$\}$
     - Sort $x_1 <= x_2 <= x_3 <= x_4 <= x_5$
       - x_3 is the median of G
     - Sorting G takes O(1) time
       - This is because G is always 5 elements; time to sort it does not scale with n

Median: Pseudocode

1. FastSelect(A,k):
   - Input: unsorted A and integer k where 1 <= k <= n
   - Output: k^{th} smallest of A
2. Pseudocode
   - Break A into n/5 groups, $G_1, G_2, \ldots, G_{n/5}$
   - For i = 1 -> n/5:
     - sort(G_i) and let m_i = median(G_i)
   - Let S = $\{m_1, m_2, \ldots, m_{n/5}\}$
   - p = FastSelect(S,n/10)
   - Partition A into $A_{<p}$, $A_{=p}$, $A_{>p}$
   - If k <= $|A_{<p}|$ then return(FastSelect($A_{<p}$, k))
   - If k > $|A_{<p}| + |A_{=p}|$ then return(FastSelect($A_{>p}$, k - $|A_{<p}|$ - $|A_{=p}|$))
   - Else output p

Median: Running Time

1. Claim: p is a good pivot
   - T(n) = T(3n/4) + T(n/5) + O(n) = O(n)
     - Key: 3/4 + 1/5 < 1

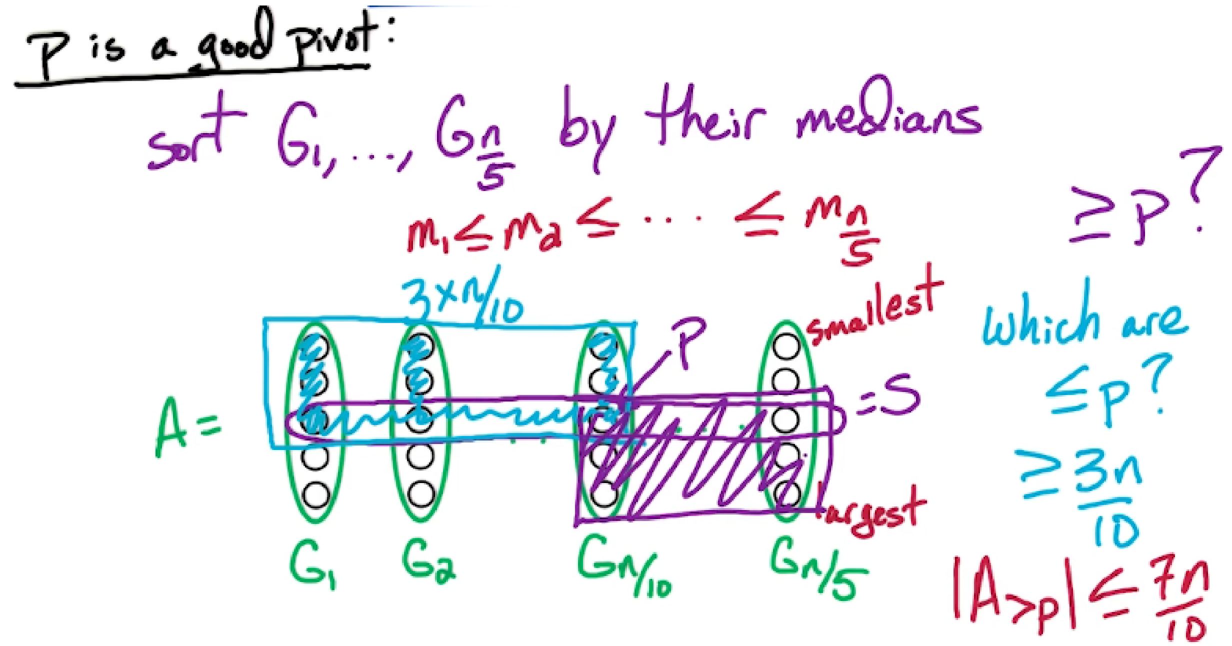


Fast Select Running Time

Linear-Time Median Correctness

1. p is a good pivot:
   - Sort $G_1, \ldots, G_{n/5}$ by their medians
     - $m_1 <= m_2 <= \ldots <= m_{n/5}$

- Which elements are <= p?
  - Guaranteed that 3 * n/10 elements are <= p
  - $|A_{>p}|$ <= 7n/10
    - We needed to guarantee that $|A_{>p}|$ <= 3n/4
- Which elements are >= p?
  - Guaranteed that 3 * n/10 elements are >= p
  - Same logic as above



Fast Select Pivot Selection

HW: Groups of 3? 7?

1. Running time for groups of 3 or 7 elements
   - What is the recurrence in these cases?
   - For 3, this does not reduce the subproblem enough, so running time is O(nlogn)
   - For n = 7, 9, . . . , the algorithm works in O(n) times, but increases a constant factor
   - Makes sense to use odd sizes because it simplifies the median