Next topic: Probabilistic Analysis, Randomized Algorithm.

Selection Problem: Find the $k^{th}$ smallest number in a list of $n$ numbers (Assume distinct numbers).

ex: 19, 13, 18, 12, 17, 16, 15, $k = 6 \Rightarrow \text{ans} = 18$

Simple solution?
- Sort list, return $k^{th}$ element. $\Rightarrow O(n \log n)$

can actually be solved in $O(n)$!
Problem is similar to finding min/max
Seems harder, but is surprisingly, (computationally) just as easy!

Algorithm is similar to Quicksort

Quick Quicksort review:

ex: 19, 13, 18, 12, 17, 16, 15

1) Choose a pivot, ex 15
2) Partition list around pivot into $L, G$

\[
\begin{array}{c}
\text{original list:} \\
19, 13, 18, 12, 17, 16, 15 \\
\text{pivot: 15} \\
\text{after partitioning:} \\
\text{L: 12, 13} \\
\text{G: 18, 16, 17, 19} \\
\end{array}
\]

ex: 19, 13, 18, 12, 17, 16, 15

\[\text{(move right until A[i] > pivot)} \quad \text{(move left until A[i] < pivot)}\]
When \( A[i] > \text{pivot} \) and \( A[j] < \text{pivot} \), swap. Eventually...

(3) Recursively call QuickSort on \( L \) and \( G \)

Time: \( 2 \times \text{Time(Partition)} \times \# \text{Partitions} \)

\[ = O(n) \times ? \]

\# Partitions:
- worst-case: \( O(n^2) \): if pivot is always largest/smallest
- \( \text{ex: pivot} = 19 \Rightarrow 13 18 12 16 17 15 19 \)

Each partition is around only 1 element, so list size is decreased
- list by 1 each time: so \( n-1 \) partitions \( \Rightarrow \) total time: \( O(n^2) \)

How to avoid:
- more [random] choice of pivot
  - \( \text{ex: pivot} = 15 \)
  - Typically randomly chosen pivot will be a number whose value
    is in middle.

Can also achieve with Median of 3: \( \text{ex: } \text{Medof3}(19, 12, 15) = 15 \)

average case: pivot chosen so each partition splits list
  roughly in half: \( O(\log n) \) partitions \( \Rightarrow \) total time \( \Rightarrow O(n \log n) \).
ex. \( cn \)
\[ \begin{array}{c}
\text{Partitioning at each} \\
\text{level: } O(n) \\
\end{array} \]
\[ \begin{array}{c}
\text{# levels: } O(\log n) \\
(n \text{ to } 1, \frac{1}{2^n} \text{ each time}).
\end{array} \]

Anything useful here for Selection Problem?

Notice: After partition, all elements:
- left of pivot < pivot
- right > pivot

Can find the \( k^\text{th} \) smallest by looking at size of \( L \) (or \( G \))

Specifically:
- if \( |L| = k - 1 \), pivot is \( k^\text{th} \) smallest \( (k-1 = 4) \)
- if \( |L| > k - 1 \), \( k^\text{th} \) smallest in \( L \) \( \| L \| > k \)
- if \( |L| < k - 1 \), \( k^\text{th} \) is \( \leq \) \( G \)

\[ \begin{array}{c}
11 = 4 \\
11 = 6 \\
14 = 2
\end{array} \]

Why is this faster (i.e. \( O(n) \)) than \( O(n \log n) \) (assuming
"good" pivot)?

⇒ After a partition, we consider only \( \frac{1}{2} \) the list (vs. both \( \frac{1}{2} \)'s)
1. Choose a pivot, p, at random from A.
2. Partition A into 2 sublists L, G:
   L = all elements < p
   G = all elements > p.
3. If |L| = |k - 1|: // k-1 elements are smaller than p.
   return p
else if (|L| > |k - 1|): // |L| > k.
   return Rand-Select (L, k).
else // (|L| < |k - 1|): G contains k-th smallest.
   return Rand-Select (G, 2 * k - |L| - 1)

\[ K \text{ needs to change.} \] update k by removing all elements in L and the pivot

\[ A = 19 \ 13 \ 18 \ 12 \ 17 \ 16 \ 15 \] k = 6, Ans. 18

p = \underbrace{15} \ 12 \ 13 \ 15 \ 19 \ 17 \ 16 \ 18 \ \underbrace{k - 1 = 5}
\begin{align*}
L &= 15 \ 12 \ 13 \ 15 \ \underbrace{19} \ 17 \ 16 \ 18
G &= \underbrace{15} \ \overbrace{12} \ 13 \ 15
\end{align*}

|L| = 2 < k-1

Now recurse on G to find 6 - 2 - 1 = 3 \text{rd} smallest in G \rightarrow 12
Running Time of Rand-Select

worst case \text{ ~ Quicksort} \quad \text{(pivot always largest/smallest)}

\rightarrow \text{partition } O(n) \text{ times, each partition takes } O(n) \Rightarrow O(n^2)

Average-case

Intuition: every time we find pivot and partition the list, we reduce the size of the sublist to consider.

Informal ways to see runtime:

- Suppose \( n = 2^p \) (for \( p \geq 0 \)) split list in exactly \( \frac{\sqrt{2}}{2} \) in each partition.

\[
\begin{align*}
\frac{c_n}{2} + \frac{c_n}{4} + \frac{c_n}{8} + \ldots & = \sum_{i=0}^{\infty} \frac{c_n}{2^i} = 2c_n \\
& \quad \text{by geometric series.}
\end{align*}
\]

- With recurrence:

Again suppose \( n = 2^p \) split list exactly in \( \frac{\sqrt{2}}{2} \) in each partition

\[
T(n) = T(n/2) + cn
\]

Masters: \( a = 1 \), \( b = 2 \), \( c = 1 \), \( a < b^c \)

Root dominates \( \Rightarrow \Theta(n^c) = \Theta(n) \)

Formal way: can show average case is \( O(n) \) even when splits are not exactly even.

Thom: Rand-Select runs in expected (average) time \( O(n) \).
Review expected value.

Interested in the expected (i.e., average) value of a variable - a random variable that can take on multiple values.

r.v. $X$ takes on possible values $x_1, x_2, x_3, \ldots, x_n$.

Expected value of $X = x_1 \cdot \Pr(X = x_1) + x_2 \cdot \Pr(X = x_2) + \ldots + x_n \cdot \Pr(X = x_n)$

\[ = \sum_{i=1}^{n} x_i \cdot \Pr(X = x_i) \]

\[ = \mathbb{E}[X]. \]

Ex: $X$ = roll of 6-sided die

\[ \mathbb{E}[X] = 1 \cdot \Pr(1) + 2 \cdot \Pr(2) + \ldots + 6 \cdot \Pr(6) = 3.5. \]

Our r.v. is $T(n)$.

We can say $T(n)$ takes on 2 values: $5$ "lucky", "unlucky"?

Recall: partition splits list into $L, G$.
- We recurse on either $L$ or $G$.

For informal analysis, saw that if partition splits list exactly in $\frac{1}{2}$, we get $O(n)$ time.
The more evenly the split the better.

\( T(n) \) is:

"lucky" - if partition splits \( L, G \) such that
\( L, G \) both have \( < 3n/4 \) elements
\( |L| = n/2 \), \( |G| = n/2 \)
\( |L| = n/3 \), \( |G| = 2n/3 \)

"unlucky" - if partition splits \( L, G \) such that
either \( L \) or \( G \) has \( > 3n/4 \) elements
\( |L| = 4n/5 \), \( |G| = n/5 \)

(Now we have possible values for our r.v. \( T(n) \) so let's write \( E[T(n)] \))

\[ E[T(n)] = \text{expected time to run Rand-Select on list } A \text{ of size } n \]
\[ = E[\text{time to partition } A] + E[\text{time to recurse on } L \text{ or } G] \]
\[ \leq n + E[\text{time to recurse on } L \text{ or } G] \]

Note: this is worst-case time and not just expected.

Want a big-O, so we'll be pessimistic

\( \Rightarrow \) Assume we recurse on larger of \( L, G \).
\[ T(n) \leq n + T(\text{max}(|L|, |G|)) \]

\[
\uparrow
\]

let's find this.

2 possible "values" for \( T(n) \): lucky, unlucky

\[ T(\text{max}(|L|, |G|)) = \Pr(\ T(n) \text{ is lucky}) \times \text{time when lucky} + \Pr(\ T(n) \text{ is unlucky}) \times \text{time when unlucky} \]

\[ \Pr(\ T(n) \text{ is lucky}) \]

\[ \text{depends on where pivot ends up after partition.} \]

Consider list split into 4:

\[ \begin{array}{cccc}
  & X & \checkmark & \checkmark & X \\
\end{array} \]

\[ \text{if pivot ends up here, "unlucky"} \]

\[ \text{if pivot ends up here, "lucky"} \]

\[ \Pr(\ T(n) \text{ is lucky}) = \Pr(\ T(n) \text{ is unlucky}) = \frac{1}{2}. \]

Back to \( T(n) \):

\[ T(n) \leq n + \frac{1}{2} \times \text{time when lucky} + \frac{1}{2} \times \text{time when unlucky} \]

\[ \leq n + \frac{1}{2} \times T(< \frac{3n}{4}) + \frac{1}{2} \times T(n-1) \]

\[ \uparrow \]

Want to get this in a form to use Master's.
\[ T(n) \leq n + \frac{1}{2} T\left(\frac{3n}{4}\right) + \frac{1}{2} T(n) \]

Multiply both sides by 2:

\[ 2T(n) \leq 2n + T\left(\frac{3n}{4}\right) + T(n) \]

\[ T(n) \leq T\left(\frac{3n}{4}\right) + 2n \]

**Master's**: \( a = 1 \), \( b = \frac{4}{3} \), \( i = 1 \)

\[ a < b^i \quad \text{Root dominates} \Rightarrow \Theta(n^i) \]

If our equation was \( T(n) \leq \ldots \)
then we could say \( T(n) = \Theta(n^i) \) but since our equation is \( T(n) \leq \ldots \), we can say:

\[ T(n) = O(n) \]

**Important Note**:
We showed average case is \( O(n) \)
Can solve in worst-case \( O(n) \) (Ch 9.3)
using Median-of-Medians to find pivot
- Better worst-case time but
- Much more complicated!
- Almost always slower in practice
- Has \( O(n) \) time but \( c \) is high!