ex. (3) \[ T(n) = \begin{cases} d & \text{for } n = 1 \\ 2T\left(\frac{n}{2}\right) + 1 & \text{for } n > 2 \end{cases} \]

1. Guess \( O(1) \), so \( f(n) = 1 \).

2. (a) Base Case:
   Find \( c \) s.t. \( T(2) \leq cf(2) \)
   \[ T(2) \leq c \]
   \[ 2T\left(\frac{2}{2}\right) \leq c \]
   \[ 2d + 1 \leq c \]

   \( \therefore \) any \( c \geq 2d + 1 \) works for Base Case.

(b) Assume true up to \( k \)
   \[ T(k) \leq cf(k) \leq c \]
   Set \( k = \frac{n}{2} \)

\( \mathbf{\therefore} T\left(\frac{n}{2}\right) \leq c \)

(c) Find \( c > 0 \) s.t. \( T(n) \leq cf(n) \) for all \( n > 0 \)

From rec.: \( T(n) = 2T\left(\frac{n}{2}\right) \)
From \((b)\) \[ \leq 2c \]

Find \( c > 0 \) s.t. \( 2c \leq cf(n) = c \)

\( \therefore \) No \( c \) exists!
Let's put this all together

Recursive function:

\[
function(A, x) \quad // A[1..n] is list, x is some value
\]

\[
\quad \text{if } |A| = 0 \\
\quad \quad \text{return } \text{false}
\]

\[
\quad \text{else if } A[1] = x \\
\quad \quad \text{return } \text{true}
\]

\[
\quad \text{else} \\
\quad \quad \text{return } (A[2..n], x)
\]

Recurrence:

\[
T(n) = \begin{cases} 
T(n-1) + d & \text{for } n \geq 1 \\
\quad d' & \text{for } n = 0, \ldots, 1
\end{cases}
\]

Solve? Master's? Does not apply!

Tree:

\[
T(n-1) \quad d \quad d \\
\quad \text{# levels?} \\
T(n-2) \quad d
\]

Time: $d^n = \Theta(n)$
Substitution Method for $T(n) = T(n-1) + d$

1. **Guess** $f(n) = n$

Show $\exists c > 0$ s.t. $T(n) \leq c \cdot n$

2. (a) **Base Case**

   Smallest $n$ with recursion? $n = 1$.

   Find $c$ s.t. $T(1) \leq c(1)$, $T(2) \leq c(2)$

   
   \[ T(1-1) + d \leq c, \quad T(2-1) + d \leq 2c \]

   
   \[ 2d \leq c \quad 2d \leq 2c \]

(b) **Ind. Hyp.** $T(k) \leq ck$ for $k < n$

Choose $k = n-1$. $\Rightarrow T(n-1) \leq c(n-1)$

(c) $T(n) = T(n-1) + d$ (from recurrence)

\[ \leq c(n-1) + d \] (from (b))

\[ T(n) \leq cn - c + d \]

When is $\frac{d}{c} \leq cn$?

\[ cp - c + d \leq cn \]

\[ d \leq c \]

For all $c \geq d$

Final: $c$: any value $\geq 2d$
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 => 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 (overview on slides)

ex: 19 13 18 12 17 16 15

"move pivot out of the way"

12 13 15 19 17 16 18

move i right,

12 13 15 19 17 16 18

move j left until

\[ \begin{array}{c}
Q5C \quad Q5C
\end{array} \]

\[ \begin{array}{c}
L > \text{pivot and } A_{i} \quad A_{j} < \text{pivot}
\end{array} \]

(1) Choose a pivot, ex 15

"swap"
(2) Partition list around pivot into L, G
   "How?" Go to previous page.

(3) Recursively call quicksort on L and G

Time: \( \theta(\text{Partition}) \times \# \text{Partitions} \)
   \( = \big{O(n^2)} \times ? \)

\# Partitions:
   worst-case: \( \big{O(n^2)} \): if pivot is always largest/smallest

ex: pivot = 19 \( \rightarrow \) 13, 18, 12, 16, 17, 15, 19
    (largest)

    (pivot)

pivot = 12 \( \rightarrow \) 12, 18, 13, 16, 17, 15
    (smallest)

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

How to avoid?

- more random choice of pivot. ex: pivot = 15,
  Typically randomly chosen pivot will be a number whose value
  is in middle.

Can also achieve with median of 3: 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 \( = O(n \log n) \).
ex: \( cn \)
\[
\begin{array}{c}
\text{\( \frac{cn}{2} \)} \\
\text{\( \frac{cn}{2} \)} \\
\text{\( \frac{cn}{2} \)} \\
\text{\( \frac{cn}{2} \)} \\
\text{\( \frac{cn}{2} \)} \\
\end{array}
\]

Partitioning at each level: \( O(n) \)

\# levels: \( O(\log n) \)

\( n \) to \( 1, \frac{1}{2}, \ldots \) each time.

Anything useful here for Selection Problem?

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

Can find the \( k \)-th smallest by looking at size of \( L \) (or \( G \))

Specifically:
- if \( |L| = k-1 \), pivot is \( k \)-th smallest \( (k-1=4) \)
- if \( |L| > k-1 \), \( k \)-th smallest in \( L \) // \( |L| \geq k \)
- if \( |L| < k-1 \), \( k \)-th smallest \( \leq G \)

\* \( k \)-th smallest

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

\( \Rightarrow \) After a partition, we consider only \( \frac{1}{2} \) the list (vs. both \( \frac{1}{2} \))
Rand-Select (A, k) // Returns k\textsuperscript{th} smallest number in list
// A of distinct integers

1. Choose a pivot, p, at random from A.
2. Partition A into 2 sublists L, G:
   \[L = \text{all elements} < p\]
   \[G = \text{all elements} > p\]
3. If (\|L\| = k-1), // k-1 elements are smaller than p.
   \[\text{return } p\]
   else if (\|L\| > k-1) // (\|L\| < k) L contains the k\textsuperscript{th} smallest
   \[\text{return Rand-Select (L, k)}\]
   else // (\|L\| < k-1) G contains k\textsuperscript{th} smallest
   \[\text{return Rand-Select (G, 2 \cdot k - \|L\| - 1)}\]

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

\[\text{ex: } A = 19 \ 13 \ 18 \ 12 \ 17 \ 16 \ 15 \ \ k = 6, \ \text{Ans: } 18\]

\(p=15\)
\(k=5\)
\(\underbrace{12\ 13\ 15}_{L} \ 19\ 17\ 16\ 18\)
\(\underbrace{19\ 17\ 16\ 18}_{G}\)
\(\|L\| = 2 < k-1\)

Now recurse on G to find \(6-2-1 = 3\text{^rd smallest}\) in G \(\Rightarrow 1\&\)