Class copy

Divide + Conquer Algorithms:

n= input of original

Step 1: Break my problem up into a subproblems
↳ a number!

Step 2: Use the recursion func on those sub-problems

Step 3: Put the answer together

f(n) we see in $T(n) = \sum T(k) + f(n)$
↑
subproblems

In-Class Exercise 05

CSCI 432

13 September 2023

Name(s):

If not handing in as one group, who did you work with today?

To solve this problem, I use a sub-problems of size $\frac{n}{b}$, and it takes f(n)-time to divide up into these sub-problems + bring them back together.

Master's Theorem

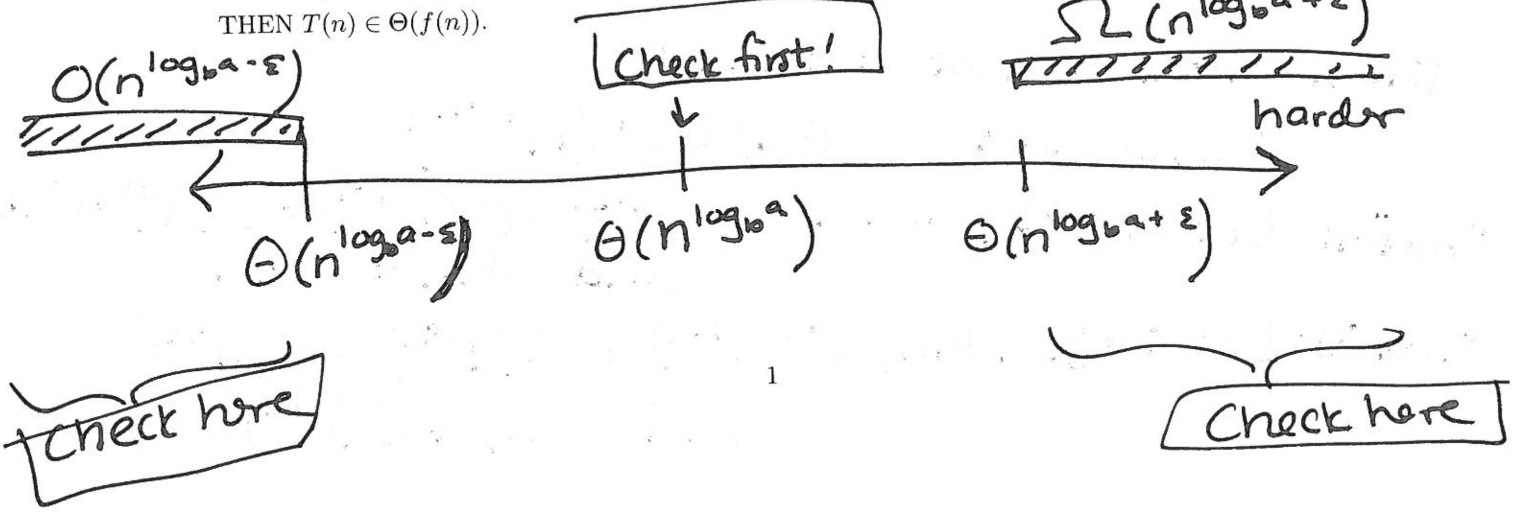
Master's theorem allows us to quickly solve recurrence relations of the form: a, b are constants ($\in \mathbb{Z}_+$)

$$\boxed{T(n) = aT(n/b) + f(n),}$$

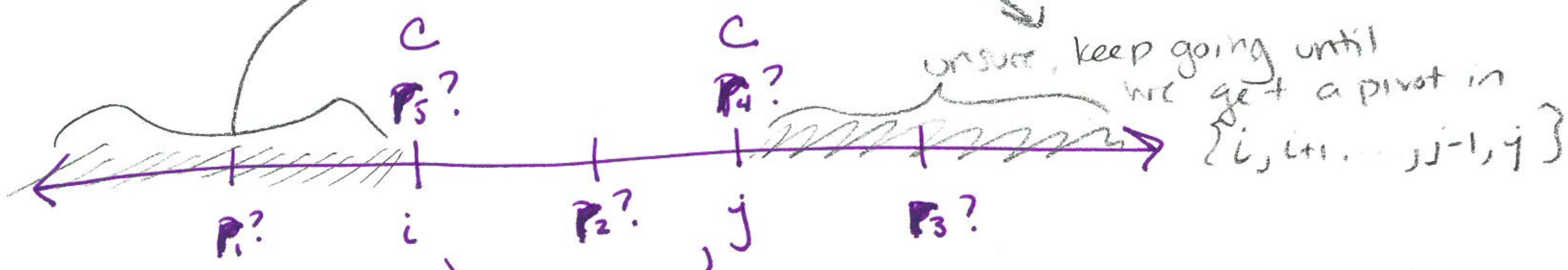
where $a, b \in \mathbb{N}$ such that $a \geq 1$ and $b > 0$ and $f(n)$ is asymptotically positive. Then, we can determine the closed-form of $T(n)$ as follows:

1. IF there exists $\varepsilon \in \mathbb{R}_+$ such that $f(n) \in O(n^{\log_b a-\varepsilon})$, THEN $T(n) \in \Theta(n^{\log_b a})$.
2. IF ~~there exists $\varepsilon \in \mathbb{R}_+$ such that~~ $f(n) \in \Theta(n^{\log_b a})$, THEN $T(n) \in \Theta(n^{\log_b a} \log n)$.
3. IF
   (a) there exists $\varepsilon \in \mathbb{R}_+$ such that $f(n) \in \Omega(n^{\log_b a+\varepsilon})$ and ← same as b4
   (b) there exists $c \in (0, 1)$ and $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$, the following holds: $af(n/b) \leq cf(n)$, ← regularity condition

   THEN $T(n) \in \Theta(f(n))$.



15 Sept 2023



Never compare $i + j$

2 ways they get compared out of $j-i+1$ ways we find out if they get compared.

```
QuickSort(A[1..n]):
1: if (n > 1)
2:        Choose a (random) pivot element A[p]
3:        r ← Partition(A, p)
4:        QuickSort(A[1..r − 1])     ⟨⟨Recurse!⟩⟩
5:        QuickSort(A[r + 1..n])     ⟨⟨Recurse!⟩⟩
6: return
```

```
Partition(A[1..n], p):
1: swap A[p] ↔ A[n]
2: ℓ ← 0                       ⟨⟨#items < pivot⟩⟩
3: for i ← 1 to n − 1
4:        if A[i] < A[n]
5:              ℓ ← ℓ + 1
6:              swap A[ℓ] ↔ A[i]
7: swap A[n] ↔ A[ℓ + 1]
8: return ℓ + 1
```

Figure 1.8. Quicksort

QuickSort $T(n) =$ RT of QS on $|A| = n$

Runtime: Line 1 $\Theta(1)$
2 $\Theta(1)$
3 $\Theta(n)$
4 $T(r-1)$
5 $T(n-r)$
6 $\Theta(1)$

$$T(n) = T(r-1) + T(n-r) + \Theta(n)$$

Worst-case is:

$$T(n) = \max_{1 \le i \le n} \{T(i-1) + T(n-i)\} + \Theta(n)$$

↳ Runtime?

Line 1, 2 $\Big\}$ $\Theta(1)$

3, 4, 5, 6 $\Big]$ For loop $\times (n-1)$ each line $\Theta(1)$ ⇓ $\Theta(n)$ total for these lines

7, 8 $\Big\}$ $\Theta(1)$

$$\Theta(n) + 2\Theta(1) = \boxed{\Theta(n)}$$

1

15 September 2023

runtime of Quicksort (cont'd)

$$T(n) = \max_{1 \le i \le n} \{T(i-1) + T(n-i)\} + \Theta(n)$$

on extremes: $i = 1$ or n

$$T(n) = T(0) + T(n-1) + \Theta(n) \in \boxed{\Theta(n^2)}$$ ← worst-case behavior

in middle:

$$T(n) \approx T(\tfrac{n}{2}) + T(n/2) + \Theta(n)$$
$$= 2T(n/2) + \Theta(n) \quad \in \Theta(n \log n)$$ ← wouldn't this be nice

But,... let's consider the average case analysis

Note: $T(n)$ is the same (asymptotically) as counting the number of comparisons.

Random variable: $X_{ij} = \begin{cases} 1, & \text{if els } i+j \text{ are compared} \\ 0, & \text{otherwise} \end{cases}$

The total # of comparisons (& hence the RT) is:

$$X = \sum_{i=1}^{n} \sum_{j=i+1}^{n} X_{ij}$$

What is the expected value of this?

$$\mathbb{E}(X) = \mathbb{E}\left(\sum_{i=1}^{n} \sum_{j=i+1}^{n} X_{ij}\right)$$

$$= \sum_{i=1}^{n} \sum_{j=i+1}^{n} \mathbb{E}(X_{ij}) \quad \text{by linearity of expectation}$$

$$= \sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \quad \text{(see "timeline")}$$

$$= \sum_{i=1}^{n} 2 \cdot \left(\frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n-i+1}\right) \le \sum_{i=1}^{n} 2 H_n = 2n H_n$$

$$\le 2n \ln(n) = \Theta(n \log n).$$

2

Recall (from long ago) the Harmonic Numbers:

$$\sum_{i=1}^{n} 1/i = H_n \in [\ln(n), \ln(n)+1]$$
$$\in \Theta(\ln(n)) = \Theta(\log n)$$

Geometric Series

$$\sum_{i=1}^{\infty} a \cdot r^i$$

→ if $r \geq 1$, diverges

→ if $0 < r < 1$, it converges to $a\left(\frac{1}{1-r}\right)$.

Why is QS correct? Use our recursion invariants?

① State our assumptions needed for a valid call to QS.

a) A is an array of real #s

b) n is the length of A

② ~~[illegible]~~ State the Recursion Invariant:

QS(A) executes correctly if:

a) A is sorted from smallest to largest

③ INITIALIZATION

- identify our base case: $n=0$
  (can't make a recursive call in our base case)

Proof:

Since $n=0$, A is empty.

In the code, I just return. Now I'm done.

Since A is still empty, (2a) is vacuously true. ∎

Note: this base case might not be the 1st call.
There could be another "A" that our empty array is a subarray of.

Hur-ray! INIT cases are almost always this "easy".

(4) MAINTENANCE: Assuming the recursion fairy is always correct, we prove that a general call is correct.

Let $\boxed{n_1 \geq 0}$.

Assume the recursion fairy works for QS(A) when $|A| \leq n_1$. That is (restate RI), after QS(A) executes, A is sorted from smallest to largest.

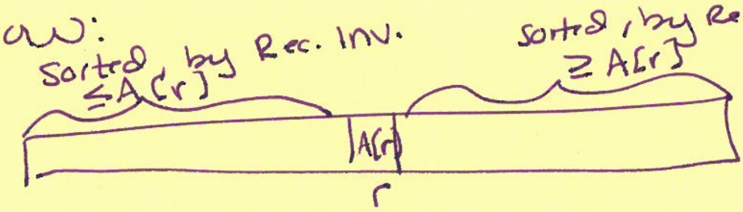
Now, we're given a call QS(A), where $|A| = n_1 + 1$ (ie, slightly bigger than my recursion fairy can handle, so I've gotta do this one)

Since this is a valid call, I know by (1) that A is an array of real #s.

Since $n_1 \geq 0 \Rightarrow n_1 + 1 \geq 1$. Thus, we evaluate TRUE on line 1. p is our pivot index chosen on line 3. Since $\boxed{\text{we proved}}$ PARTITION (A, p) will return r, the new index of A[p] from the input A in the new order of A, and that A[1 ... r-1] are all $\leq$ A[r] and A[r+1 ... n] are all $\geq$ A[r] and A has the same #s, just in a diff order than initially given.

well, not yet, but H2...

Then, the RF solves lines 4 + 5 for me, so I know:



∴ the whole array A is sorted!
(which is exactly what I wanted to prove from 2a)

18 Sept 202

QUICK SELECT

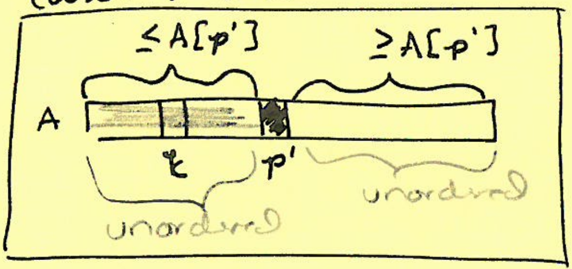
input: ① An array A of length n, not sorted from smallest A to largest A[n
② An index $k \in \underline{n} = \{1, 2, \dots, n\}$

output: The k^{th} smallest element of A

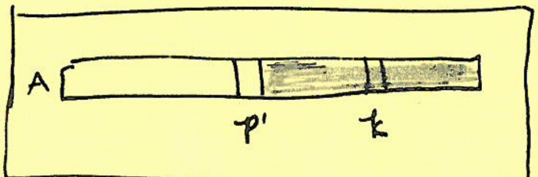
QUICK SELECT (A[1...n], k)

```
if n = 1          // k ∈ {1} => k = 1      } base case,
 | return A[1]                              } handled directly
endif
Choose a pivot p ∈ n.   // can be random, maybe not
p' ← PARTITION (A, p)
if p' = k
 | return A[p']
else if p' > k
 | return QUICKSELECT (A[1...p'-1], k)
else // p' < k
 | return QUICKSELECT (A[p'+1,...,n], k-p')
end else
```

case: $p' > k$



case: $p' < k$



Exercise: In groups, do the following:

① Create an array of length 15

② Follow this algorithm, diving into the recursions (today, the recursion fairy is napping)

③ What is the worst-case runtime? Why?

PARTITION (A, k)

→ input: Array A of length n
index $k \in \underline{n}$

→ output: ~~rearrange A AS such~~
rearranges A so that
① all values $\le A^{in}[k]$ ~~are to~~
are to the "left" of $A[k]$
② all values $\ge A^{in}[k]$ are to the "right" of $A[k]$
③ returns new index of $A[k]$.

ex: $[\pi, 6, 30, 18, 2.5]$, $k = 2$
1, 2, 3, 4, 5

new A: $[\underbrace{\pi, 2.5}_{\le A[k_{in}]}, \boxed{\underset{A[k]_{input}}{6}}, \underbrace{30, 18}_{\ge A^{input}[k]}]$

new index is 3

Worst-case runtime:

Let $T(n)$ denote the worst-case RT on input A of size n.

$$T(n) = \begin{cases} \overbrace{\Theta(1)}^{\text{end at line 2.}}, & n = 1 \\ \underbrace{\Theta(1)}_{\text{Lin 1 check}} + \underbrace{\Theta(1)}_{\text{Line 4, choose pivot}} + \underbrace{\Theta(n)}_{\text{Line 5}} + \max \begin{cases} \Theta(1) \text{ Lin} \\ T(p'-1) \\ T(n-p \end{cases} \end{cases}$$

there's still a choice of p

Let's simplify ~~restate worst p's choice~~

$$T(n) = \Theta(n) + \max \{T(p'-1), T(n-p')\}$$

Choose worst p':

$$T(n) = \Theta(n) + \max_{p' \in \underline{n}} \max \{T(p'-1), T(n-p')\}$$

We've seen this before!
IN The worst case, just like QSort,

$$T(n) = \Theta(n) + T(n-1) \in \Theta(n^2)$$

But, this algo: ① sort A
② return A[k]

runs in $\Theta(n \log n)$ worst case time!

What if... we could always choose a pivot such that our recurrence relation becomes

$$T(n) = T(n/b) + \Theta(n)$$

where $b > 1$

(ie. guarantee we slice off a % of the input

Using Master's:

$$\boxed{\begin{array}{l} a=1, \ b=b \\ f(n) \in \Theta(n) \end{array}} \Rightarrow \log_b a = \log_b 1 = 0$$

variables

compare $f(n)$ to $\Theta(n^{\log_b a})$

ie, $\Theta(n) >$ to $\Theta(n^0) = \Theta(1)$

$\Rightarrow$ Let's try case 3

(a) Find ε such that

$$\Theta(n) \in \Omega(n^{0+\varepsilon})$$

$\varepsilon = 1$ works!

(b) $c = 0.75$ and $n_0 = 1$

By case 3, $T(n) \in \Theta(n)$

20 Sept 2023

First pass of Quick select,
the worst-case recurrence relation was:

$$T(n) = T(n-1) + \Theta(n)$$

$$\Rightarrow T(n) \in \Theta(n^2).$$

What if I can guarantee my partition element "shaves off" a %-age each time?
Then, recursion would be

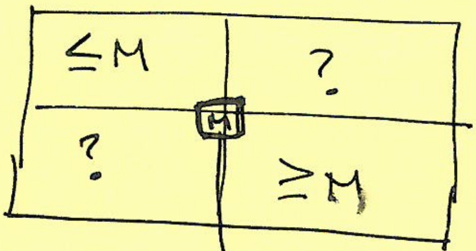
$$T(n) = T(n/b) + \Theta(n)$$

$$\Rightarrow T(n) \in \Theta(n \log n) \text{ by Master's Theor}$$

So, we will change line 4 from:

4: Choose a pivot $p \in \underline{n}$

to the "Median of Medians" approach



At worst, every elt in [?] regions in the size we recurse on

M = median of medians

Median of Medians approach

- Have: A = an array of length n, unsorted

  Want: central-ish pivot

  (at least $\Theta(n/b)$ are on Both sides of it

  $\uparrow$ $\in \mathbb{R}$, ≥ 1

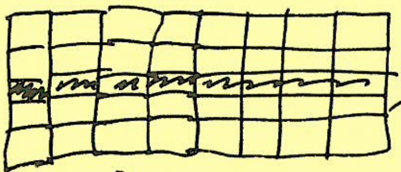
How?

$\Theta(n)$ ① Divide A into $k = \lceil n/5 \rceil$ groups of size 5.

$k \cdot \Theta(1)$ ② For each group, calculate the median m_i

③ Find the median by calling

$T(k)$ QUICK SELECT $\left(\{m_i\}_{i=1}^{k}, k/2 \right)$



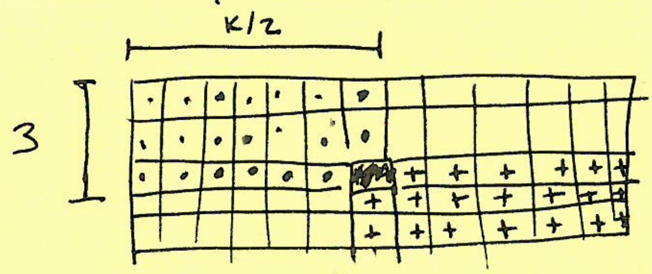
} less than ▨ directly below it

} bigger than ▨ directly above it

$\uparrow$ i^{th} group is a column of values

> Imagine: re-arranged so sorted by median element

Then, I know:



smallest med $\longrightarrow$ largest median

every $\boxed{+}$ is bigger than my median of medians

▨ $\geq$ $\boxed{\cdot}$

And there are

$\left(\frac{k}{2}\right) \cdot 3 = \left(\frac{n/5}{2} \cdot 3\right)$

$= \frac{3n}{10} - 1$ elem

$\Rightarrow$ there are $\Theta(\frac{7}{10}$ elements in $\square$

Lets' Revisit The recurrence relation (using Line #s from Monday)

$$T(n) = \Theta(1) \quad \text{, lines 1-3}$$

$$+ \Theta(n) + \Theta(n/5) + T(n/5) \quad \text{, new line 4}$$

$$+ T\left(\tfrac{7}{10}n\right) \quad \text{, new worst-case recurrence in lines 8-12}$$

$$= T\left(\tfrac{7}{10}n\right) + T\left(\tfrac{n}{5}\right) + \Theta(n)$$

But wait! Now there are 2 recurrence relations
Let's look at that recursion tree

"cost of le

$\cancel{T(n)}\ \Theta(n) \longrightarrow \Theta(n)$

$\cancel{T(\tfrac{7}{10}n)}\ \Theta(7/10\, n) \qquad \cancel{T(\tfrac{n}{5})}\ \Theta(n/5) \longrightarrow \Theta\left(\tfrac{9}{10}n\right)$

$T\left(\left(\tfrac{7}{10}\right)^2 n\right) \quad T\left(\left(\tfrac{7}{10}\right)\left(\tfrac{1}{5}\right)n\right) \quad T\left(\left(\tfrac{7}{10}\right)\left(\tfrac{1}{5}\right)n\right) \quad T\left(\left(\tfrac{n}{5}\right)^2 n\right) \rightarrow \Theta\left(\left(\tfrac{9}{10}\right)^2 n\right)$

$\vdots$

$$T(n) \leq c \cdot n \sum_{i=0}^{\infty} \left(\tfrac{9}{10}\right)^i \text{, a geometric series} \qquad l^{th} \text{ level: } \Theta\left(\left(\tfrac{9}{10}\right)^l n\right)$$

$$= c \cdot n \cdot 10 \in \Theta(n) \implies \text{we have a worst-case linear time selection!}$$

What if ... we used groups of size 3 instead?

① What is the recurrence relation?

$$T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + \Theta(n)$$

② What is the asymptotics of that RR? Use MT on the following:

$$T(n) \leq 2T\left(\frac{2n}{3}\right) + \Theta(n)$$

$$T(n) \geq 2T\left(\frac{n}{3}\right) + \Theta(n)$$