Lec 2

Thursday, August 21, 2025 4:44 AM

Lecture 2: Divide & Conquer

Recap:
- * Course Policies
- * Big O
- * Merge Sort
- * Tree Method for solving a recursion

Announcements:
- * Exam 1 on Sep 18 (instead of Sep 16)
- * HW 1 is out (due Aug 28) ← Piazza
- * Recordings on Canvas
  Handwritten on Piazza

Plan:
- * Proof by induction
- * More D & C: (a) Max-diff
  (b) Median of Median

How to Solve a Recursion?

Idea: Guess a solution and verify by induction

E.g.: $T(n) = 2\,T(n/2) + c \cdot n$ and $T(1) = 1$

$\underbrace{\text{Guess}}_{\text{Experience / Tree Method heuristic}}$ that the soln is $T(n) = O(n \log n)$

Theorem: $T(n) \leq k\, n \log_2 n + 1$ where k is any constant $\geq 2c$

Pf by induc:

Step 1: Induction Statement/hypothesis: $T(i) \leq k\, i \log i + 1$

Step 2: Base Case: True because $T(1) \leq 1$

Step 3: Induc Step: Assuming I.H is true for all $i \leq N$,
we want to prove I.H. for $i = N+1$

$$T(N+1) \overset{\text{Recurrence}}{=} 2\, T\left(\frac{N+1}{2}\right) + c\,(N+1)$$

$$\overset{\text{I.H.}}{\leq} \cancel{2} \cdot k \frac{(N+1)}{\cancel{2}} \log \frac{N+1}{2} + 2 + c\,(N+1)$$

$$= k(N+1)\log(N+1) - k(N+1)\log 2 + c(N+1) + 2$$

$$\leq k(N+1)\log(N+1) + 1 \quad \text{for constant } k \geq 2c$$

$\Rightarrow$ Induc step is true ∎

Exercise: Prove by induction that $T(n) \geq \frac{c}{2} n \log n$.

Remark on Master's Theorem: $T(n) = a\,T(n/b) + n^d$ with $T(1) = 1$

then $$T(n) = \begin{cases} O(n^d) & \text{if } d > \log_b a \\ O(n^d \log n) & \text{if } d = \log_b a \\ O(n^{\log_b a}) & \text{if } d < \log_b a \end{cases}$$

Max - Difference Problem

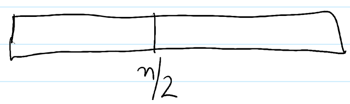
Given n numbers $A[0], A[1], \ldots, A[n-1]$.

Find $i < j$ to maximize $A[i] - A[j]$

Eg.

2	5	1	7	3	2	4	6

Ans $= 7 - 2 = 5$



Approach 1: D & C where we break in 2 halves and recursively compute max difference.

To merge, we return the best of $\{$left-soln, right soln, cross soln$\}$

cross soln: $i < n/2$ & $j \geq \frac{n}{2}$

$$T(n) = 2\,T(n/2) + \underbrace{O(n)}_{\text{Merge}}$$

$$\Rightarrow T(n) = \Theta(n \log n)$$

Approach 2: D & C where we break in 2 halves and recursively compute max difference and also store the max & min elements.

I.e., we return 3 things

Merge can be done in $O(1)$ time because we can compute

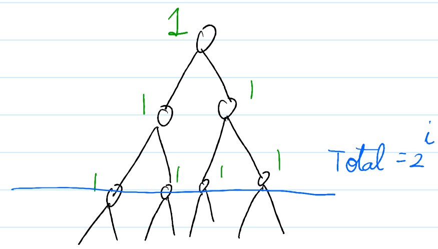
(1) Max - diff in $O(1)$

Merge can be done in $O(1)$ time, as we can compute

(1) Max − diff in $O(1)$

(2) Max − overall in $O(1)$

(3) Min − overall in $O(1)$



$$T(n) = 2T(n/2) + O(1)$$

$$\text{Sum} = \sum_{i=0}^{\log n} 2^i = O(n).$$

Exercise: Prove $T(n) = \Theta(n)$ by induction

Median Finding

Given n numbers $A[0, \ldots n-1]$. ← Say distinct for simplicity

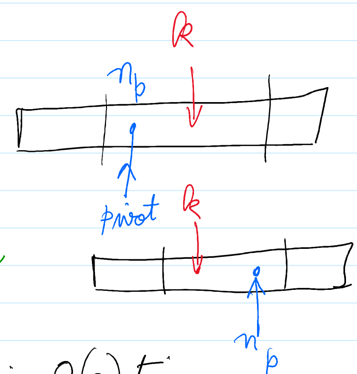
Given $k \in \{1, 2, \ldots, n\}$

Goal: Find the k-th largest number ← $(k-1)$ smaller #s & $n-k$ large #s

Sorting takes $\Theta(n \log n)$ time. Can we do it faster?

Thm: We can achieve this in $O(n)$ time.

Blum, Pratt, Rivest, Tarjan - 1972



Idea: (1) Find a $\underbrace{\text{'good pivot'}}_{\geq \frac{3n}{10} \text{ elements larger } \& \geq \frac{3n}{10} \text{ elems smaller}}$ recursively. ← $T\left(\frac{n}{5}\right)$

(2) Argue that we can drop one of sides of the pivot in $O(n)$ time

Pf: Calculate $\underbrace{\text{how many elems}}_{n_p}$ are less than pivot p

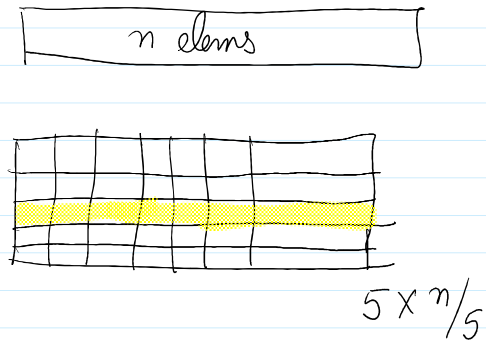
Case 1: $n_p > k$
Drop all $e > p$ & find k-th larg in rem

Case 2: $n_p \leq k$
Drop all $e < p$ & find $(k - n_p)$ th largest in remain

(3) Get Recursion: $T(n) \leq \underbrace{T\left(\frac{7n}{10}\right)}_{n - 3n \text{ remaining}} + \underbrace{T\left(\frac{n}{5}\right)}_{\uparrow} + \underbrace{O(n)}_{\text{Merge operation}}$

$\underbrace{\qquad}_{n - \frac{3n}{10} \text{ remaining elems}}$ $\underbrace{\qquad}_{\uparrow \text{ Finding pivot}}$ $\underbrace{\qquad}_{\text{Merge operation}}$

Exercise: Prove this recursion gives $T(n) = O(n)$.



$O(n)$ Step 1: Construct $5 * \frac{n}{5}$ array

$O(n)$ Step 2: Sort each colm of this array

$T(\frac{n}{5})$ Step 3: Recursively find the median of the column – medians.

Claim: This returned median of medians is a good pivot.