02 D&C + Complexity

Thursday, January 9, 2025 13:59

- Aproach for finding D&C algo (Q3)
- Complexity for recurrences (Q1 + Q2)

---

1) Think about the base case

2) Split problem in 2.
   If you have the solution for the 2 parts, what can we do with it?
   Is one of the 2 the final answer or is there some information missing?

3) Is there a way to make "merge" faster?
   Can any additional information be obtained from recursion?

Example: Max-Difference

Input: $A[1 \dots n]$

Find $i \leq j$ st $A[i] - A[j]$ is as large as possible.

$A = [3\ \underline{1}\ 2\ \underline{\underline{8}}\ 6\ \underline{4}\ 5]$

$8 - 4 = 4$

$8 - 1 = 7$ invalid because 1 came before 8

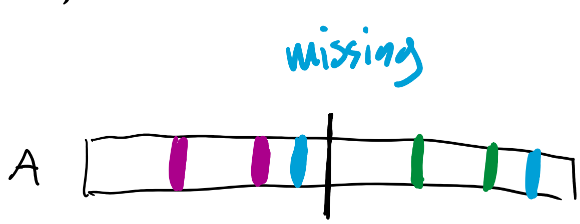
Naive

for $i = 1 \dots n$
  for $j = i \dots n$
    ...

$O(n^2)$

MaxDiff$(A[1 \dots n])$
  if $n == 1$
    return 0 // $A[i] - A[i]$

  $\underline{\text{left}}$ = MaxDiff$(A[1 \dots \frac{n}{2}])$
  $\underline{\text{right}}$ = MaxDiff$(A[\frac{n}{2}+1 \dots n])$
  $\text{maxL} = \max_{i \leq \frac{n}{2}} A[i]$ // $O(n)$
  ... $A[i]$ // $O(n)$



$\max_{i \leq j \leq \frac{n}{2}} A[i] - A[j]$

$\max_{\frac{n}{2} < i \leq j} A[i] - A[j]$

$\max_{\frac{n}{2}<i\le j}$

$$\text{maxL} = \max_{i \le \frac{n}{2}} A[i] \quad // \ O(n)$$

$$\text{minR} = \min_{\frac{n}{2} < j} A[j] \quad // \ O(n)$$

missing $\max_{i \le \frac{n}{2} < j} A[i] - A[j]$

$$= \left(\max_{i \le \frac{n}{2}} A[i]\right) - \left(\min_{\frac{n}{2} < j} A[j]\right)$$

crossing = maxL - minR

return max(left, right, crossing)

$$T(n) = 2 \cdot T\left(\tfrac{n}{2}\right) + O(n) \qquad a=2,\ b=2$$

$$= O(n \log n) \qquad c = 1 \quad // \ O(n) = O(n^1)$$

Alternative:

MaxDiff (A[1...n])

  if n == 1

    return 0, A[1], A[1] // max difference, smallest entry, largest entry

  left, minL, maxL = MaxDiff (A[1 ... n/2])

  right, minR, maxR = MaxDiff (A[n/2+1 ... n])

  crossing = maxL - minR

  return max(left, right, crossing), min(minL, minR), max(maxL, maxR)

$$T(n) = 2 \cdot T\left(\tfrac{n}{2}\right) + O(1) \qquad a = 2,\ b = 2$$

$$= O(n) \qquad c = 0 \quad // \ O(1) = O(n^0)$$

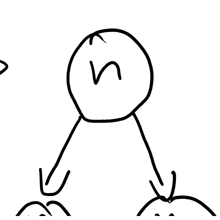
Q3 HW D&C only

2:40

General Approach

Recursion Trees

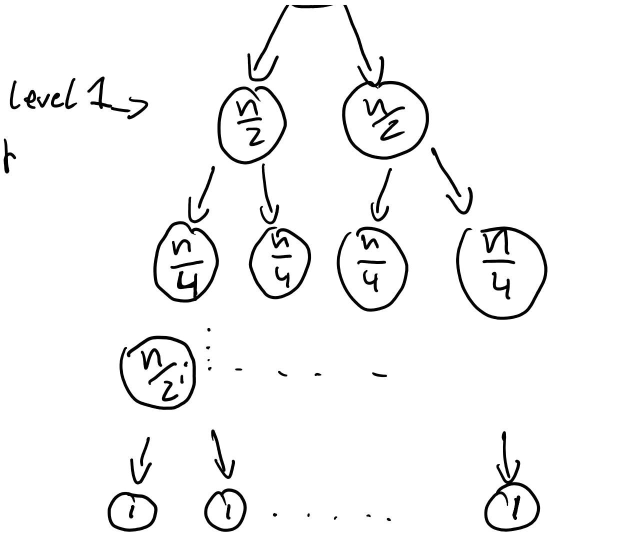
level 0 ->



level 1 ->

Recursion Trees

- each node is one function call
- each node gets a label = size of input
- add edges to represent the call



$$T(n) = 2 \cdot T\left(\tfrac{n}{2}\right) + O(n^2)$$

Each horizontal line is a "Level".

height of this tree is $\log_2(n)$

i-th level has 2^i many nodes and a node in i-th level has $\underline{n/2^i}$ as input

Total time $\displaystyle T(n) = \sum_{i=0}^{\log_2(n)} 2^i \cdot O\left(\left(\frac{n}{2^i}\right)^2\right)$

$i=0$: this sum goes over all levels

2^i: # nodes in ith level

$\left(\frac{n}{2^i}\right)^2$: time spent on merge by 1 node in ith level

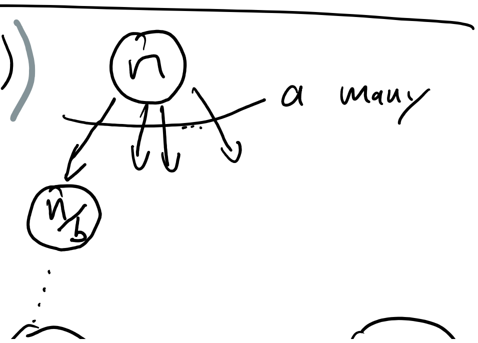
$$= \sum_{i=0}^{\log_2(n)} O\left(n^2/2^i\right) = O(n^2) \cdot \sum_{i=0}^{\log_2(n)} \frac{1}{2^i}$$

$$\le O(n^2) \cdot \sum_{i=0}^{\infty} 1/2^i = O(n^2) \cdot 2$$
$$= O(n^2)$$

$$T(n) = 2 \cdot T\left(\tfrac{n}{2}\right) \cdot O(n)$$
$$= \sum_{i=0}^{\log_2(n)} \cancel{2^i} \cdot O\left(n/\cancel{2^i}\right) = \sum_{i=0}^{\log_2 n} O(n) = O(n \log n)$$

$$T(n) = a \cdot T\left(\tfrac{n}{b}\right) + n^c \quad (T(d) = O(d))$$

$$T(1) = 1$$



$$= \sum_{i=0}^{\log_b(n)} a^i \cdot O\left(\left(\frac{n}{b^i}\right)^c\right)$$

$$O(n^c) \sum^{\log_b(n)} \left(\frac{a}{\ldots}\right)^i$$

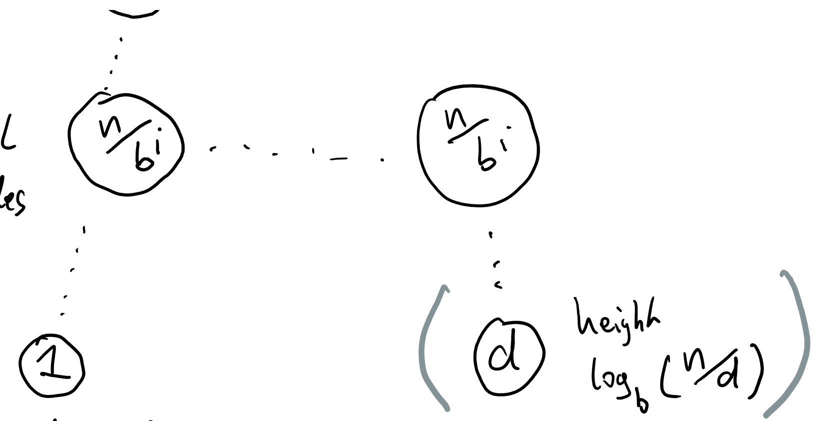
$i=0$

$$= O(n^c) \cdot \sum_{i=0}^{\log_b(n)} \left(\frac{a}{b^c}\right)^i$$

$$\left(\log_b(n) = \frac{\log_2(n)}{\log_2(b)}\right)$$

i-th level has a^i nodes

height of tree is $\log_b(n)$



$$O(n^c) \cdot \sum_{i=0}^{\log_b n} \left(\frac{a}{b^c}\right)^i = \begin{cases} O(n^c \cdot \log n) & \text{if } a = b^c \\ O(n^c) & \text{if } a < b^c \\ \underbrace{\qquad\qquad}_{} & \text{if } a > b^c \end{cases}$$

$$O\left(n^c \cdot \left(\frac{a}{b^c}\right)^{\log_b n}\right) = O\left(\cancel{n^c} \cdot \underbrace{\frac{1}{b^{\log_b(n) \cdot c}}}_{\cancel{1/n^c}} \cdot a^{\log_b(n)}\right)$$

$$= O\left(a^{\log_b(n)}\right)$$

$$= O\left(2^{\log(a) \cdot \log_b(n)}\right)$$

$$= O\left(2^{\log_2(a) \cdot \frac{\log_2(n)}{\log_2(b)}}\right)$$

$$= O\left(n^{\frac{\log_2(a)}{\log_2(b)}}\right) = O\left(n^{\log_b(a)}\right)$$

Lemma: $$\sum_{i=0}^{t} q^i = \begin{cases} O(t) & \text{if } q = 1 \\ O(q^t) & \text{if } q > 1 \\ O(1) & \text{if } q < 1 \end{cases}$$

if $q>1$ $$\sum_{i=0}^{t} q^i = q^t \cdot \sum_{i=0}^{t} q^{i-t} = q^t \sum_{i=0}^{t} q^{-i}$$

if $q>1$

$$\sum_{i=0}^{t} q^i = q^t \cdot \sum_{t=0}^{t} q^{i-t} = q^t \sum_{i=0}^{t} q^{-i}$$

$$= q^t \sum_{i=0}^{t} \left(\frac{1}{q}\right)^i$$

$$\leq q^t \underbrace{\sum_{i=0}^{\infty} \left(\frac{1}{q}\right)^i}_{O(1)} = O(q^t)$$