CS 381 – FALL 2019

Week 3.3, Friday, Sept 6

Announcement: Homework 1 Solutions released on Piazza
Homework 2 Due: September 16th, 2019 @ 11:59PM on Gradescope

Recap: Master Theorem

- Derived by analyzing recursion tree

$$T(n) = \begin{cases} \mathbf{1000000} & \boldsymbol{if\ n \leq 100} \\ a \times T\left(\frac{n}{b} + \mathbf{1}\right) + n^c & otherwise \end{cases}$$

Obtain Geometric Series: $T(n) = \Theta\left(n^c \sum_{i=1}^{\log_b n} \left(\frac{a}{b^c}\right)^i\right)$

- **Key Ratio: a/b^c** $\boldsymbol{log_b\, a \geq c \leftrightarrow \frac{a}{b^c} \geq 1}$
- Case 1: $\left(\frac{a}{b^c}\right) < 1$ $T(n) = \Theta(n^c)$
- Case 2: $\left(\frac{a}{b^c}\right) = 1$ $T(n) = \Theta(n^c \log n)$
- Case 3: $\left(\frac{a}{b^c}\right) > 1$ $T(n) = \Theta\left(n^{\log_b a}\right)$

Other Form of Master Theorem

- What if MergeCost is not exactly f(n)=n^c?
  - f(n) = n log n? or
  - $\mathrm{f(n)} = n^{\frac{1}{10}+\log_b a}/\log(n)$?

$$T(n) = \begin{cases} \boldsymbol{O(1)} & \boldsymbol{if\ n \leq 100} \\ a \times T\left(\frac{n}{b} + \mathbf{50}\right) + f(n) & otherwise \end{cases}$$

Assume $f(n) \geq 0$

Case 1: $f(n) = \Omega\left(n^{\varepsilon+\log_b a}\right)$ $\quad T(n) = \Theta(f(n))$

Case 2: $f(n) = \Theta\left(n^{\log_b a} \log^k n\right)$ $\quad T(n) = \Theta\left(n^{\log_b a} \log^{k+1} n\right)$
(assumes $k \geq 0$)

Case 3: $f(n) = O\left(n^{\log_b a-\varepsilon}\right)$ $\quad T(n) = \Theta\left(n^{\log_b a}\right)$

Clicker Question

$$T(n) \le \begin{cases} \mathbf{1000000} & \boldsymbol{if\ n \le 100} \\ a \times T\left(\frac{n}{b} + \mathbf{50}\right) + f(n) & otherwise \end{cases}$$

Assume $f(n) \ge 0$

Case 1: $f(n) = \Omega\left(n^{\varepsilon + \log_b a}\right)$ $\quad T(n) = \Theta(f(n))$

Case 2: $f(n) = \Theta\left(n^{\log_b a} \log^k n\right)$ $\quad T(n) = \Theta\left(n^{\log_b a} \log^{k+1} n\right)$
(assumes $k \ge 0$)

Case 3: $f(n) = O\left(n^{\log_b a - \varepsilon}\right)$ $\quad T(n) = \Theta\left(n^{\log_b a}\right)$

Suppose that $\mathrm{f(n)} = n^{\frac{1}{10} + \log_b a} / \log(n)$ above what is $T(n)$?

A. $\Theta(f(n))$

B. $\Theta\left(n^{\log_b a} \log^{k+1} n\right)$

C. $T(n) = \Theta\left(n^{\log_b a}\right)$

D. More info required

Clicker Question

$$T(n) \leq \begin{cases} \mathbf{1000000} & \boldsymbol{if\ n \leq 100} \\ a \times T\left(\frac{n}{b} + \mathbf{50}\right) + f(n) & otherwise \end{cases}$$

Assume $f(n) \geq 0$

Case 1: $f(n) = \Omega\left(n^{\varepsilon + \log_b a}\right)$ $\quad T(n) = \Theta(f(n))$

Case 2: $f(n) = \Theta\left(n^{\log_b a} \log^k n\right)$ $\quad T(n) = \Theta\left(n^{\log_b a} \log^{k+1} n\right)$
(assumes $k \geq 0$)

Case 3: $f(n) = O\left(n^{\log_b a - \varepsilon}\right)$ $\quad T(n) = \Theta\left(n^{\log_b a}\right)$

Suppose that $\mathrm{f(n)} = n^{\frac{1}{10} + \log_b a} / \log(n)$ above what is $T(n)$?

A. $\Theta(f(n))$

B. $\Theta\left(n^{\log_b a} \log^{k+1} n\right)$

C. $T(n) = \Theta\left(n^{\log_b a}\right)$

D. More info required

Other Types of Recurrences

- $T(n) = T(n-1) + 1$ **(Unroll: $T(n) = \Theta(n)$)**
  - $T(n) = T(n-1) + 1 = T(n-2) + 1 + 1$
    $= T(n-3) + 1 + 1 + 1 = \cdots = T(n-k) + k$
    $= T(1) + n - 1$

- $T(n) = 2 \times T(n-10)$ **(Exponential)**

Two branches

Only constant reduction in input size

$$T(n) = 2T(n-10) = 2\big(2T(n-20)\big) = 4T(n-20)$$

$$= 8T(n-30) = \cdots = 2^{i}T(n-10i)$$

$$= 2^{\frac{n}{10}-1}T(10) = \Theta\left(2^{\frac{n}{10}}\right)$$

Other Recurrences

- $\mathrm{T(n) = T(n-1) + T(}n - 3)$ (Exponential)

Two branches

Only constant reduction in input size

$$\mathrm{T(n)} = \Theta(\mathrm{c^n})$$

- Unrolling gets messy fast! How to find c? [Trick]

  Assume $T(n) = c^n$ for some c

$$c^n = T(n) = \mathrm{T(n-1) + T(}n-3) = c^{n-1} + c^{n-3}$$

$$\rightarrow c^n = c^{n-1} + c^{n-3} \rightarrow \mathrm{c}^3 = c^2 + 1$$

$$\rightarrow \mathrm{c} \approx 1.46577$$

(Root of Characteristic Equation)

Must verify solution by induction

Other Recurrences

- $\mathrm{T(n)} = \mathrm{T(n-1)} + \mathrm{T}(n-3)$ **(Exponential)**

Two branches   Only constant reduction in input size

$$\rightarrow c^n = c^{n-1} + c^{n-3} \rightarrow \mathrm{c}^3 = c^2 + 1$$

$$\rightarrow \mathrm{c} \approx 1.46577$$

(Root of Characteristic Equation)

Claim: $\mathrm{T(n)} \leq kc^n$ (pick k s.t. T(0) < k)

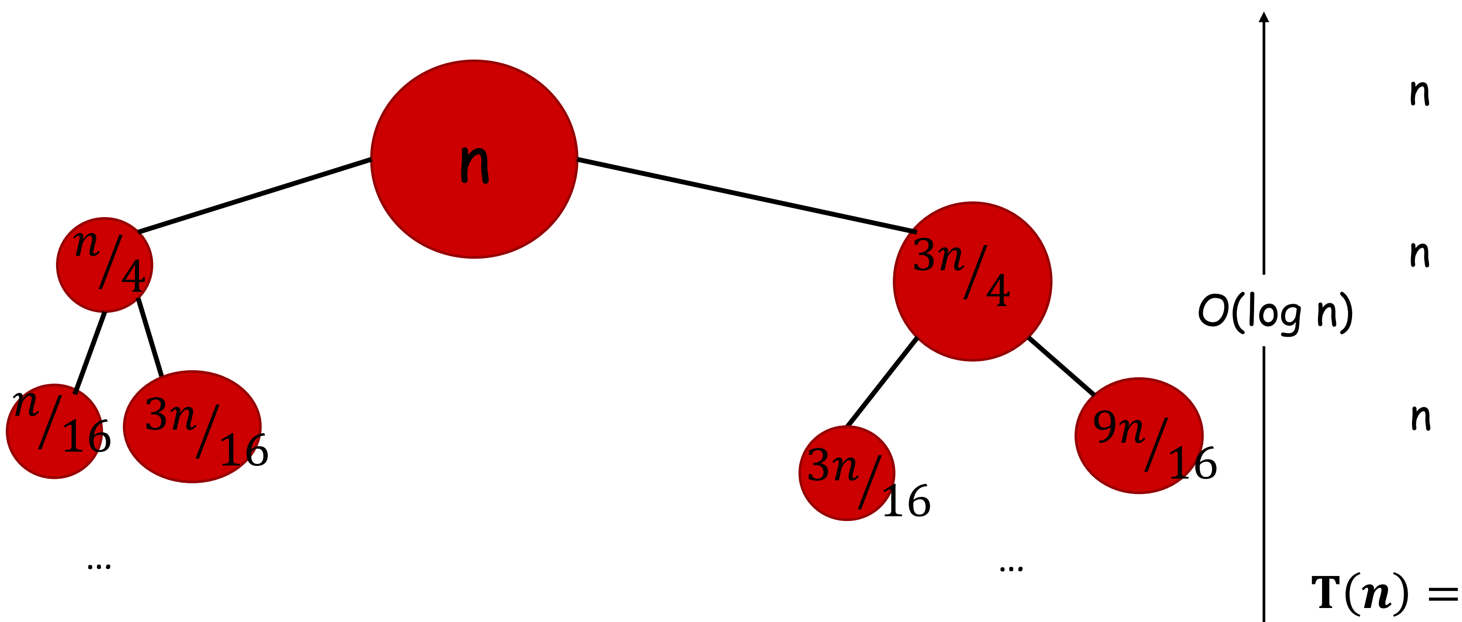
Inductive Step:

$$\begin{aligned} \mathrm{T(n)} &= \mathrm{T(n-1)} + \mathrm{T}(n-3) \\ &\leq k(c^{n-1} + c^{n-3}) && \text{(IH)} \\ &= k(c^n) && \text{(Choice of c)} \end{aligned}$$

- MergeSort with Uneven Split: Split L into A, B of sizes n/4 and 3n/4.

$$\mathrm{T(n)} = \mathrm{T}\left(\frac{\mathrm{n}}{4}\right) + T\left(\frac{3n}{4}\right) + n$$

A bit harder to Analyze with recursion tree



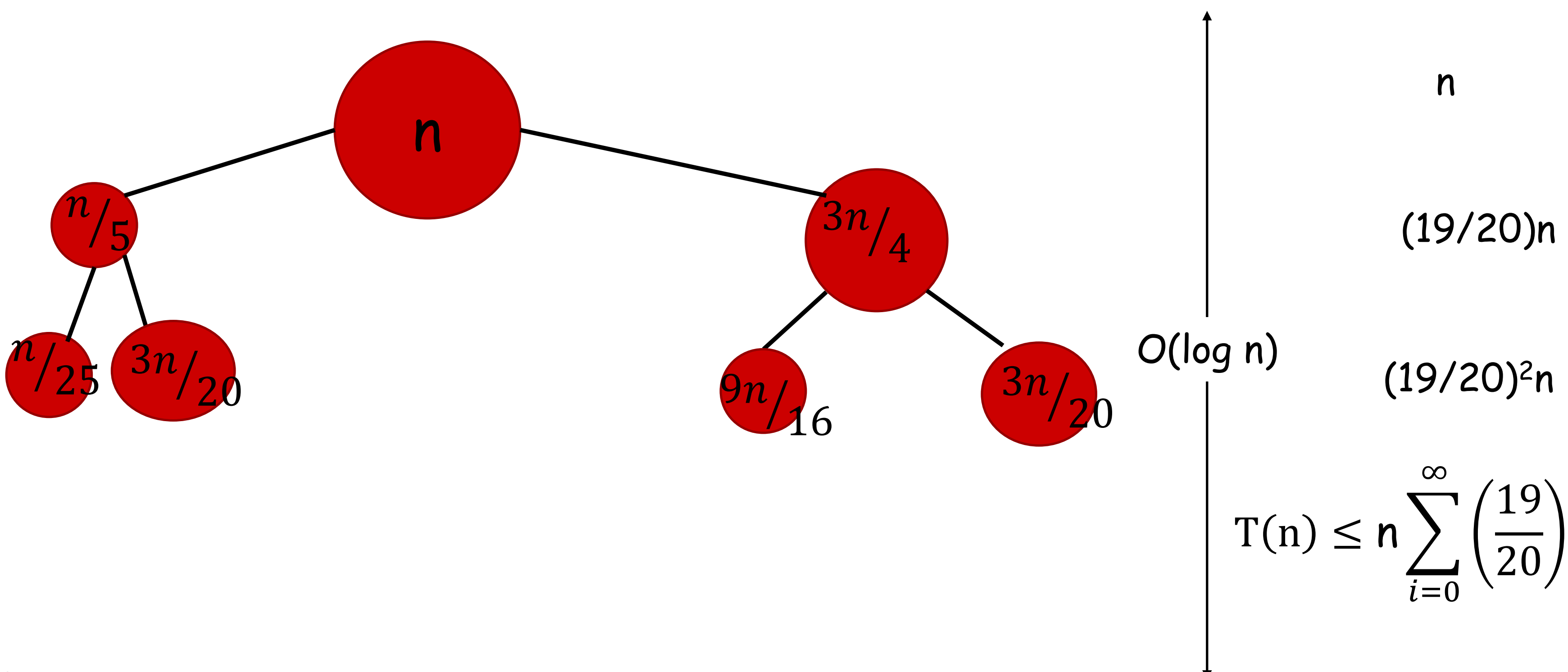
$$\mathbf{T}(\boldsymbol{n}) = \boldsymbol{\Theta}(\boldsymbol{n}\log\boldsymbol{n})$$

Another Unbalanced Recurrence

- **Geometric Series**

$$T(n) = T\left(\frac{n}{5}\right) + T\left(\frac{3n}{4}\right) + n$$

- **Claim:** $T(n) = \Theta(n)$



Divide and conquer algorithms

- Mergesort
- Quicksort
- Binary Search
- *Linear-time selection*
- *Skyline Problem*
- *Maximum Subarray*
- *Counting inversions*

Maximum Subarray Problem

Given an array A of n (positive and negative) numbers, find the contiguous subarray whose sum has the largest value.

10 5 -20 $\boxed{5\ 12\ -6\ 33\ 6\ 2\ -52\ \ 6\ \ 45\ \ 3}$ -4

Brute Force

- For every pair i and j, i≤j, compute the sum from A(i) to A(j). Remember the pair resulting in the maximum.
- How many pairs? $O(n^2)$
- Using previously computed values, the total running time is $O(n^2)$ (n^3 is excessive brute force)

Aim for O(n) or O(n log n)?

Divide and conquer?

- Split the problem into two halves and solve each recursively
- Combine two solutions to produce the final answer

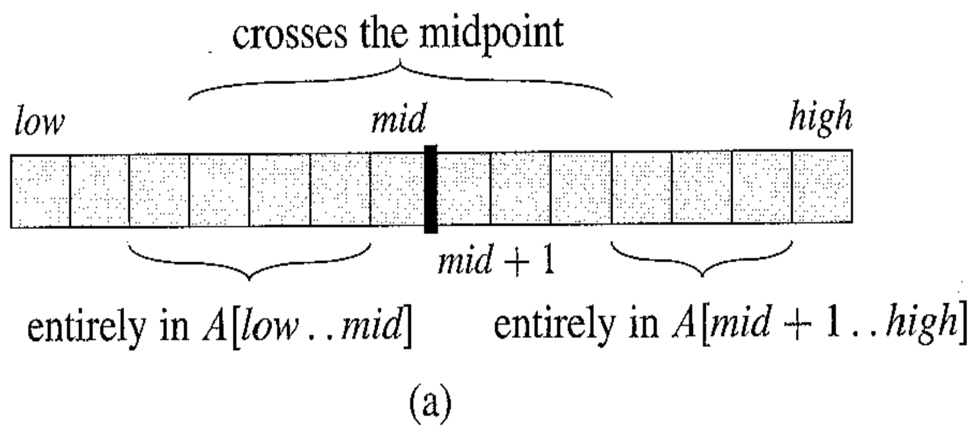


Figure 4.4 **(a)** Possible locations of subarrays of $A[low..high]$: entirely in $A[low..mid]$, entirely in $A[mid+1..high]$, or crossing the midpoint *mid*. **(b)** Any subarray of $A[low..high]$ crossing the midpoint comprises two subarrays $A[i..mid]$ and $A[mid+1..j]$, where $low \le i \le mid$ and $mid < j \le high$.

10 5 -20 21 12 -18 12 5 15 -22 6 5 3 -4 -12 55

10 5 -20 21 12 -18 12 5 15 -22 6 5 3 -4 -12 55

10 5 -20 21 12 -18 12 5 15 -22 6 5 3 -4 -12 55

32 **46**

Combining the answers from the two subproblems

- The maximum subarray can be in one the two halves (easy case) or it can cross the midpoint
- To determine the maximum subarray crossing the midpoint, we can compute the two maximum subarrays "anchored" at the midpoint.
  - Running Sum: sweep left (resp. right) from midpoint
    - keeping track of max
    - Requires time $O(n)$ for merge step

Results in $T(n) = 2T(n/2) + cn$

Gives an **O(n log n)** time algorithm

A Faster O(n) time solution

Idea: Each recursive call returns ***additional information*** to make merge step easier

- ***Optimal Solution:* (i,j) *and value*** $\boldsymbol{v} = \sum_{x=i}^{j} A[x]$
- ***Total Sum:*** $T = \sum_{x=1}^{n} A[x]$
- $\boldsymbol{i_{end}}$ ***maximizing*** *value* $\boldsymbol{v_{end}} = \sum_{x=i_{end}}^{n} A[x]$
- $\boldsymbol{j_{begin}}$ ***maximizing value*** $\boldsymbol{v_{begin}} = \sum_{x=1}^{j_{begin}} A[x]$

i $\quad$ j , $\boldsymbol{j_{begin}}$ $\quad$ $\boldsymbol{i_{end}}$

10 5 -20 21 12 -18 12 5 -50 2 5

$\boldsymbol{v_{begin} = 27}$ $\quad$ $\boldsymbol{v = 32}$ $\quad$ $\boldsymbol{v_{end} = 5}$

A Faster O(n) time solution

Merge in Constant Time: Suppose A was split into L and R

- Three possibilities for (i,j): (i^L, j^L), (i^R, j^R), (i^L_{end}, j^L_{begin})
  - Case 1: Opt in L
  - Case 2: Opt in R
  - Case 3: Opt crosses L and R

$$\underbrace{(i^L, j^L)}_{v^L}, \underbrace{(i^R, j^R)}_{v^R}, \underbrace{(i^L_{end}, j^L_{begin})}_{v^L_{end} + v^R_{begin}}$$

10 5 -20 21 12 -18 12 5 -50 2 5 | -5 50 1 -2 -80 44 1 1 6

$v^L_{begin} = 27$ $v^L = 32$ $v^L_{end} = 7$ $v^R_{begin} = 46$ $v^R = 52$ $v^R_{end} = 52$

$$v = \max\{v^L, v^R, \boxed{v^L_{end} + v^R_{begin}}\} = v^L_{end} + v^R_{begin} = 53$$

$$\rightarrow (i,j) = (i^L_{end}, j^L_{begin})$$

A Faster O(n) time solution

Merge in Constant Time: Suppose A was split into L and R

- Still Needs to Compute Extra Values

  For any merges higher in the recursion tree!

  - **Update Total:** $T = T^L + T^R$
  - $\boldsymbol{i_{end}}$ ***maximizing value*** $\boldsymbol{v_{end}} = \sum_{x=i_{end}}^{n} A[x]$
    - Case 1: $\boldsymbol{i_{end}} = i_{end}^R$ (interval in R)
    - Case 2: $\boldsymbol{i_{end}} = i_{end}^L$ (interval crosses L)

10 5 -20 21 12 -18 12 5 -50 $\boxed{2\ 5}$ -5 50 1 -2 -80 $\boxed{44\ 1\ 1\ 6}$

$\mathbf{v}_{\mathbf{end}}^{\mathbf{L}} = \mathbf{7}$ $\mathbf{v}_{\mathbf{end}}^{\mathbf{R}} = \mathbf{52}$

$\mathbf{v}_{\boldsymbol{end}} = \max\{\mathbf{v}_{\boldsymbol{end}}^{\mathbf{R}}, \mathbf{T}^{\boldsymbol{R}} + \mathbf{v}_{\mathbf{end}}^{\mathbf{L}}\} = \mathbf{v}_{\boldsymbol{end}}^{\mathbf{R}} = \mathbf{52} \rightarrow \boldsymbol{i_{end}} = i_{end}^R$

A Faster O(n) time solution

Merge in Constant Time: Suppose A was split into L and R

- Merge Needs to Compute Extra Values
  - $T = T^L + T^R$
  - $\boldsymbol{j_{begin}}$ ***maximizing value*** $\boldsymbol{v_{begin}} = \sum_{x=1}^{j_{begin}} A[x]$
    - Similar to computing $\boldsymbol{i_{end}}$
    - Case 1: $\boldsymbol{j_{begin}} = j_{begin}^L$ (interval in L)
    - Case 2: $\boldsymbol{j_{begin}} = j_{begin}^R$ (interval crosses R)

$\boldsymbol{v_{begin}^L = 27}$

$\mathbf{v_{begin}^R = 46}$

10 5 -20 21 12 -18 12 5 -50 2 5 | -5 50 1 -2 -80 44 1 1 6

$\mathbf{v_{begin}} = \max\{\mathbf{v}_{\boldsymbol{begin}}^{\mathbf{L}}, \mathbf{T}^{\boldsymbol{L}} + \mathbf{v}_{\mathbf{begin}}^{\boldsymbol{R}}\} = \mathbf{T}^{\boldsymbol{L}} + \mathbf{v}_{\mathbf{begin}}^{\boldsymbol{R}} = \mathbf{-16 + 46 = 30}$

$\rightarrow \boldsymbol{j_{begin}} = j_{begin}^R$

Summary: A Faster O(n) time solution

Idea: Each recursive call returns ***additional information*** to make merge step easier

Constant Time Merge:

$$T(n) = 2T\left(\frac{n}{2}\right) + 1$$

Master Theorem (a=2, b=2, c=0): $T(n) = \Theta(n)$

Summary: Maximum Subarray problem

Given an array A of n (positive and negative) numbers, find the contiguous subarray whose sum has the largest value.

10 5 -20 $\boxed{5\ 12\ {-6}\ 33\ 6\ 2\ {-52}\ 6\ 45\ 3}$ -4

- *All pairs: $O(n^2)$*
- *D&C with simple combine step: O(n log n)*
- *D&C with extra information: O(n)*
- Simple non-D&C solution: O(n)