CS 341: Algorithms

Lecture 2: Solving recurrences

Éric Schost

based on lecture notes by many other CS341 instructors

David R. Cheriton School of Computer Science, University of Waterloo

Fall 2024

From exact to sloppy recurrences

Overview

Consider a recursive algorithm Algo.

Assumption: for an input size $n > 1$, Algo does

- $\boldsymbol{a}$ recursive calls, in size either $\lfloor \boldsymbol{n/b} \rfloor$ or $\lceil \boldsymbol{n/b} \rceil$ ($\boldsymbol{a > 0}$ and $\boldsymbol{b > 1}$, constants)
- between $\boldsymbol{c'n^y}$ and $\boldsymbol{cn^y}$ extra operations. ($\boldsymbol{c}$ and $\boldsymbol{c'}$ nonzero constants, $\boldsymbol{y}$ constant)

Claim

Solving the sloppy recurrence $\boldsymbol{T(n) = aT(n/b) + cn^y}$ for powers of b gives a valid Θ-bound for best and worst-case runtimes.

Overview

Consider a recursive algorithm Algo.

Assumption: for an input size $n > 1$, Algo does

- $\boldsymbol{a}$ recursive calls, in size either $\lfloor \boldsymbol{n/b} \rfloor$ or $\lceil \boldsymbol{n/b} \rceil$ ($\boldsymbol{a > 0}$ and $\boldsymbol{b > 1}$, constants)
- between $\boldsymbol{c'n^y}$ and $\boldsymbol{cn^y}$ extra operations. ($\boldsymbol{c}$ and $\boldsymbol{c'}$ nonzero constants, $\boldsymbol{y}$ constant)

Claim

Solving the sloppy recurrence $\boldsymbol{T(n) = aT(n/b) + cn^y}$ for powers of b gives a valid Θ-bound for best and worst-case runtimes.

Remark 1: if we only know that we do **at most** $\boldsymbol{cn^y}$ extra operations, we only get a big-O.

Remark 2: to be concrete, we'll do the proof for mergesort.

- one recursive call with $\lfloor n/2 \rfloor$, the other with $\lceil n/2 \rceil$, and roughly n extra operations.
- so $a = b = 2$ and $y = 1$

Best and worst-case recurrence relations

Let $T^w(n), T^b(n)$ be the **worst case**, resp. **best case** in size n.

Worst-case recurrence: $T^w(1) = d$ and

$$T^w(n) \leq T^w\left(\left\lceil \frac{n}{2} \right\rceil\right) + T^w\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + cn \quad \text{if } n > 1$$

Best-case recurrence: $T^b(1) = d'$ and

$$T^b(n) \geq T^b\left(\left\lceil \frac{n}{2} \right\rceil\right) + T^b\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + c'n \quad \text{if } n > 1$$

Best and worst-case recurrence relations

Let $T^w(n), T^b(n)$ be the **worst case**, resp. **best case** in size n.

Worst-case recurrence: $T^w(1) = d$ and

$$T^w(n) \leq T^w\left(\left\lceil \frac{n}{2} \right\rceil\right) + T^w\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + cn \quad \text{if } n > 1$$

Best-case recurrence: $T^b(1) = d'$ and

$$T^b(n) \geq T^b\left(\left\lceil \frac{n}{2} \right\rceil\right) + T^b\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + c'n \quad \text{if } n > 1$$

Remark: could be possible to write $=$ instead or $\leq$ or $\geq$, but harder to prove

Worst-case analysis

Use an equal sign: define T by

$$T(1) = d, \qquad T(n) = T\left(\left\lceil \frac{n}{2} \right\rceil\right) + T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + cn \quad \text{if } n > 1$$

Exercise

$T^w(n) \leq T(n)$ and $T(n)$ increasing (easy induction)

Remark: same thing can be done for $T^b(n)$.

Worst-case analysis (cont.)

Sloppy recurrence:

$$t(1) = d, \qquad t(n) = 2t\left(\frac{n}{2}\right) + cn \quad \text{if } n > 1$$

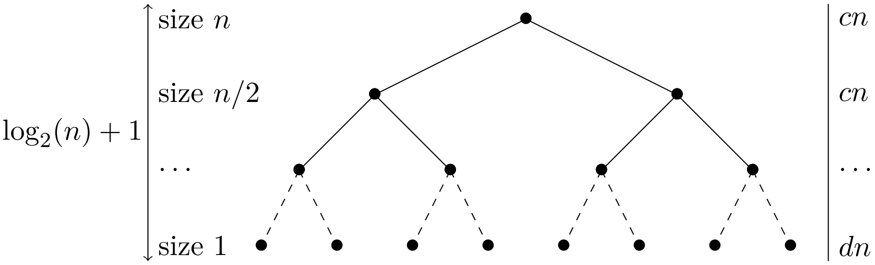
Observations

- this only defines $t(n)$ for powers of 2.
- $T(2^k) = t(2^k)$ for any k
- T is increasing so $T(n) \leq T(\text{next power of 2}) = t(\text{next power of 2})$

Conclusion:

- enough to analyze $t(n)$, n a power of 2
- we'll do it using the recursion tree

The mergesort recursion tree

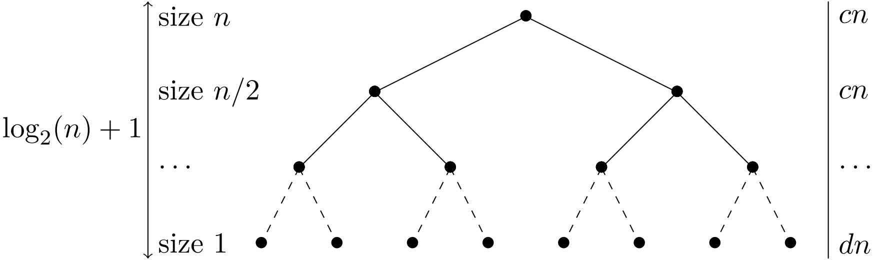


Total: $t(n) = cn\log_2(n) + dn$ for n a power of 2.

Consequences

- $\boldsymbol{T(n) \in O(n\log(n))}$
- $\boldsymbol{T^w(n) \in O(n\log(n))}$

The mergesort recursion tree



Total: $t(n) = cn\log_2(n) + dn$ for n a power of 2.

Consequences

- $\boldsymbol{T(n) \in O(n\log(n))}$
- $\boldsymbol{T^w(n) \in O(n\log(n))}$

Remark: same approach proves $\boldsymbol{T^b(n) \in \Omega(n\log(n))}$, and so

$$\boldsymbol{T^b(n), T^w(n) \in \Theta(n\log(n))}$$

The master theorem

The master theorem

Solves many recurrence relations coming from divide-and-conquer algorithms.

Suppose that $\boldsymbol{a \geq 1}$ and $\boldsymbol{b > 1}$. Consider the recurrence

$$T(n) = a\,T\left(\frac{n}{b}\right) + cn^y \quad n > 1$$

Let

$$\boldsymbol{x = \log_b a} \quad (\text{so } a = b^x).$$

Then

$$T(n) \in \begin{cases} \Theta(\boldsymbol{n^y}) & \text{if } \boldsymbol{y > x} \quad (\text{root heavy}) \\ \Theta(\boldsymbol{n^y \log n}) & \text{if } \boldsymbol{y = x} \quad (\text{balanced}) \\ \Theta(\boldsymbol{n^x}) & \text{if } \boldsymbol{y < x} \quad (\text{leaf heavy}) \end{cases}$$

We do the proof for n a power of b; result true for $n \in \mathbb{R}_{\geq 0}$.

Recursion tree

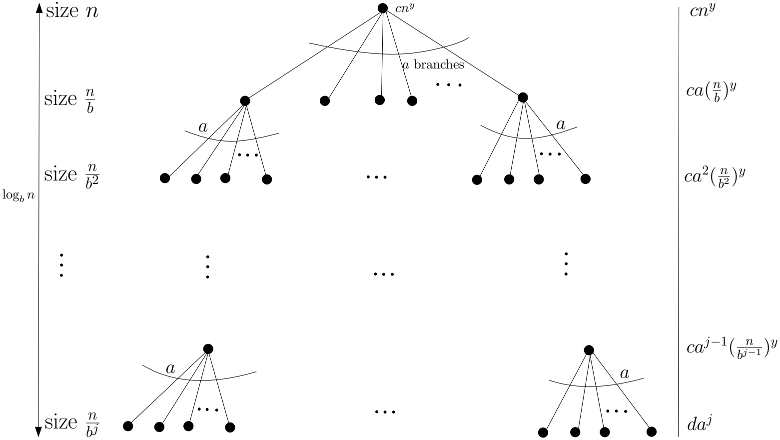
Suppose that $n = b^j$, $a \geq 1$, $b \geq 2$ are integers and

$$T(n) = a\,T\left(\frac{n}{b}\right) + c\,n^y, \qquad T(1) = d.$$

Recursion tree

Suppose that $n = b^j$, $a \geq 1$, $b \geq 2$ are integers and

$$T(n) = a\,T\left(\frac{n}{b}\right) + c\,n^y, \qquad T(1) = d.$$



Breakdown of the cost

Suppose that $a \geq 1$ and $b \geq 2$ are integers and

$$T(n) = a\,T\left(\frac{n}{b}\right) + c\,n^y, \qquad T(1) = d.$$

Let $n = b^j$.

size of subproblem	# nodes	cost/node	total cost
$n = b^j$	1	$c\,n^y$	$c\,n^y$
$n/b = b^{j-1}$	a	$c\,(n/b)^y$	$c\,a\,(n/b)^y$
$n/b^2 = b^{j-2}$	a^2	$c\,(n/b^2)^y$	$c\,a^2\,(n/b^2)^y$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$n/b^{j-1} = b$	a^{j-1}	$c\,(n/b^{j-1})^y$	$c\,a^{j-1}\,(n/b^{j-1})^y$
$n/b^j = 1$	a^j	d	$d\,a^j$

Computing $T(n)$

Total:

$$T(n) = d\,a^j + c\,n^y \sum_{i=0}^{j-1} \left(\frac{a}{b^y}\right)^i = dn^x + cn^y \sum_{i=0}^{j-1} \left(\frac{a}{b^y}\right)^i .$$

Proof: $a = b^x$ and $n = b^j$, so $a^j = (b^x)^j = (b^j)^x = n^x$.

Computing $T(n)$

Total:

$$T(n) = d\,a^j + c\,n^y \sum_{i=0}^{j-1} \left(\frac{a}{b^y}\right)^i = dn^x + cn^y \sum_{i=0}^{j-1} \left(\frac{a}{b^y}\right)^i .$$

Proof: $a = b^x$ and $n = b^j$, so $a^j = (b^x)^j = (b^j)^x = n^x$.

Observation: geometric sum with ratio $\boldsymbol{r = \frac{a}{b^y} = b^{x-y}}$:

- if $\boldsymbol{r < 1} \iff \boldsymbol{x < y}$: $\sum r^i \in \Theta(1)$, so $T(n) \in \Theta(n^y)$
- if $\boldsymbol{r = 1} \iff \boldsymbol{x = y}$: $\sum r^i \in \Theta(\log n)$, so $T(n) \in \Theta(n^y \log n)$
- if $\boldsymbol{r > 1} \iff \boldsymbol{x > y}$: $\sum r^i \in \Theta(r^j)$, so $T(n) \in \Theta(n^x)$

Proof (last item):

$$r^j = \frac{a^j}{b^{yj}} = \frac{n^x}{n^y}$$

Examples

$T(n) = 4T(n/2) + n$ multiplying polynomials

- $a = 4$, $b = 2$, $y = 1$ so $x = \log_b a = 2$ and $\boldsymbol{T(n) = \Theta(n^2)}$

Examples

$\boldsymbol{T(n) = 4T(n/2) + n}$ multiplying polynomials

- $a = 4$, $b = 2$, $y = 1$ so $x = \log_b a = 2$ and $\boldsymbol{T(n) = \Theta(n^2)}$

$\boldsymbol{T(n) = 2T(n/2) + n^2}$

- $a = 2$, $b = 2$, $y = 2$ so $x = \log_b a = 1$ and $\boldsymbol{T(n) = \Theta(n^2)}$

Examples

$T(n) = 4T(n/2) + n$ multiplying polynomials

- $a = 4$, $b = 2$, $y = 1$ so $x = \log_b a = 2$ and $\boldsymbol{T(n) = \Theta(n^2)}$

$T(n) = 2T(n/2) + n^2$

- $a = 2$, $b = 2$, $y = 2$ so $x = \log_b a = 1$ and $\boldsymbol{T(n) = \Theta(n^2)}$

$T(n) = 2T(n/4) + 1$ kd-trees

- $a = 2$, $b = 4$, $y = 0$ so $x = \log_b a = 1/2$ and $\boldsymbol{T(n) = \Theta(\sqrt{n})}$

Examples

$\boldsymbol{T(n) = T(n/2) + 1}$ binary search

- $a = 1$, $b = 2$, $y = 0$ so $x = \log_b a = 0$ and $\boldsymbol{T(n) = \Theta(\log n)}$

Examples

$\boldsymbol{T(n) = T(n/2) + 1}$ binary search

- $a = 1$, $b = 2$, $y = 0$ so $x = \log_b a = 0$ and $\boldsymbol{T(n) = \Theta(\log n)}$

$\boldsymbol{T(n) = T(n/2) + n}$ amortized analysis of dynamic arrays

- $a = 1$, $b = 2$, $y = 1$ so $x = \log_b a = 0$ and $\boldsymbol{T(n) = \Theta(n)}$

Examples

$T(n) = T(n/2) + 1$ binary search

- $a = 1$, $b = 2$, $y = 0$ so $x = \log_b a = 0$ and $\boldsymbol{T(n) = \Theta(\log n)}$

$T(n) = T(n/2) + n$ amortized analysis of dynamic arrays

- $a = 1$, $b = 2$, $y = 1$ so $x = \log_b a = 0$ and $\boldsymbol{T(n) = \Theta(n)}$

$T(n) = T(n/2)$

- does not fit in our framework, but obvious

Examples

$T(n) = T(n/2) + 1$ binary search

- $a = 1$, $b = 2$, $y = 0$ so $x = \log_b a = 0$ and $\boldsymbol{T(n) = \Theta(\log n)}$

$T(n) = T(n/2) + n$ amortized analysis of dynamic arrays

- $a = 1$, $b = 2$, $y = 1$ so $x = \log_b a = 0$ and $\boldsymbol{T(n) = \Theta(n)}$

$T(n) = T(n/2)$

- does not fit in our framework, but obvious

$T(n) = 2T(n/2) + n \log(n)$

- does not fit in our framework, have to redo the recursion tree analysis

Alternative: guess and prove

Consider $T(n) = 2T(n/2) + n$, $T(1) = 0$, n power of 2.

Alternative: guess and prove

Consider $T(n) = 2T(n/2) + n$, $T(1) = 0$, n power of 2.

Guess: $T(n) \leq n$. Proof by induction? Assume $T(n/2) \leq n/2$.

$$T(n) = 2T(n/2) + n \leq 2(n/2) + n = \boldsymbol{2n} \not\leq n$$

Alternative: guess and prove

Consider $T(n) = 2T(n/2) + n$, $T(1) = 0$, n power of 2.

Guess: $T(n) \le n$. Proof by induction? Assume $T(n/2) \le n/2$.

$$T(n) = 2T(n/2) + n \le 2(n/2) + n = \boldsymbol{2n} \nleq n$$

Guess: $T(n) \le kn$, k TBD? Assume $T(n/2) \le kn/2$.

$$T(n) = 2T(n/2) + n \le 2(kn/2) + n = \boldsymbol{kn + n} \nleq kn$$

Alternative: guess and prove

Consider $T(n) = 2T(n/2) + n$, $T(1) = 0$, n power of 2.

Guess: $T(n) \leq n$. Proof by induction? Assume $T(n/2) \leq n/2$.

$$T(n) = 2T(n/2) + n \leq 2(n/2) + n = \boldsymbol{2n} \not\leq n$$

Guess: $T(n) \leq kn$, k TBD? Assume $T(n/2) \leq kn/2$.

$$T(n) = 2T(n/2) + n \leq 2(kn/2) + n = \boldsymbol{kn + n} \not\leq kn$$

Guess: $T(n) \leq kn \log_2 n$, k TBD? Assume $T(n/2) \leq kn/2 \log_2(n/2)$.

$$T(n) = 2T(n/2) + n \leq 2(kn/2 \log_2(n/2)) + n = \boldsymbol{kn \log_2 n - kn + n}$$

proof by induction OK if $k \geq 1$.

Alternative: guess and prove

Consider $T(n) = 2T(n/2) + n$, $T(1) = 0$, n power of 2.

Guess: $T(n) \leq n$. Proof by induction? Assume $T(n/2) \leq n/2$.

$$T(n) = 2T(n/2) + n \leq 2(n/2) + n = \boldsymbol{2n} \nleq n$$

Guess: $T(n) \leq kn$, k TBD? Assume $T(n/2) \leq kn/2$.

$$T(n) = 2T(n/2) + n \leq 2(kn/2) + n = \boldsymbol{kn + n} \nleq kn$$

Guess: $T(n) \leq kn \log_2 n$, k TBD? Assume $T(n/2) \leq kn/2 \log_2(n/2)$.

$$T(n) = 2T(n/2) + n \leq 2(kn/2 \log_2(n/2)) + n = \boldsymbol{kn \log_2 n - kn + n}$$

proof by induction OK if $k \geq 1$.

Remark: usually harder to prove $\boldsymbol{T(n) = \cdots}$