Lecture 2 Solving Recurrences

Examples from previous courses - Sorting

MergeSort(A):

- Divide array into left and right halves.
- 2 Subproblems: Left and right half subarrays, recurse on both.
- Merge the two sorted arrays $\Rightarrow O(n)$ work to conquer.

Analysis:

Best-case/Worst-case: $T(n) = 2T(n/2) + O(n) \in O(n \log n)$

Rigorously, $T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + cn$

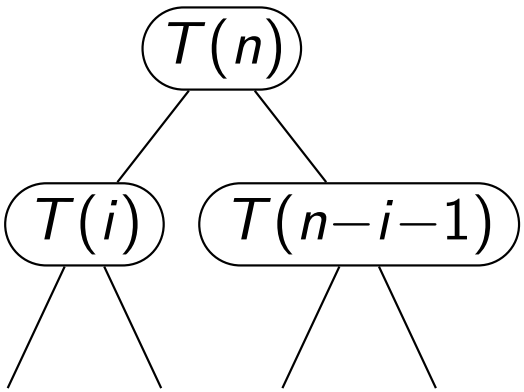
Solving Recurrence Relations

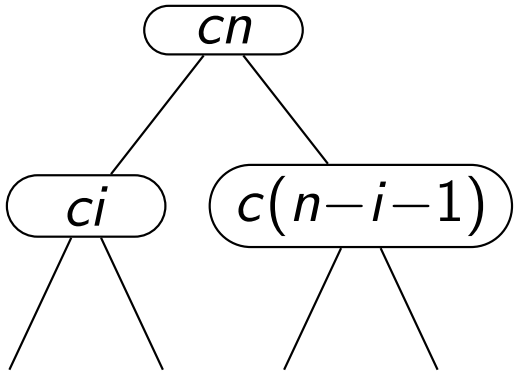
CS 240 typically only expected you to simply expand a few steps and find the pattern but introduced the ideas of a recursion tree and the substitution method.

Solving Recurrence Relations

CS 240 typically only expected you to simply expand a few steps and find the pattern but introduced the ideas of a recursion tree and the substitution method.

Quicksort: randomly chooses a pivot, partition to find the location i of the pivot and create two subproblems. The recursion tree traces the recursion and shows the work done for each subproblem.

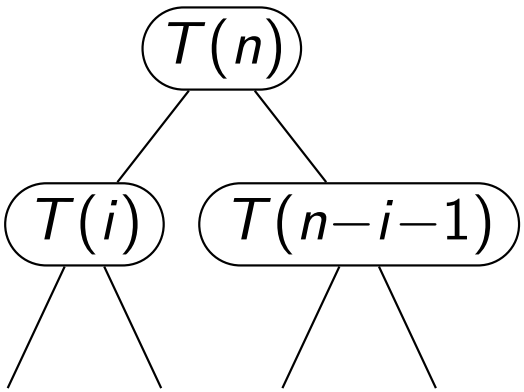


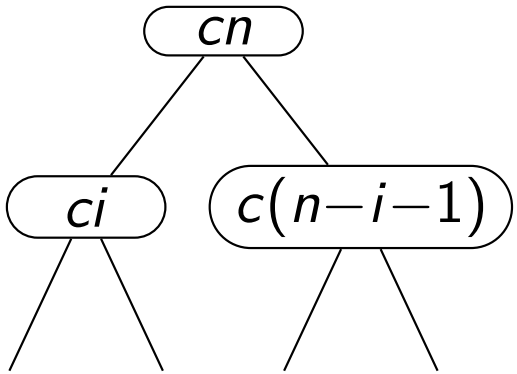


Solving Recurrence Relations

CS 240 typically only expected you to simply expand a few steps and find the pattern but introduced the ideas of a recursion tree and the substitution method.

Quicksort: randomly chooses a pivot, partition to find the location i of the pivot and create two subproblems. The recursion tree traces the recursion and shows the work done for each subproblem.





Worst-Case (worst luck): $T(n) = T(n-1) + cn \in \Theta(n^2)$
Best-Case (best luck): $T(n) = 2T(n/2) + cn \in \Theta(n \log n)$
Expected Runtime: $\Theta(n \log n)$

Recursion Tree for MergeSort

$T(n) = 2T(n/2) + cn$, if n is even
$T(1) = 0$, if counting number of comparisons

Recursion Tree where n is a power of 2:

Total work in the recursion tree sums to: $c \cdot n \log n$

Solving Mergesort

If we want to be precise, the math is not trivial:

$$T(n) = \begin{cases} T(\lceil \frac{n}{2} \rceil) + T(\lfloor \frac{n}{2} \rfloor) + (n-1) & \text{if } n > 1 \\ 0 & \text{if } n = 1 \end{cases}$$

Solving Mergesort

If we want to be precise, the math is not trivial:

$$T(n) = \begin{cases} T(\lceil \frac{n}{2} \rceil) + T(\lfloor \frac{n}{2} \rfloor) + (n-1) & \text{if } n > 1 \\ 0 & \text{if } n = 1 \end{cases}$$

Resolves to $T(n) = n\lceil \log n \rceil - 2^{\lceil \log n \rceil} + 1$

The recursion tree method finds the exact solution of the recurrence when n is a power of 2, $T(n) \in O(n \log n)$.

Solving Mergesort

If we want to be precise, the math is not trivial:

$$T(n) = \begin{cases} T(\lceil \frac{n}{2} \rceil) + T(\lfloor \frac{n}{2} \rfloor) + (n-1) & \text{if } n > 1 \\ 0 & \text{if } n = 1 \end{cases}$$

Resolves to $T(n) = n\lceil \log n \rceil - 2^{\lceil \log n \rceil} + 1$

The recursion tree method finds the exact solution of the recurrence when n is a power of 2, $T(n) \in O(n \log n)$.

When given n that is not a power of 2, we can also argue that performing the analysis using $n' = s$mallest power of 2 larger than n will also give a precise result since $n' < 2 \cdot n$, runtimes are typically increasing and we only want the growth rate.

Solving Recurrences by Substitution

Also known as "Lucky Guess", "Guess and Check", "Guess and Prove", ...

$$T(n) = \begin{cases} T(\lceil \frac{n}{2} \rceil) + T(\lfloor \frac{n}{2} \rfloor) + (n-1) & \text{if } n > 1 \\ 0 & \text{if } n = 1 \end{cases}$$

Guess and prove by *Induction* that $T(n) \leq c \cdot n \log n, \forall n \geq 1$.

Solving Recurrences by Substitution

Also known as "Lucky Guess", "Guess and Check", "Guess and Prove", ...

$$T(n) = \begin{cases} T(\lceil \frac{n}{2} \rceil) + T(\lfloor \frac{n}{2} \rfloor) + (n-1) & \text{if } n > 1 \\ 0 & \text{if } n = 1 \end{cases}$$

Guess and prove by *Induction* that $T(n) \leq c \cdot n \log n, \forall n \geq 1$.

Base case: For $n = 1$, $T(1) = 0$ and $c \cdot n \log n = 0$; i.e. $0 \leq 0$.

Solving Recurrences by Substitution

Also known as "Lucky Guess", "Guess and Check", "Guess and Prove", ...

$$T(n) = \begin{cases} T(\lceil \frac{n}{2} \rceil) + T(\lfloor \frac{n}{2} \rfloor) + (n-1) & \text{if } n > 1 \\ 0 & \text{if } n = 1 \end{cases}$$

Guess and prove by *Induction* that $T(n) \leq c \cdot n \log n, \forall n \geq 1$.

Base case: For $n = 1$, $T(1) = 0$ and $c \cdot n \log n = 0$; i.e. $0 \leq 0$.

Induction Hypothesis: Assume that $T(k) \leq c \cdot k \log k$ for all $k < n$ where $k \geq 2$.

Induction Step: One method to give a rigorous proof is to separate into even and odd cases. If n is even, we don't need floors and ceilings. If n is odd, ...

Careful! Watchout!

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

Claim: $T(n) \in O(n)$

Prove $T(n) \leq cn, \forall n \geq n_0$.

Induction Hypothesis: Assume $T(k) \leq c \cdot k, \forall k < n, k \geq n_0$.

$$\begin{aligned} T(n) &= 2T(n/2) + n \\ &\leq 2c(n/2) + n \text{ by the Induction Hypothesis} \\ &= (c+1)n \end{aligned}$$

Careful! Watchout!

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

Claim: $T(n) \in O(n)$

Prove $T(n) \leq cn, \forall n \geq n_0$.

Induction Hypothesis: Assume $T(k) \leq c \cdot k, \forall k < n, k \geq n_0$.

$$\begin{aligned} T(n) &= 2T(n/2) + n \\ &\leq 2c(n/2) + n \text{ by the Induction Hypothesis} \\ &= (c+1)n \end{aligned}$$

Conclusion: $T(n) \in O(n)$.

Careful! Watchout!

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

Claim: $T(n) \in O(n)$

Prove $T(n) \leq cn, \forall n \geq n_0$.

Induction Hypothesis: Assume $T(k) \leq c \cdot k, \forall k < n, k \geq n_0$.

$$\begin{aligned} T(n) &= 2T(n/2) + n \\ &\leq 2c(n/2) + n \text{ by the Induction Hypothesis} \\ &= (c+1)n \end{aligned}$$

Conclusion: $T(n) \in O(n)$.

The conclusion is clearly incorrect!

The problem is the constant is continuously growing.

Substitution - Changing the Guess

$$T(n) = \begin{cases} T(\lceil \frac{n}{2} \rceil) + T(\lfloor \frac{n}{2} \rfloor) + 1 & \text{if } n > 1 \\ 1 & \text{if } n = 1 \end{cases}$$

Guess $T(n) \in O(n)$.
Prove by induction that $T(n) \leq cn$ for some constant c.

Substitution - Changing the Guess

$$T(n) = \begin{cases} T(\lceil \frac{n}{2} \rceil) + T(\lfloor \frac{n}{2} \rfloor) + 1 & \text{if } n > 1 \\ 1 & \text{if } n = 1 \end{cases}$$

Guess $T(n) \in O(n)$.
Prove by induction that $T(n) \leq cn$ for some constant c.

Induction Step:

$$\begin{aligned} T(n) &= T(\lceil \frac{n}{2} \rceil) + T(\lfloor \frac{n}{2} \rfloor) + 1 \\ &\leq c \cdot \lceil \frac{n}{2} \rceil + c \cdot \lfloor \frac{n}{2} \rfloor + 1 \\ &= cn + 1 \end{aligned}$$

Substitution - Changing the Guess

$$T(n) = \begin{cases} T(\lceil \frac{n}{2} \rceil) + T(\lfloor \frac{n}{2} \rfloor) + 1 & \text{if } n > 1 \\ 1 & \text{if } n = 1 \end{cases}$$

Guess $T(n) \in O(n)$.
Prove by induction that $T(n) \leq cn$ for some constant c.

Induction Step:

$$\begin{aligned} T(n) &= T(\lceil \frac{n}{2} \rceil) + T(\lfloor \frac{n}{2} \rfloor) + 1 \\ &\leq c \cdot \lceil \frac{n}{2} \rceil + c \cdot \lfloor \frac{n}{2} \rfloor + 1 \\ &= cn + 1 \end{aligned}$$

What went wrong?

Substitution - Changing the Guess

$$T(n) = \begin{cases} T(\lceil \frac{n}{2} \rceil) + T(\lfloor \frac{n}{2} \rfloor) + 1 & \text{if } n > 1 \\ 1 & \text{if } n = 1 \end{cases}$$

Guess $T(n) \in O(n)$.
Prove by induction that $T(n) \leq cn$ for some constant c.

Induction Step:

$$\begin{aligned} T(n) &= T(\lceil \frac{n}{2} \rceil) + T(\lfloor \frac{n}{2} \rfloor) + 1 \\ &\leq c \cdot \lceil \frac{n}{2} \rceil + c \cdot \lfloor \frac{n}{2} \rfloor + 1 \\ &= cn + 1 \end{aligned}$$

What went wrong?
Fix? Add a lower order term: Try $T(n) \leq cn - 1$.

Substitution - Changing the Guess

$$T(n) = \begin{cases} T(\lceil \frac{n}{2} \rceil) + T(\lfloor \frac{n}{2} \rfloor) + 1 & \text{if } n > 1 \\ 1 & \text{if } n = 1 \end{cases}$$

Guess $T(n) \in O(n)$.
Prove by induction that $T(n) \leq cn$ for some constant c.

Induction Step:

$$\begin{aligned} T(n) &= T(\lceil \frac{n}{2} \rceil) + T(\lfloor \frac{n}{2} \rfloor) + 1 \\ &\leq c \cdot \lceil \frac{n}{2} \rceil + c \cdot \lfloor \frac{n}{2} \rfloor + 1 \\ &= cn + 1 \end{aligned}$$

What went wrong?
Fix? Add a lower order term: Try $T(n) \leq cn - 1$.
$T(n) \leq c \cdot \lceil \frac{n}{2} \rceil - 1 + c \cdot \lfloor \frac{n}{2} \rfloor - 1 + 1 = cn - 1$

Substituion - Changing Variables

$$T(n) = 2T(\lfloor \sqrt{n} \rfloor) + \log n$$

Let $m = \log n$ so $n = 2^m$.
Our new recurrence relation is:

$$T(2^m) = 2T(2^{m/2}) + m$$

Substituion - Changing Variables

$$T(n) = 2T(\lfloor \sqrt{n} \rfloor) + \log n$$

Let $m = \log n$ so $n = 2^m$.
Our new recurrence relation is:

$$T(2^m) = 2T(2^{m/2}) + m$$

Let $S(m) = T(2^m)$ so $S(m/2) = T(2^{m/2})$.

Then $S(m) = 2S(m/2) + m$ which is a recurrence we know.

Substituion - Changing Variables

$$T(n) = 2T(\lfloor \sqrt{n} \rfloor) + \log n$$

Let $m = \log n$ so $n = 2^m$.
Our new recurrence relation is:

$$T(2^m) = 2T(2^{m/2}) + m$$

Let $S(m) = T(2^m)$ so $S(m/2) = T(2^{m/2})$.

Then $S(m) = 2S(m/2) + m$ which is a recurrence we know.

$S(m) \in O(m \log m)$ so $T(2^m) \in O(m \log m)$ and

$T(n) \in O((\log n)(\log \log n))$

Common Recurrences

We often see recurrences of the form:

$$T(n) = aT(\frac{n}{b}) + cn^k$$

Example: Mergesort $k = 1, a = 2, b = 2$

$$T(n) = 2T(\frac{n}{2}) + cn \in O(n \log n)$$

Example: $k = 1, a = 1, b = 2$

$$T(n) = T(\frac{n}{2}) + cn \in O(n)$$

Example: $k = 1, a = 4, b = 2$

$$T(n) = 4T(\frac{n}{2}) + cn \in O(n^2)$$

Master Theorem

Theorem: Given

$$T(n) = aT(\frac{n}{b}) + cn^k$$

where $a \geq 1, b > 1, c > 0, k \geq 0$, then

$$T(n) \in \begin{cases} \Theta(n^k) & \text{if } a < b^k \text{ i.e. } \log_b a < k \\ \Theta(n^k \log n) & \text{if } a = b^k \\ \Theta(n^{\log_b a}) & \text{if } a > b^k \end{cases}$$

Proof: For a rigorous proof, use induction.
Less rigorous, think about the recursion tree.