CS 561, Lecture 3 - Recurrences

Jared Saia
University of New Mexico

Recurrence Relations

"Oh how should I not lust after eternity and after the nuptial ring of rings, the ring of recurrence" - Friedrich Nietzsche, Thus Spoke Zarathustra

- Getting the run times of recursive algorithms can be challenging
- Consider an algorithm for binary search (next slide)
- Let $T(n)$ be the run time of this algorithm on an array of size n
- Then we can write $T(1) = 1$, $T(n) = T(n/2) + 1$

Alg: Binary Search

```
bool BinarySearch (int arr[], int s, int e, int key){
  if (e-s<=0) return false;
  int mid = (e+s)/2;
  if (key==arr[mid]){
    return true;
  }else if (key < arr[mid]){
    return BinarySearch (arr,s,mid,key);}
  else{
    return BinarySearch (arr,mid,e,key)}
}
```

Recurrence Relations

- $T(n) = T(n/2) + 1$ is an example of a *recurrence* relation
- A *Recurrence Relation* is any equation for a function T, where T appears on both the left and right sides of the equation.
- We always want to "solve" these recurrence relation by getting an equation for T, where T appears on just the left side of the equation

Recurrence Relations

- Whenever we analyze the run time of a recursive algorithm, we will first get a recurrence relation
- To get the actual run time, we need to solve the recurrence relation

Substitution Method

- One way to solve recurrences is the substitution method aka "guess and check"
- What we do is make a good guess for the solution to $T(n)$, and then try to prove this is the solution by induction

Example

- Let's guess that the solution to $T(n) = T(n/2) + 1$, $T(1) = 1$ is $T(n) = O(\log n)$
- In other words, $T(n) \leq c \log n$ for all $n \geq n_0$, for some positive constants c, n_0
- We can prove that $T(n) \leq c \log n$ is true by plugging back into the recurrence

Proof

We prove this by induction:

- B.C.: $T(2) = 2 \leq c \log 2$ provided that $c \geq 2$
- I.H.: For all $j < n$, $T(j) \leq c \log(j)$
- I.S.:

$$\begin{aligned} T(n) &= T(n/2) + 1 && (1) \\ &\leq (c \log(n/2)) + 1 && (2) \\ &= c(\log n - \log 2) + 1 && (3) \\ &= c \log n - c + 1 && (4) \\ &\leq c \log n && (5) \end{aligned}$$

Second step holds by IH. Last step holds for all $n > 0$ if $c \geq 1$. Thus, entire proof holds if $n \geq 2$ and $c \geq 2$.

Recurrences and Induction

Recurrences and Induction are closely related:

- To *find* a solution to $f(n)$, solve a recurrence
- To *prove* that a solution for $f(n)$ is correct, use induction

For both recurrences and induction, we always solve a big problem by reducing it to smaller problems!

Some Examples

- The next several problems can be attacked by induction/recurrences
- For each problem, we'll need to reduce it to smaller problems
- Question: How can we reduce each problem to a smaller subproblem?

Sum Problem

- $f(n)$ is the sum of the integers $1, \ldots, n$

Tree Problem

- $f(n)$ is the maximum number of leaf nodes in a binary tree of height n

Recall:

- In a binary tree, each node has at most two children
- A *leaf* node is a node with no children
- The height of a tree is the length of the longest path from the root to a leaf node.

Binary Search Problem

- $f(n)$ is the maximum number of queries that need to be made for binary search on a sorted array of size n.

Dominoes Problem

- $f(n)$ is the number of ways to tile a 2 by n rectangle with dominoes (a domino is a 2 by 1 rectangle)

Simpler Subproblems

- Sum Problem: What is the sum of all numbers between 1 and $n-1$ (i.e. $f(n-1)$)?
- Tree Problem: What is the maximum number of leaf nodes in a binary tree of height $n-1$? (i.e. $f(n-1)$)
- Binary Search Problem: What is the maximum number of queries that need to be made for binary search on a sorted array of size $n/2$? (i.e. $f(n/2)$)
- Dominoes problem: What is the number of ways to tile a 2 by $n-1$ rectangle with dominoes? What is the number of ways to tile a 2 by $n-2$ rectangle with dominoes? (i.e. $f(n-1)$, $f(n-2)$)

Recurrences

- Sum Problem: $f(n) = f(n-1) + n$, $f(1) = 1$
- Tree Problem: $f(n) = 2 * f(n-1)$, $f(0) = 1$
- Binary Search Problem: $f(n) = f(n/2) + 1$, $f(2) = 1$
- Dominoes problem: $f(n) = f(n-1) + f(n-2)$, $f(1) = 1$, $f(2) = 2$

Guesses

- Sum Problem: $f(n) = (n+1)n/2$
- Tree Problem: $f(n) = 2^n$
- Binary Search Problem: $f(n) = \log n$
- Dominoes problem: $f(n) = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n$

Inductive Proofs

"Trying is the first step to failure" - Homer Simpson

- Now that we've made these guesses, we can try using induction to prove they're correct (the substitution method)
- We'll give inductive proofs that these guesses are correct for the first three problems

Sum Problem

- Want to show that $f(n) = (n+1)n/2$.
- Prove by induction on n
- Base case: $f(1) = 2 * 1/2 = 1$
- Inductive hypothesis: for all $j < n$, $f(j) = (j+1)j/2$
- Inductive step:

$$\begin{aligned} f(n) &= f(n-1) + n && (6) \\ &= n(n-1)/2 + n && (7) \\ &= (n+1)n/2 && (8) \end{aligned}$$

Where the second step holds by the IH.

Tree Problem

- Want to show that $f(n) = 2^n$.
- Prove by induction on n
- Base case: $f(0) = 2^0 = 1$
- Inductive hypothesis: for all $j < n$, $f(j) = 2^j$
- Inductive step:

$$
\begin{aligned}
f(n) &= 2 * f(n-1) && (9)\\
&= 2 * (2^{n-1}) && (10)\\
&= 2^n && (11)
\end{aligned}
$$

(second step holds by IH)

Binary Search Problem

- Want to show that $f(n) = \log n$. (assume n is a power of 2)
- Prove by induction on n
- Base case: $f(2) = \log 2 = 1$
- Inductive hypothesis: for all $j < n$, $f(j) = \log j$
- Inductive step:

$$\begin{aligned} f(n) &= f(n/2) + 1 && (12) \\ &= \log n/2 + 1 && (13) \\ &= \log n - \log 2 + 1 && (14) \\ &= \log n && (15) \end{aligned}$$

(second step holds by IH)

In Class Exercise

- Consider the recurrence $f(n) = 2f(n/2) + 1$, $f(1) = 1$
- Guess that $f(n) \leq cn - 1$:
- Q1: Show the base case - for what values of c does it hold?
- Q2: What is the inductive hypothesis?
- Q3: Show the inductive step.

Recurrences and Inequalities

- Often easier to prove that a recurrence is no more than some quantity than to prove that it equals something
- Consider: $f(n) = f(n-1) + f(n-2)$, $f(1) = f(2) = 1$
- "Guess" that $f(n) \leq 2^n$

Inequalities (II)

Goal: Prove by induction that for $f(n) = f(n-1) + f(n-2)$, $f(1) = f(2) = 1$, $f(n) \leq 2^n$

- Base case: $f(1) = 1 \leq 2^1$, $f(2) = 1 \leq 2^2$
- Inductive hypothesis: For all $j < n$, $f(j) \leq 2^j$
- Inductive step:

$$
\begin{aligned}
f(n) &= f(n-1) + f(n-2) && (16)\\
&\leq 2^{n-1} + 2^{n-2} && (17)\\
&< 2 * 2^{n-1} && (18)\\
&= 2^n && (19)
\end{aligned}
$$

(second step holds by IH)

Recursion-tree method

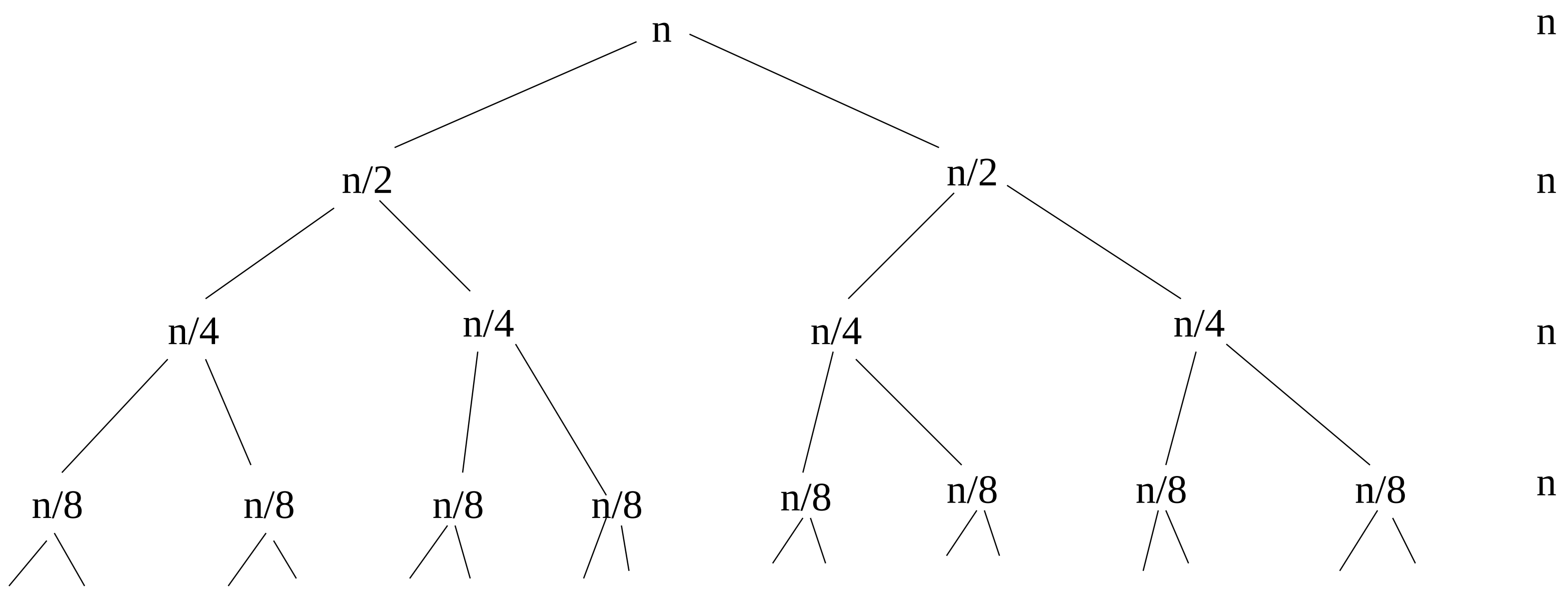
- Each node represents the cost of a single subproblem in a recursive call
- First, we sum the costs of the nodes in each level of the tree
- Then, we sum the costs of all of the levels

Recursion-tree method

- Can use to get a good guess which is then refined and verified using substitution method
- Best method (usually) for recurrences where a term like $T(n/c)$ appears on the right hand side of the equality

Example 1

- Consider the recurrence for the running time of Mergesort: $T(n) = 2T(n/2) + n$, $T(1) = O(1)$

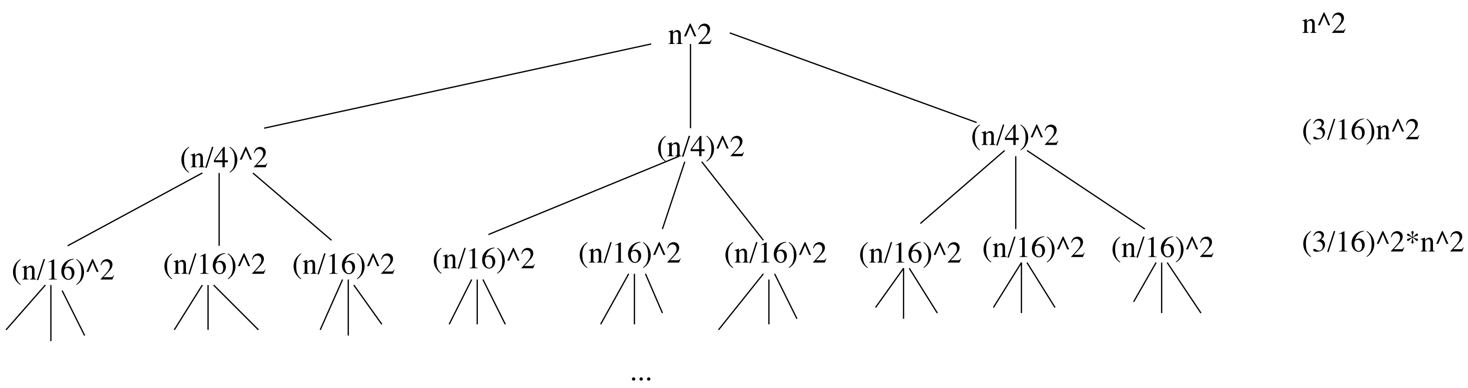


Example 1

- We can see that each level of the tree sums to n
- Further the depth of the tree is $\log n$ ($n/2^d = 1$ implies that $d = \log n$).
- Thus there are $\log n + 1$ levels each of which sums to n
- Hence $T(n) = \Theta(n \log n)$

Example 2

- Let's solve the recurrence $T(n) = 3T(n/4) + n^2$
- Note: For simplicity, from now on, we'll assume that $T(i) = \Theta(1)$ for all small constants i. This will save us from writing the base cases each time.



Example 2

- We can see that the i-th level of the tree sums to $(3/16)^i n^2$.
- Further the depth of the tree is $\log_4 n$ ($n/4^d = 1$ implies that $d = \log_4 n$)
- So we can see that $T(n) = \sum_{i=0}^{\log_4 n} (3/16)^i n^2$

Solution

$$T(n) = \sum_{i=0}^{\log_4 n} (3/16)^i n^2 \quad (20)$$

$$< n^2 \sum_{i=0}^{\infty} (3/16)^i \quad (21)$$

$$= \frac{1}{1-(3/16)} n^2 \quad (22)$$

$$= O(n^2) \quad (23)$$

Master Theorem

- Divide and conquer algorithms often give us running-time recurrences of the form

$$T(n) = a\,T(n/b) + f(n) \tag{24}$$

- Where a and b are constants and $f(n)$ is some other function.
- The so-called "Master Method" gives us a general method for solving such recurrences

Master Theorem

- Unfortunately, the Master Theorem doesn't work for all functions $f(n)$
- Further many useful recurrences don't look like $T(n)$
- However, the theorem allows for very fast solution of recurrences when it applies

Master Theorem

- Master Theorem is just a special case of the use of recursion trees
- Consider equation $T(n) = a\,T(n/b) + f(n)$
- We start by drawing a recursion tree

The Recursion Tree

- The root contains the value $f(n)$
- It has a children, each of which contains the value $f(n/b)$
- Each of these nodes has a children, containing the value $f(n/b^2)$
- In general, level i contains a^i nodes with values $f(n/b^i)$
- Hence the sum of the nodes at the i-th level is $a^i f(n/b^i)$

Details

- The tree stops when we get to the base case for the recurrence
- We'll assume $T(1) = f(1) = \Theta(1)$ is the base case
- Thus the depth of the tree is $\log_b n$ and there are $\log_b n + 1$ levels

Recursion Tree

- Let $T(n)$ be the sum of all values stored in all levels of the tree:

$$T(n) = f(n) + a\,f(n/b) + a^2\,f(n/b^2) + \cdots + a^i\,f(n/b^i) + \cdots + a^L\,f(n/b^L)$$

- Where $L = \log_b n$ is the depth of the tree
- Since $f(1) = \Theta(1)$, the last term of this summation is $\Theta(a^L) = \Theta(a^{\log_b n}) = \Theta(n^{\log_b a})$

A "Log Fact" Aside

- It's not hard to see that $a^{\log_b n} = n^{\log_b a}$

$$a^{\log_b n} = n^{\log_b a} \tag{25}$$
$$a^{\log_b n} = a^{\log_a n * \log_b a} \tag{26}$$
$$\log_b n = \log_a n * \log_b a \tag{27}$$

- We get to the last eqn by taking $\log_a$ of both sides
- The last eqn is true by our third basic log fact

Master Theorem

- We can now state the Master Theorem
- We will state it in a way slightly different from the book
- Note: The Master Method is just a "short cut" for the recursion tree method. It is less powerful than recursion trees.

Master Method

The recurrence $T(n) = aT(n/b) + f(n)$ can be solved as follows:

- If $a\,f(n/b) \leq K f(n)$ for some constant $K < 1$, then $T(n) = \Theta(f(n))$.
- If $a\,f(n/b) \geq K\,f(n)$ for some constant $K > 1$, then $T(n) = \Theta(n^{\log_b a})$.
- If $a\,f(n/b) = f(n)$, then $T(n) = \Theta(f(n)\log_b n)$.

Proof

- If $f(n)$ is a *constant factor larger* than $a\,f(n/b)$, then the sum is a descending geometric series. The sum of any geometric series is a constant times its largest term. In this case, the largest term is the first term $f(n)$.
- If $f(n)$ is a *constant factor smaller* than $a\,f(n/b)$, then the sum is an ascending geometric series. The sum of any geometric series is a constant times its largest term. In this case, this is the last term, which by our earlier argument is $\Theta(n^{\log_b a})$.
- Finally, if $a\,f(n/b) = f(n)$, then each of the $L+1$ terms in the summation is equal to $f(n)$.

Example

- $T(n) = T(3n/4) + n$
- If we write this as $T(n) = aT(n/b) + f(n)$, then $a = 1$,$b = 4/3$,$f(n) = n$
- Here $a\,f(n/b) = 3n/4$ is smaller than $f(n) = n$ by a factor of $4/3$, so $T(n) = \Theta(n)$

Example

- **Karatsuba's multiplication algorithm:** $\boldsymbol{T(n) = 3T(n/2) + n}$
- If we write this as $T(n) = aT(n/b) + f(n)$, then $a = 3$,$b = 2$,$f(n) = n$
- Here $a\, f(n/b) = 3n/2$ is bigger than $f(n) = n$ by a factor of $3/2$, so $T(n) = \Theta(n^{\log_2 3})$

Example

- **Mergesort:** $\boldsymbol{T(n) = 2T(n/2) + n}$
- If we write this as $T(n) = aT(n/b) + f(n)$, then $a = 2$,$b = 2$,$f(n) = n$
- Here $a\,f(n/b) = f(n)$, so $T(n) = \Theta(n \log n)$

Example

- $T(n) = T(n/2) + n \log n$
- If we write this as $T(n) = aT(n/b) + f(n)$, then $a = 1$,$b = 2$,$f(n) = n \log n$
- Here $a\, f(n/b) = n/2 \log n/2$ is smaller than $f(n) = n \log n$ by a constant factor, so $T(n) = \Theta(n \log n)$

In-Class Exercise

- Consider the recurrence: $\boldsymbol{T(n) = 4T(n/2) + n \lg n}$
- Q: What is $f(n)$ and $a\,f(n/b)$?
- Q: Which of the three cases does the recurrence fall under (when n is large)?
- Q: What is the solution to this recurrence?

In-Class Exercise

- Consider the recurrence: $\boldsymbol{T(n) = 2T(n/4) + n \lg n}$
- Q: What is $f(n)$ and $a\,f(n/b)$?
- Q: Which of the three cases does the recurrence fall under (when n is large)?
- Q: What is the solution to this recurrence?

Take Away

- Recursion tree and Master method are good tools for solving many recurrences
- However these methods are limited (they can't help us get guesses for recurrences like $f(n) = f(n-1) + f(n-2)$)
- For info on how to solve these other more difficult recurrences, review the notes on annihilators on the class web page.

Intro to Annihilators

"Listen and Understand! That terminator is out there. It can't be bargained with, it can't be reasoned with! It doesn't feel pity, remorse, or fear. And it absolutely will not stop, ever, until you are dead!" - The Terminator

- Suppose we are given a sequence of numbers $A = \langle a_0, a_1, a_2, \cdots \rangle$
- This might be a sequence like the Fibonacci numbers
- I.e. $A = \langle a_0, a_1, a_2, \dots) = (T(1), T(2), T(3), \cdots \rangle$

Annihilator Operators

We define three basic operations we can perform on this sequence:

1. Multiply the sequence by a constant: $cA = \langle ca_0, ca_1, ca_2, \cdots \rangle$
2. Shift the sequence to the left: $\mathbf{L}A = \langle a_1, a_2, a_3, \cdots \rangle$
3. Add two sequences: if $A = \langle a_0, a_1, a_2, \cdots \rangle$ and $B = \langle b_0, b_1, b_2, \cdots \rangle$, then $A + B = \langle a_0 + b_0, a_1 + b_1, a_2 + b_2, \cdots \rangle$

Annihilator Description

- We first express our recurrence as a sequence T
- *We use these three operators to "annihilate"* T*, i.e. make it all* 0*'s*
- Key rule: can't multiply by the constant 0
- We can then determine the solution to the recurrence from the sequence of operations performed to annihilate T

Example

- Consider the recurrence $T(n) = 2T(n-1)$, $T(0) = 1$
- If we solve for the first few terms of this sequence, we can see they are $\langle 2^0, 2^1, 2^2, 2^3, \cdots \rangle$
- Thus this recurrence becomes the sequence:

$$T = \langle 2^0, 2^1, 2^2, 2^3, \cdots \rangle$$

Example (II)

Let's annihilate $T = \langle 2^0, 2^1, 2^2, 2^3, \cdots \rangle$

- Multiplying by a constant $c = 2$ gets:

$$2T = \langle 2 * 2^0, 2 * 2^1, 2 * 2^2, 2 * 2^3, \cdots \rangle = \langle 2^1, 2^2, 2^3, 2^4, \cdots \rangle$$

- Shifting one place to the left gets $\mathbf{L}T = \langle 2^1, 2^2, 2^3, 2^4, \cdots \rangle$
- Adding the sequence $\mathbf{L}T$ and $-2T$ gives:

$$\mathbf{L}T - 2T = \langle 2^1 - 2^1, 2^2 - 2^2, 2^3 - 2^3, \cdots \rangle = \langle 0, 0, 0, \cdots \rangle$$

- The annihilator of T is thus $\mathbf{L} - 2$

Distributive Property

- The distributive property holds for these three operators
- Thus can rewrite $\mathbf{L}T - 2T$ as $(\mathbf{L} - 2)T$
- The operator $(\mathbf{L} - 2)$ annihilates T (makes it the sequence of all 0's)
- Thus $(\mathbf{L} - 2)$ is called the *annihilator* of T

0, the “Forbidden Annihilator”

- Multiplication by 0 will annihilate *any* sequence
- Thus we disallow multiplication by 0 as an operation
- In particular, we disallow $(c-c) = 0$ for any c as an annihilator
- Must always have at least one **L** operator in any annihilator!

Uniqueness

- An annihilator annihilates exactly *one* type of sequence
- In general, the annihilator $\mathbf{L} - c$ annihilates any sequence of the form $\langle a_0 c^n \rangle$
- If we find the annihilator, we can find the type of sequence, and thus solve the recurrence
- We will need to use the base case for the recurrence to solve for the constant a_0

If we apply operator $(\mathbf{L} - 3)$ to sequence T above, it fails to annihilate T

$$
\begin{aligned}
(\mathbf{L} - 3)T &= \mathbf{L}T + (-3)T \\
&= \langle 2^1, 2^2, 2^3, \cdots \rangle + \langle -3 \times 2^0, -3 \times 2^1, -3 \times 2^2, \cdots \rangle \\
&= \langle (2-3) \times 2^0, (2-3) \times 2^1, (2-3) \times 2^2, \cdots \rangle \\
&= (2-3)T = -T
\end{aligned}
$$

Example (II)

What does $(\mathbf{L}-c)$ do to other sequences $A = \langle a_0 d^n \rangle$ when $d \neq c$?:

$$\begin{aligned}
(\mathbf{L}-c)A &= (\mathbf{L}-c)\langle a_0, a_0 d, a_0 d^2, a_0 d^3, \cdots \rangle \\
&= \mathbf{L}\langle a_0, a_0 d, a_0 d^2, a_0 d^3, \cdots \rangle - c\langle a_0, a_0 d, a_0 d^2, a_0 d^3, \cdots \rangle \\
&= \langle a_0 d, a_0 d^2, a_0 d^3, \cdots \rangle - \langle c a_0, c a_0 d, c a_0 d^2, c a_0 d^3, \cdots \rangle \\
&= \langle a_0 d - c a_0, a_0 d^2 - c a_0 d, a_0 d^3 - c a_0 d^2, \cdots \rangle \\
&= \langle (d-c) a_0, (d-c) a_0 d, (d-c) a_0 d^2, \cdots \rangle \\
&= (d-c)\langle a_0, a_0 d, a_0 d^2, \cdots \rangle \\
&= (d-c)A
\end{aligned}$$

Uniqueness

- The last example implies that an annihilator annihilates one type of sequence, but does not annihilate other types of sequences
- Thus Annihilators can help us classify sequences, and thereby solve recurrences

- The annihilator $\mathbf{L} - a$ annihilates any sequence of the form $\langle c_1 a^n \rangle$

Example

First calculate the annihilator:

- Recurrence: $T(n) = 4 * T(n-1)$, $T(0) = 2$
- Sequence: $T = \langle 2, 2*4, 2*4^2, 2*4^3, \cdots \rangle$
- Calulate the annihilator:
  - $\mathbf{L}T = \langle 2*4, 2*4^2, 2*4^3, 2*4^4, \cdots \rangle$
  - $4T = \langle 2*4, 2*4^2, 2*4^3, 2*4^4, \cdots \rangle$
  - Thus $\mathbf{L}T - 4T = \langle 0, 0, 0, \cdots \rangle$
  - And so $\mathbf{L} - 4$ is the annihilator

Example (II)

Now use the annihilator to solve the recurrence

- Look up the annihilator in the "Lookup Table"
- It says: "The annihilator $\mathbf{L} - 4$ annihilates any sequence of the form $\langle c_1 4^n \rangle$"
- Thus $T(n) = c_1 4^n$, but what is c_1?
- We know $T(0) = 2$, so $T(0) = c_1 4^0 = 2$ and so $c_1 = 2$
- Thus $T(n) = 2 * 4^n$

In Class Exercise

Consider the recurrence $T(n) = 3 * T(n-1)$, $T(0) = 3$,

- Q1: Calculate $T(0)$,$T(1)$,$T(2)$ and $T(3)$ and write out the sequence T
- Q2: Calculate $\mathbf{L}T$, and use it to compute the annihilator of T
- Q3: Look up this annihilator in the lookup table to get the general solution of the recurrence for $T(n)$
- Q4: Now use the base case $T(0) = 3$ to solve for the constants in the general solution

Remaining Outline

- Annihilators with Multiple Operators
- Annihilators for recurrences with non-homogeneous terms
- Transformations

Multiple Operators

- We can apply multiple operators to a sequence
- For example, we can multiply by the constant c and then by the constant d to get the operator cd
- We can also multiply by c and then shift left to get $c\mathbf{L}T$ which is the same as $\mathbf{L}cT$
- We can also shift the sequence twice to the left to get $\mathbf{L}\mathbf{L}T$ which we'll write in shorthand as $\mathbf{L}^2T$

Multiple Operators

- We can string operators together to annihilate more complicated sequences
- Consider: $T = \langle 2^0 + 3^0, 2^1 + 3^1, 2^2 + 3^2, \cdots \rangle$
- We know that $(\mathbf{L} - 2)$ annihilates the powers of 2 while leaving the powers of 3 essentially untouched
- Similarly, $(\mathbf{L} - 3)$ annihilates the powers of 3 while leaving the powers of 2 essentially untouched
- Thus if we apply both operators, we'll see that $(\mathbf{L} - 2)(\mathbf{L} - 3)$ annihilates the sequence T

The Details

- Consider: $T = \langle a^0 + b^0, a^1 + b^1, a^2 + b^2, \cdots \rangle$
- $\mathbf{L}T = \langle a^1 + b^1, a^2 + b^2, a^3 + b^3, \cdots \rangle$
- $aT = \langle a^1 + a * b^0, a^2 + a * b^1, a^3 + a * b^2, \cdots \rangle$
- $\mathbf{L}T - aT = \langle (b-a)b^0, (b-a)b^1, (b-a)b^2, \cdots \rangle$
- We know that $(\mathbf{L} - a)T$ annihilates the a terms and multiplies the b terms by $b - a$ (a constant)
- Thus $(\mathbf{L} - a)T = \langle (b-a)b^0, (b-a)b^1, (b-a)b^2, \cdots \rangle$
- And so the sequence $(\mathbf{L} - a)T$ is annihilated by $(\mathbf{L} - b)$
- Thus the annihilator of T is $(\mathbf{L} - b)(\mathbf{L} - a)$

Key Point

- In general, the annihilator $(\mathbf{L} - a)(\mathbf{L} - b)$ (where $a \neq b$) will anihilate *only* all sequences of the form $\langle c_1 a^n + c_2 b^n \rangle$
- We will often multiply out $(\mathbf{L} - a)(\mathbf{L} - b)$ to $\mathbf{L}^2 - (a+b)\mathbf{L} + ab$
- Left as an exercise to show that $(\mathbf{L} - a)(\mathbf{L} - b)T$ is the same as $(\mathbf{L}^2 - (a+b)\mathbf{L} + ab)T$

Lookup Table

- The annihilator $\mathbf{L}-a$ annihilates sequences of the form $\langle c_1 a^n \rangle$
- The annihilator $(\mathbf{L}-a)(\mathbf{L}-b)$ (where $a \neq b$) anihilates sequences of the form $\langle c_1 a^n + c_2 b^n \rangle$

Fibonnaci Sequence

- We now know enough to solve the Fibonnaci sequence
- Recall the Fibonnaci recurrence is $T(0) = 0$, $T(1) = 1$, and $T(n) = T(n-1) + T(n-2)$
- Let T_n be the n-th element in the sequence
- Then we've got:

$$T = \langle T_0, T_1, T_2, T_3, \cdots \rangle \quad (28)$$

$$\mathbf{L}T = \langle T_1, T_2, T_3, T_4, \cdots \rangle \quad (29)$$

$$\mathbf{L}^2 T = \langle T_2, T_3, T_4, T_5, \cdots \rangle \quad (30)$$

- Thus $\mathbf{L}^2 T - \mathbf{L}T - T = \langle 0, 0, 0, \cdots \rangle$
- In other words, $\mathbf{L}^2 - \mathbf{L} - 1$ is an annihilator for T

Factoring

- $\mathbf{L}^2 - \mathbf{L} - 1$ is an annihilator that is not in our lookup table
- However, we can *factor* this annihilator (using the quadratic formula) to get something similar to what's in the lookup table
- $\mathbf{L}^2 - \mathbf{L} - 1 = (\mathbf{L} - \phi)(\mathbf{L} - \hat{\phi})$, where $\phi = \frac{1+\sqrt{5}}{2}$ and $\hat{\phi} = \frac{1-\sqrt{5}}{2}$.

Quadratic Formula

"Me fail English? That's Unpossible!" - *Ralph, the Simpsons*

High School Algebra Review:

- To factor something of the form $ax^2 + bx + c$, we use the *Quadratic Formula*:
- $ax^2 + bx + c$ factors into $(x - \phi)(x - \hat{\phi})$, where:

$$\phi = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \quad (31)$$

$$\hat{\phi} = \frac{-b - \sqrt{b^2 - 4ac}}{2a} \quad (32)$$

Example

- To factor: $\mathbf{L}^2 - \mathbf{L} - 1$
- Rewrite: $1 * \mathbf{L}^2 - 1 * \mathbf{L} - 1$, $a = 1$, $b = -1$, $c = -1$
- From Quadratic Formula: $\phi = \frac{1+\sqrt{5}}{2}$ and $\hat{\phi} = \frac{1-\sqrt{5}}{2}$
- So $\mathbf{L}^2 - \mathbf{L} - 1$ factors to $(\mathbf{L} - \phi)(\mathbf{L} - \hat{\phi})$

Back to Fibonnaci

- Recall the Fibonnaci recurrence is $T(0) = 0$, $T(1) = 1$, and $T(n) = T(n-1) + T(n-2)$
- We've shown the annihilator for T is $(\mathbf{L} - \phi)(\mathbf{L} - \hat{\phi})$, where $\phi = \frac{1+\sqrt{5}}{2}$ and $\hat{\phi} = \frac{1-\sqrt{5}}{2}$
- If we look this up in the "Lookup Table", we see that the sequence T must be of the form $\langle c_1\phi^n + c_2\hat{\phi}^n \rangle$
- All we have left to do is solve for the constants c_1 and c_2
- Can use the base cases to solve for these

Finding the Constants

- We know $T = \langle c_1\phi^n + c_2\hat{\phi}^n \rangle$, where $\phi = \frac{1+\sqrt{5}}{2}$ and $\hat{\phi} = \frac{1-\sqrt{5}}{2}$
- We know

$$
\begin{aligned}
T(0) &= c_1 + c_2 = 0 \quad (33) \\
T(1) &= c_1\phi + c_2\hat{\phi} = 1 \quad (34)
\end{aligned}
$$

- We've got two equations and two unknowns
- Can solve to get $c_1 = \frac{1}{\sqrt{5}}$ and $c_2 = -\frac{1}{\sqrt{5}}$,

The Punchline

- Recall Fibonnaci recurrence: $T(0) = 0$, $T(1) = 1$, and $T(n) = T(n-1) + T(n-2)$
- The final explicit formula for $T(n)$ is thus:

$$T(n) = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n$$

(Amazingly, $T(n)$ is *always* an integer, in spite of all of the square roots in its formula.)

A Problem

- Our lookup table has a big gap: What does $(\mathbf{L} - a)(\mathbf{L} - a)$ annihilate?
- It turns out it annihilates sequences such as $\langle na^n \rangle$

Example

$$
\begin{aligned}
(\mathbf{L}-a)\langle na^n\rangle &= \langle (n+1)a^{n+1} - (a)na^n\rangle \\
&= \langle (n+1)a^{n+1} - na^{n+1}\rangle \\
&= \langle (n+1-n)a^{n+1}\rangle \\
&= \langle a^{n+1}\rangle \\
(\mathbf{L}-a)^2\langle na^n\rangle &= (\mathbf{L}-a)\langle a^{n+1}\rangle \\
&= \langle 0\rangle
\end{aligned}
$$

Generalization

- It turns out that $(\mathbf{L} - a)^d$ annihilates sequences of the form $\langle p(n)a^n \rangle$ where $p(n)$ is any polynomial of degree $d - 1$
- Example: $(\mathbf{L} - 1)^3$ annihilates the sequence $\langle n^2 * 1^n \rangle = \langle 1, 4, 9, 16, 25 \rangle$ since $p(n) = n^2$ is a polynomial of degree $d - 1 = 2$

Lookup Table

- $(\mathbf{L} - a)$ annihilates only all sequences of the form $\langle c_0 a^n \rangle$
- $(\mathbf{L}-a)(\mathbf{L}-b)$ annihilates only all sequences of the form $\langle c_0 a^n + c_1 b^n \rangle$
- $(\mathbf{L} - a_0)(\mathbf{L} - a_1) \ldots (\mathbf{L} - a_k)$ annihilates only sequences of the form $\langle c_0 a_0^n + c_1 a_1^n + \ldots c_k a_k^n \rangle$, here $a_i \neq a_j$, when $i \neq j$
- $(\mathbf{L} - a)^2$ annihilates only sequences of the form $\langle (c_0 n + c_1) a^n \rangle$
- $(\mathbf{L} - a)^k$ annihilates only sequences of the form $\langle p(n) a^n \rangle$, $degree(p(n)) = k - 1$

Lookup Table (Final!)

$$(\mathbf{L}-a_0)^{b_0}(\mathbf{L}-a_1)^{b_1}\ldots(\mathbf{L}-a_k)^{b_k}$$

annihilates only sequences of the form:

$$\langle p_1(n)a_0^n + p_2(n)a_1^n + \ldots p_k(n)a_k^n\rangle$$

where $p_i(n)$ is a polynomial of degree $b_i - 1$ (and $a_i \neq a_j$, when $i \neq j$)

Examples

- Q: What does $(\mathbf{L}-3)(\mathbf{L}-2)(\mathbf{L}-1)$ annihilate?
- A: $c_0 1^n + c_1 2^n + c_2 3^n$
- Q: What does $(\mathbf{L}-3)^2(\mathbf{L}-2)(\mathbf{L}-1)$ annihilate?
- A: $c_0 1^n + c_1 2^n + (c_2 n + c_3)3^n$
- Q: What does $(\mathbf{L}-1)^4$ annihilate?
- A: $(c_0 n^3 + c_1 n^2 + c_2 n + c_3)1^n$
- Q: What does $(\mathbf{L}-1)^3(\mathbf{L}-2)^2$ annihilate?
- A: $(c_0 n^2 + c_1 n + c_2)1^n + (c_3 n + c_4)2^n$

Annihilator Method

- Write down the annihilator for the recurrence
- Factor the annihilator
- Look up the factored annihilator in the "Lookup Table" to get general solution
- Solve for constants of the general solution by using initial conditions

Annihilator Method

- Write down the annihilator for the recurrence
- Factor the annihilator
- Look up the factored annihilator in the "Lookup Table" to get general solution
- Solve for constants of the general solution by using initial conditions

Lookup Table

$$(\mathbf{L} - a_0)^{b_0}(\mathbf{L} - a_1)^{b_1} \ldots (\mathbf{L} - a_k)^{b_k}$$

annihilates only sequences of the form:

$$\langle p_0(n)a_0^n + p_1(n)a_1^n + \ldots p_k(n)a_k^n \rangle$$

where $p_i(n)$ is a polynomial of degree $b_i - 1$ (and $a_i \neq a_j$, when $i \neq j$)

Examples

- Q: What does $(\mathbf{L}-3)(\mathbf{L}-2)(\mathbf{L}-1)$ annihilate?
- A: $c_0 1^n + c_1 2^n + c_2 3^n$
- Q: What does $(\mathbf{L}-3)^2(\mathbf{L}-2)(\mathbf{L}-1)$ annihilate?
- A: $c_0 1^n + c_1 2^n + (c_2 n + c_3) 3^n$
- Q: What does $(\mathbf{L}-1)^4$ annihilate?
- A: $(c_0 n^3 + c_1 n^2 + c_2 n + c_3) 1^n$
- Q: What does $(\mathbf{L}-1)^3(\mathbf{L}-2)^2$ annihilate?
- A: $(c_0 n^2 + c_1 n + c_2) 1^n + (c_3 n + c_4) 2^n$

Example

Consider the recurrence $T(n) = 7T(n-1) - 16T(n-2) + 12T(n-3)$, $T(0) = 1$, $T(1) = 5$, $T(2) = 17$

- **Write down the annihilator**: From the definition of the sequence, we can see that $\mathbf{L}^3 T - 7\mathbf{L}^2 T + 16\mathbf{L}T - 12T = 0$, so the annihilator is $\mathbf{L}^3 - 7\mathbf{L}^2 + 16\mathbf{L} - 12$
- **Factor the annihilator**: We can factor by hand or using a computer program to get $\mathbf{L}^3 - 7\mathbf{L}^2 + 16\mathbf{L} - 12 = (\mathbf{L}-2)^2(\mathbf{L}-3)$
- **Look up to get general solution**: The annihilator $(\mathbf{L}-2)^2(\mathbf{L}-3)$ annihilates sequences of the form $\langle (c_0 n + c_1)2^n + c_2 3^n \rangle$
- **Solve for constants**: $T(0) = 1 = c_1 + c_2$, $T(1) = 5 = 2c_0 + 2c_1 + 3c_2$, $T(2) = 17 = 8c_0 + 4c_1 + 9c_2$. We've got three equations and three unknowns. Solving by hand, we get that $c_0 = 1$,$c_1 = 0$,$c_2 = 1$. **Thus:** $T(n) = n2^n + 3^n$

Example (II)

Consider the recurrence $T(n) = 2T(n-1) - T(n-2)$, $T(0) = 0$, $T(1) = 1$

- **Write down the annihilator**: From the definition of the sequence, we can see that $\mathbf{L}^2 T - 2\mathbf{L}T + T = 0$, so the annihilator is $\mathbf{L}^2 - 2\mathbf{L} + 1$
- **Factor the annihilator**: We can factor by hand or using the quadratic formula to get $\mathbf{L}^2 - 2\mathbf{L} + 1 = (\mathbf{L} - 1)^2$
- **Look up to get general solution**: The annihilator $(\mathbf{L} - 1)^2$ annihilates sequences of the form $(c_0 n + c_1)1^n$
- **Solve for constants**: $T(0) = 0 = c_1$, $T(1) = 1 = c_0 + c_1$, We've got two equations and two unknowns. Solving by hand, we get that $c_0 = 0$,$c_1 = 1$. **Thus:** $T(n) = n$

Consider the recurrence $T(n) = 6T(n-1) - 9T(n-2)$, $T(0) = 1$, $T(1) = 6$

- Q1: What is the annihilator of this sequence?
- Q2: What is the factored version of the annihilator?
- Q3: What is the general solution for the recurrence?
- Q4: What are the constants in this general solution?

(Note: You can check that your general solution works for $T(2)$)

Non-homogeneous terms

- Consider a recurrence of the form $T(n) = T(n-1) + T(n-2) + k$ where k is some constant
- The terms in the equation involving T (i.e. $T(n-1)$ and $T(n-2)$) are called the *homogeneous* terms
- The other terms (i.e.k) are called the *non-homogeneous* terms

Example

- In a *height-balanced tree*, the height of two subtrees of any node differ by at most one
- Let $T(n)$ be the smallest number of nodes needed to obtain a height balanced binary tree of height n
- Q: What is a recurrence for $T(n)$?
- A: Divide this into smaller subproblems
  - To get a height-balanced tree of height n with the smallest number of nodes, need one subtree of height $n-1$, and one of height $n-2$, plus a root node
  - Thus $T(n) = T(n-1) + T(n-2) + 1$

Example

- Let's solve this recurrence: $T(n) = T(n-1) + T(n-2) + 1$ (Let $T_n = T(n)$, and $T = \langle T_n \rangle$)
- We know that $(\mathbf{L}^2 - \mathbf{L} - 1)$ annihilates the homogeneous terms
- Let's apply it to the entire equation:

$$\begin{aligned}(\mathbf{L}^2 - \mathbf{L} - 1)\langle T_n \rangle &= \mathbf{L}^2 \langle T_n \rangle - \mathbf{L}\langle T_n \rangle - 1\langle T_n \rangle \\ &= \langle T_{n+2} \rangle - \langle T_{n+1} \rangle - \langle T_n \rangle \\ &= \langle T_{n+2} - T_{n+1} - T_n \rangle \\ &= \langle 1, 1, 1, \cdots \rangle\end{aligned}$$

Example

- This is close to what we want but we still need to annihilate $\langle 1, 1, 1, \cdots \rangle$
- It's easy to see that $\mathbf{L} - 1$ annihilates $\langle 1, 1, 1, \cdots \rangle$
- Thus $(\mathbf{L}^2 - \mathbf{L} - 1)(\mathbf{L} - 1)$ annihilates $T(n) = T(n-1) + T(n-2) + 1$
- When we factor, we get $(\mathbf{L} - \phi)(\mathbf{L} - \hat{\phi})(\mathbf{L} - 1)$, where $\phi = \frac{1+\sqrt{5}}{2}$ and $\hat{\phi} = \frac{1-\sqrt{5}}{2}$.

Lookup

- Looking up $(\mathbf{L}-\phi)(\mathbf{L}-\widehat{\phi})(\mathbf{L}-1)$ in the table
- We get $T(n) = c_1\phi^n + c_2\widehat{\phi}^n + c_3 1^n$
- If we plug in the appropriate initial conditions, we can solve for these three constants
- We'll need to get equations for $T(2)$ in addition to $T(0)$ and $T(1)$

General Rule

To find the annihilator for recurrences with non-homogeneous terms, do the following:

- Find the annihilator a_1 for the homogeneous part
- Find the annihilator a_2 for the non-homogeneous part
- The annihilator for the whole recurrence is then $a_1 a_2$

Another Example

- Consider $T(n) = T(n-1) + T(n-2) + 2$.
- The residue is $\langle 2, 2, 2, \cdots \rangle$ and
- The annihilator is still $(\mathbf{L}^2 - \mathbf{L} - 1)(\mathbf{L} - 1)$, but the equation for $T(2)$ changes!

Another Example

- Consider $T(n) = T(n-1) + T(n-2) + 2^n$.
- The residue is $\langle 1, 2, 4, 8, \cdots \rangle$ and
- The annihilator is now $(\mathbf{L}^2 - \mathbf{L} - 1)(\mathbf{L} - 2)$.

Another Example

- Consider $T(n) = T(n-1) + T(n-2) + n$.
- The residue is $\langle 1, 2, 3, 4, \cdots \rangle$
- The annihilator is now $(\mathbf{L}^2 - \mathbf{L} - 1)(\mathbf{L} - 1)^2$.

Another Example

- Consider $T(n) = T(n-1) + T(n-2) + n^2$.
- The residue is $\langle 1, 4, 9, 16, \cdots \rangle$ and
- The annihilator is $(\mathbf{L}^2 - \mathbf{L} - 1)(\mathbf{L} - 1)^3$.

Another Example

- Consider $T(n) = T(n-1) + T(n-2) + n^2 - 2^n$.
- The residue is $\langle 1-1, 4-4, 9-8, 16-16, \cdots \rangle$ and the
- The annihilator is $(\mathbf{L}^2 - \mathbf{L} - 1)(\mathbf{L} - 1)^3(\mathbf{L} - 2)$.

In Class Exercise

- Consider $T(n) = 3 * T(n-1) + 3^n$
- Q1: What is the homogeneous part, and what annihilates it?
- Q2: What is the non-homogeneous part, and what annihilates it?
- Q3: What is the annihilator of $T(n)$, and what is the general form of the recurrence?

Limitations

- Our method does not work on $T(n) = T(n-1) + \frac{1}{n}$ or $T(n) = T(n-1) + \lg n$
- The problem is that $\frac{1}{n}$ and $\lg n$ do not have annihilators.
- Our tool, as it stands, is limited.
- Key idea for strengthening it is *transformations*

Transformations Idea

- Consider the recurrence giving the run time of mergesort $T(n) = 2T(n/2) + kn$ (for some constant k), $T(1) = 1$
- How do we solve this?
- We have no technique for annihilating terms like $T(n/2)$
- However, we can *transform* the recurrence into one with which we can work

- Let $n = 2^i$ and rewrite $T(n)$:
- $T(2^0) = 1$ and $T(2^i) = 2T(\frac{2^i}{2}) + k2^i = 2T(2^{i-1}) + k2^i$
- Now define a new sequence t as follows: $t(i) = T(2^i)$
- Then $t(0) = 1$, $t(i) = 2t(i-1) + k2^i$

Now Solve

- We've got a new recurrence: $t(0) = 1$, $t(i) = 2t(i-1) + k2^i$
- We can easily find the annihilator for this recurrence
- $(\mathbf{L}-2)$ annihilates the homogeneous part, $(\mathbf{L}-2)$ annihilates the non-homogeneous part, So $(\mathbf{L}-2)(\mathbf{L}-2)$ annihilates $t(i)$
- Thus $t(i) = (c_1 i + c_2)2^i$

Reverse Transformation

- We've got a solution for $t(i)$ and we want to transform this into a solution for $T(n)$
- Recall that $t(i) = T(2^i)$ and $2^i = n$

$$\begin{aligned} t(i) &= (c_1 i + c_2)2^i & (35) \\ T(2^i) &= (c_1 i + c_2)2^i & (36) \\ T(n) &= (c_1 \lg n + c_2)n & (37) \\ &= c_1 n \lg n + c_2 n & (38) \\ &= O(n \lg n) & (39) \end{aligned}$$

Success!

Let's recap what just happened:

- We could not find the annihilator of $T(n)$ so:
- We did a *transformation* to a recurrence we could solve, $t(i)$ (we let $n = 2^i$ and $t(i) = T(2^i)$)
- We found the annihilator for $t(i)$, and solved the recurrence for $t(i)$
- We *reverse transformed* the solution for $t(i)$ back to a solution for $T(n)$

Another Example

- Consider the recurrence $T(n) = 9T(\frac{n}{3}) + kn$, where $T(1) = 1$ and k is some constant
- Let $n = 3^i$ and rewrite $T(n)$:
- $T(3^0) = 1$ and $T(3^i) = 9T(3^{i-1}) + k3^i$
- Now define a sequence t as follows $t(i) = T(3^i)$
- Then $t(0) = 1$, $t(i) = 9t(i-1) + k3^i$

Now Solve

- $t(0) = 1$, $t(i) = 9t(i-1) + k3^i$
- This is annihilated by $(\mathbf{L} - 9)(\mathbf{L} - 3)$
- So $t(i)$ is of the form $t(i) = c_1 9^i + c_2 3^i$

Reverse Transformation

- $t(i) = c_1 9^i + c_2 3^i$
- Recall: $t(i) = T(3^i)$ and $3^i = n$

$$
\begin{aligned}
t(i) &= c_1 9^i + c_2 3^i \\
T(3^i) &= c_1 9^i + c_2 3^i \\
T(n) &= c_1 (3^i)^2 + c_2 3^i \\
&= c_1 n^2 + c_2 n \\
&= O(n^2)
\end{aligned}
$$

In Class Exercise

Consider the recurrence $T(n) = 2T(n/4) + kn$, where $T(1) = 1$, and k is some constant

- Q1: What is the transformed recurrence $t(i)$? How do we rewrite n and $T(n)$ to get this sequence?
- Q2: What is the annihilator of $t(i)$? What is the solution for the recurrence $t(i)$?
- Q3: What is the solution for $T(n)$? (i.e. do the reverse transformation)

A Final Example

Not always obvious what sort of transformation to do:

- Consider $T(n) = 2T(\sqrt{n}) + \log n$
- Let $n = 2^i$ and rewrite $T(n)$:
- $T(2^i) = 2T(2^{i/2}) + i$
- Define $t(i) = T(2^i)$:
- $t(i) = 2t(i/2) + i$

A Final Example

- This final recurrence is something we know how to solve!
- $t(i) = O(i \log i)$
- The reverse transform gives:

$$\begin{aligned} t(i) &= O(i \log i) && (40) \\ T(2^i) &= O(i \log i) && (41) \\ T(n) &= O(\log n \log \log n) && (42) \end{aligned}$$