Recurrences

CS 206: Discrete Structures II

Recurrence

A recurrence describes a sequence of numbers.

Here's a recurrence for the sequence 1, 2, 3, ...:

$$T(n) = \begin{cases} 1 & n = 1 \\ T(n-1) + 1 & n \geq 2 \end{cases}$$

Finding closed forms

Solving techniques

- guess and verify
- plug and chug

Classes of recurrences

- linear
- divide and conquer

Towers of Hanoi

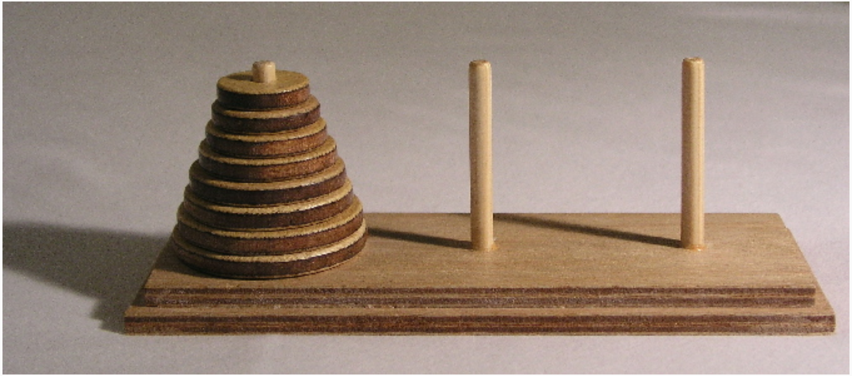


Image credit: Evanherk

https://commons.wikimedia.org/wiki/File:Tower_of_Hanoi.jpeg

Towers of Hanoi recursive solution

```python
# move n discs from src peg to dest peg
def moveDiscs(src, dest, n):
    if n == 1:
        moveDisc(src, dest)
    else:
        moveDiscs(src, otherPeg, n - 1)
        moveDisc(src, dest)
        moveDiscs(otherPeg, dest, n - 1)
```

Towers of Hanoi: guess and verify

Let $T(n)$ be the number of steps the solution takes:

$$T(n) = \begin{cases} 1 & n = 1 \\ 2T(n-1) + 1 & n \geq 2 \end{cases}$$

What is a closed form solution for this?

Towers of Hanoi: guess and verify

We can calculate a few values by hand:

n	$T(n)$
1	1
2	3
3	7
4	15
5	31
6	63
7	127

Looks rather like $T(n) = 2^n - 1$... but we'd have to prove it!

Proof.

By induction. When $n = 1$, we have $2^1 - 1 = 1$ and $T(1)$ is defined to be 1.

Then if $T(k) = 2^k - 1$, we have that

$$\begin{aligned} T(k+1) &= 2T(k) + 1 && \text{(def. of } T(n)\text{)} \\ &= 2(2^k - 1) + 1 && \text{(ind. hyp.)} \\ &= 2^{k+1} - 1 && \text{(simplify)} \end{aligned}$$

$\square$

Towers of Hanoi: plug and chug

Another approach is to expand the recurrence a few times:

$$\begin{aligned}
T(n) &= 2T(n-1)+1 \\
&= 2(2T(n-2)+1)+1 \\
&= 4T(n-2)+2+1 \\
&= 4(2T(n-3)+1)+2+1 \\
&= 8T(n-3)+4+2+1 \\
&= 8(2T(n-4)+1)+4+2+1 \\
&= 16T(n-4)+8+4+2+1
\end{aligned}$$

Towers of Hanoi: plug and chug

This looks like

$$T(n) = 2^k T(n-k) + \sum_{i=0}^{k-1} 2^i$$

or

$$T(n) = 2^k T(n-k) + 2^k - 1$$

Towers of Hanoi: plug and chug

Theorem

For all $k \geq 1$, $T(n) = 2^k T(n-k) + 2^k - 1$.

Proof.

The base case ($k = 1$) gives the original recurrence:

$$\begin{aligned} T(n) &= 2^1 T(n-1) + 2^1 - 1 \\ &= 2T(n-1) + 1 \end{aligned}$$

Verify the inductive step by expanding it once more:

$$\begin{aligned} T(n) &= 2^k T(n-k) + 2^k - 1 \\ &= 2^k (2T(n-k-1) + 1) + 2^k - 1 \\ &= 2^{k+1} T(n-k-1) + 2^{k+1} - 1 \end{aligned}$$

Towers of Hanoi: plug and chug

Then plug in values for early terms of the sequence.

Let $k = n - 1$, then

$$\begin{aligned} T(n) &= 2^k T(n-k) + 2^k - 1 \\ &= 2^{n-1} T(n-(n-1)) + 2^{n-1} - 1 \\ &= 2^{n-1} T(1) + 2^{n-1} - 1 \\ &= 2^{n-1} + 2^{n-1} - 1 \\ &= 2^n - 1 \end{aligned}$$

Merge sort

To sort an array with mergesort:

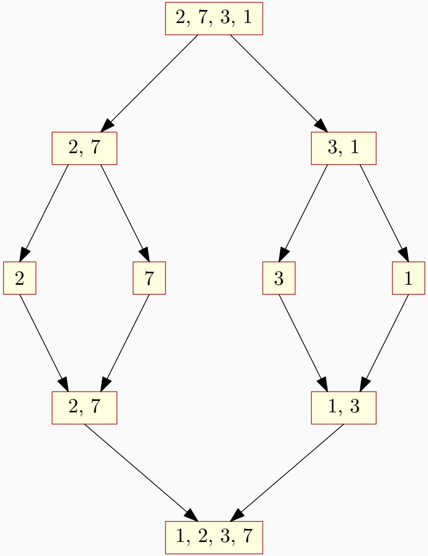
- divide it in half
- recursively sort each
- merge the results

Note: an array of size 1 is already sorted!

Merge sort pseudocode

```
def mergesort(arr):
    if len(arr) == 1:
        return arr
    leftSorted  = mergesort(leftHalf(arr))
    rightSorted = mergesort(rightHalf(arr))
    return merge(leftSorted, rightSorted)
```

Merge sort example



Merge sort runtime

How many comparisons are required?

$$T(n) = \begin{cases} 0 & n = 1 \\ 2T\left(\frac{n}{2}\right) + n - 1 & n \geq 2 \end{cases}$$

Merge sort: guess and verify

n	$T(n)$
1	0
2	1
4	5
8	17
16	49
32	129

The pattern doesn't seem obvious...

Merge sort: plug and chug, take 1

If we can simplify slightly, let $T(n) = 2T(n/2) + n$:

$$\begin{aligned}
T(n) &= 2T(n/2) + n = 2(2T(n/4) + n/2) + n \\
&= 4T(n/4) + 2n = 4(2T(n/8) + n/4) + 2n \\
&= 8T(n/8) + 3n \\
&= \ldots \\
&= 2^k T(n/2^k) + kn
\end{aligned}$$

Merge sort: plug and chug, take 1

We have:

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

Let $n = 2^k$:

$$\begin{aligned} T(n) &= nT(n/n) + kn \\ &= nT(1) + kn \\ &= n \cdot 0 + kn \\ &= kn \end{aligned}$$

Merge sort: plug and chug, take 1

Now we have:

$$T(n) = kn$$

To get rid of k, observe that $n = 2^k$ implies $k = \log n$:

$$T(n) = n \log n$$

Merge sort: plug and chug, take 2

Going back to our actual recurrence:

$$\begin{aligned}
T(n) &= 2T(n/2) + n - 1 \\
&= 2(2T(n/4) + n/2 - 1) + n - 1 \\
&= 4T(n/4) + 2n - 3 \\
&= 4(2T(n/8) + n/4 - 1) + 2n - 3 \\
&= 8T(n/8) + 3n - 7 \\
&= 8(2T(n/16) + n/8 - 1) + 3n - 7 \\
&= 16T(n/16) + 4n - 15
\end{aligned}$$

Seems like $2^k T(n/2^k) + kn - (2^k - 1)$...

Merge sort: plug and chug, take 2

Proof.

Base case: $2^1 T(n/2^1) + 1 \cdot n - (2^1 - 1) = 2T(n/2) + n - 1$

Inductive case:

$$\begin{aligned}
T(n) &= 2^k T(n/2^k) + kn - (2^k - 1) \\
&= 2^k (2T(n/2^{k+1}) + n/2^k - 1) + kn - (2^k - 1) \\
&= 2^{k+1} T(n/2^{k+1}) + n - 2^k + kn - 2^k + 1 \\
&= 2^{k+1} T(n/2^{k+1}) + (k+1)n - 2^{k+1} + 1 \\
&= 2^{k+1} T(n/2^{k+1}) + (k+1)n - (2^{k+1} - 1)
\end{aligned}$$

$\square$

Merge sort: plug and chug

Let $k = \log n$. Then $2^k = 2^{\log n} = n$:

$$\begin{aligned} 2^k T(n/2^k) + kn - (2^k - 1) &= nT(n/n) + kn - (n-1) \\ &= nT(1) + n\log n - n + 1 \\ &= n\log n - n + 1 \end{aligned}$$

Climbing stairs

Let’s suppose you can either climb one step or two.

How many ways could you climb n steps?

n	#ways
0	1
1	1
2	2
3	3

Climbing stairs

Climbing one step reduces the problem to $n-1$ steps.

Climbing two steps reduces the problem to $n-2$ steps.

$$f(n) = f(n-1) + f(n-2)$$

Fibonacci

The famous Fibonacci sequence:

$$f(n) = f(n-1) + f(n-2)$$

where $f(0) = f(1) = 1$.

Fibonacci

n	$f(n)$
0	1
1	1
2	2
3	3
4	5
5	8
6	13
7	21

Homogeneous linear recurrences

A homogeneous linear recurrence of order d has the form:

$$f(n) = a_1 f(n-1) + a_2 f(n-2) + \cdots + a_d f(n-d)$$

Fibonacci

Let's guess: $f(n) = x^n$ for some x

Then:

$$f(n) = f(n-1) + f(n-2)$$
$$x^n = x^{n-1} + x^{n-2}$$

Divide by x^{n-2}:

$$x^2 = x + 1$$

Then

$$x = \frac{1 \pm \sqrt{5}}{2}$$

Fibonacci

Since $f(n) = x^n$, this means

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

Sum of homogeneous linear recurrences

Theorem

If $f(n)$ and $g(n)$ are solutions to a homogeneous linear recurrence, then for all $\alpha, \beta \in \mathbb{R}$, $h(n) = \alpha f(n) + \beta g(n)$ is as well.

Proof.

$$\begin{aligned}
\alpha f(n) + \beta g(n) &= \alpha \sum_{i=1}^{d} a_i f(n-i) + \beta \sum_{i=1}^{d} a_i g(n-i) \\
&= \sum_{i=1}^{d} a_i(\alpha f(n-i) + \beta g(n-i)) \\
&= \sum_{i=1}^{d} a_i h(n-i)
\end{aligned}$$

□

Fibonacci

Now we can combine our two solutions:

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

And we know $f(0) = f(1) = 1$.

Fibonacci

$$f(0) = \alpha \left(\frac{1+\sqrt{5}}{2}\right)^0 + \beta \left(\frac{1-\sqrt{5}}{2}\right)^0 = 1$$

$$\Rightarrow \alpha + \beta = 1$$

$$f(1) = \alpha \left(\frac{1+\sqrt{5}}{2}\right)^1 + \beta \left(\frac{1-\sqrt{5}}{2}\right)^1 = 1$$

$$\Rightarrow \alpha \left(\frac{1+\sqrt{5}}{2}\right) + \beta \left(\frac{1-\sqrt{5}}{2}\right) = 1$$

Solving these equations gives:

$$\alpha = \frac{1}{\sqrt{5}} \cdot \frac{1+\sqrt{5}}{2} \quad \text{and} \quad \beta = -\frac{1}{\sqrt{5}} \cdot \frac{1-\sqrt{5}}{2}$$

So

$$\begin{aligned} f(n) &= \frac{1}{\sqrt{5}} \cdot \frac{1+\sqrt{5}}{2} \left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}} \cdot \frac{1-\sqrt{5}}{2} \left(\frac{1-\sqrt{5}}{2}\right)^n \\ &= \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2}\right)^{n+1} - \frac{1}{\sqrt{5}} \left(\frac{1-\sqrt{5}}{2}\right)^{n+1} \end{aligned}$$

(Binet's formula)

Solving homogeneous linear recurrences

Can we use the same approach on an arbitrary homogeneous linear recurrence?

$$f(n) = a_1 f(n-1) + a_2 f(n-2) + \cdots + a_d f(n-d)$$

Solving homogeneous linear recurrences

Let $f(n) = x^n$:

$$f(n) = a_1 f(n-1) + a_2 f(n-2) + \cdots + a_d f(n-d)$$
$$x^n = a_1 x^{n-1} + a_2 x^{n-2} + \cdots + a_d x^{n-d}$$

Divide by x^{n-d} to get the characteristic equation:

$$x^d = a_1 x^{d-1} + a_2 x^{d-2} + \cdots + a_{d-1} x + a_d$$

Solving homogeneous linear recurrences

For each root r:

- if r is nonrepeated, r^n is a solution
- if r is repeated k times, r^n, nr^n, ..., $n^{k-1}r^n$ are solutions

Then every linear combination of solutions is a solution.

Solving homogeneous linear recurrences

For example, with roots s, t, u, u (again):

Solutions: s^n, t^n, u^n, nu^n

Linear combination: $a \cdot s^n + b \cdot t^n + c \cdot u^n + d \cdot nu^n$

Solving homogeneous linear recurrences

Linear combination: $a \cdot s^n + b \cdot t^n + c \cdot u^n + d \cdot nu^n$

Given boundary conditions:

n	$f(n)$
0	0
1	1
2	4
3	9

Solving homogeneous linear recurrences

Linear combination: $a \cdot s^n + b \cdot t^n + c \cdot u^n + d \cdot nu^n$

$$
\begin{aligned}
a \cdot s^0 + b \cdot t^0 + c \cdot u^0 + d \cdot 0u^0 &= 0 \\
a \cdot s^1 + b \cdot t^1 + c \cdot u^1 + d \cdot 1u^1 &= 1 \\
a \cdot s^2 + b \cdot t^2 + c \cdot u^2 + d \cdot 2u^2 &= 4 \\
a \cdot s^3 + b \cdot t^3 + c \cdot u^3 + d \cdot 3u^3 &= 9
\end{aligned}
$$

Then solve for a, b, c, d.

Solving non-homogeneous linear recurrences

Consider the Towers of Hanoi recurrence:

$$f(n) = 2f(n-1) + 1$$

The extra $+1$ makes this non-homogeneous.

If we drop the non-homogeneous part:

$$f(n) = 2f(n-1)$$

We have a homogeneous recurrence of order 1, and characteristic equation

$$x = 2$$

Hence $f(n) = c2^n$ is a solution.

Solving non-homogeneous linear recurrences

Let's add back the $+1$ and guess that $f(n) = an + b$:

$$\begin{aligned} f(n) &= 2f(n-1) + 1 \\ an + b &= 2(a(n-1) + b) + 1 \\ &= 2an - 2a + 2b + 1 \\ 0 &= an - 2a + b + 1 \\ &= an + (b - 2a + 1) \end{aligned}$$

$$0 = an + (b - 2a + 1)$$

This holds if

$$a = 0 \quad \text{and} \quad b - 2a + 1 = 0$$

or

$$b = -1$$

So $f(n) = an + b = -1$ is a particular solution

Then we add the homogeneous solution and particular solution:

$$f(n) = c2^n - 1$$

Solving non-homogeneous linear recurrences

Finally, use the boundary condition of $f(1) = 1$:

$$\begin{aligned} c2^1 - 1 &= 1 \\ c &= 1 \end{aligned}$$

So

$$f(n) = 2^n - 1$$

Divide and conquer recurrences

Recall merge sort:

$$T(n) = \begin{cases} 0 & n = 1 \\ 2T\left(\frac{n}{2}\right) + n - 1 & n \geq 2 \end{cases}$$

This is not a linear recurrence!

Divide and conquer recurrences

In general:

$$T(n) = \sum_{i=1}^{k} a_i T(b_i n) + g(n)$$

where

- $a_i > 0$
- $0 \leq b_i \leq 1$
- $g(n) \geq 0$

Divide and conquer recurrences

For merge sort:

$$T(n) = a_1 T(b_1 n) + g(n)$$

where

- $a_1 = 2$
- $b_1 = 1/2$
- $g(n) = n - 1$

Akra-Bazzi theorem

Then

$$T(n) = \Theta\left(n^p\left(1 + \int_1^n \frac{g(u)}{u^{p+1}}du\right)\right)$$

where

$$\sum_{i=1}^{k} a_i b_i^p = 1$$

Akra-Bazzi theorem

For merge sort:

$$
\begin{aligned}
2 \cdot (1/2)^p &= 1 \\
p &= 1
\end{aligned}
$$

So

$$
\begin{aligned}
T(n) &= \Theta\left(n^p\left(1+\int_1^n \frac{g(u)}{u^{p+1}}du\right)\right) \\
&= \Theta\left(n\left(1+\int_1^n \frac{u-1}{u^2}du\right)\right)
\end{aligned}
$$

Akra-Bazzi theorem

$$\begin{aligned}
T(n) &= \Theta\left(n\left(1+\int_1^n \frac{u-1}{u^2}du\right)\right) \\
&= \Theta\left(n\left(1+\left[\frac{1}{u}+\log u\right]_1^n\right)\right) \\
&= \Theta\left(n\left(1+\left(\frac{1}{n}+\log n-1-\log 1\right)\right)\right) \\
&= \Theta\left(n\left(\frac{1}{n}+\log n\right)\right) \\
&= \Theta\left(1+n\log n\right) \\
&= \Theta\left(n\log n\right)
\end{aligned}$$

Master theorem

Divide and conquer solves a problem of size n by:

- splitting it into a subproblems of size n/b
- combining the answers in $O(n^d)$ time

where $a, b, d > 0$

Master theorem

If $T(n) = aT(n/b) + O(n^d)$ and $a > 0, b > 1, d \geq 0$, then

$$T(n) = \begin{cases} O(n^{\log_b a}) & \text{if } d < \log_b a \\ O(n^d \log n) & \text{if } d = \log_b a \\ O(n^d) & \text{if } d > \log_b a \end{cases}$$