SC2001-Analysis-Techniques

Algorithm Design and Analysis (Nanyang Technological University)

8. Analysis Techniques

▼ Review of Big Oh, Big Omega, Big Theta

Big Oh: Upper Bound, Growing at same or slower rate as g(n)

- Worst Case

Big Theta: Tight bound, Neither faster nor slower than g(n), Growing at same rate as g(n)

- Using Big Theta is more accurate (provides more info)
- But it is hard to prove!

Big Omega: Lower Bound, Growing faster rate than g(n)

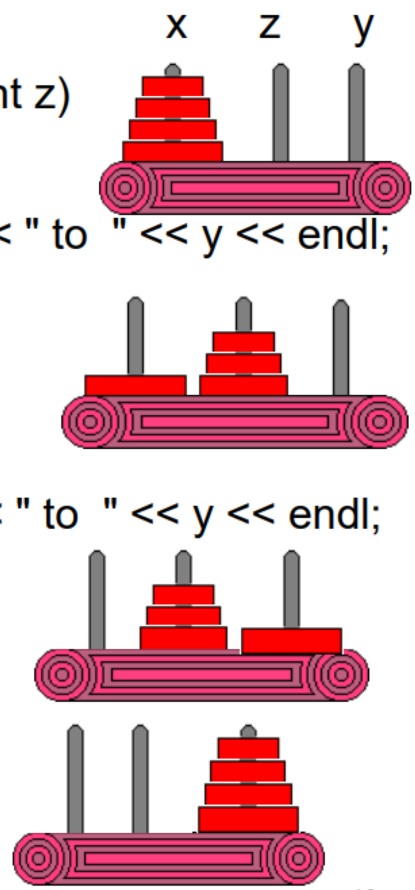
- Best Case

▼ Recursive Algorithms

Many problems have recursive solutions. To analyze such solutions, algorithms will involve a recurrence relation that needs to be solved.

Example 1: Towers of Hanoi

Move all disks from first pole to third pole. Only one disk can be moved at a time and no disk is ever placed on top of a smaller one.




```
void TowersOfHanoi(int n, int x, int y, int z)
{ // Let M(n) be the total no. of disk moves
  if (n == 1)
     cout << "Move  disk from  " << x << " to  " << y << endl;
     // this has one disk move
  else {
    TowersOfHanoi(n-1, x, z, y);
    // this involves M(n-1) disk moves

    cout << "Move  disk from  " << x << " to  " << y << endl;
   // one disk move


    TowersOfHanoi(n-1, z, y, x);
    // another M(n-1) disk moves
  }
}
```

Analysis Techniques SC2001/CX2101 13

Let $M(n)$ = total no. of disk moves

- Base Case: $M(1) = 1$
- $M(n) = M(n-1) + 1 + M(n-1) = 2M(n-1) + 1$

Example 2: Merge sort

```
void mergesort(int l, int m)
{
    int mid = (l+m)/2;
    if (m-l > 1) {
        mergesort(l, mid);
        mergesort(mid+1, m);
    }
    merge(l, m);
}
```

Let $M(n)$ = total no. of comparisons between array elements. n is a power of 2.

- Base Case: $M(2) = 1$
- $M(n) = 2M(\frac{n}{2}) + n - 1$

Solving Recurrences

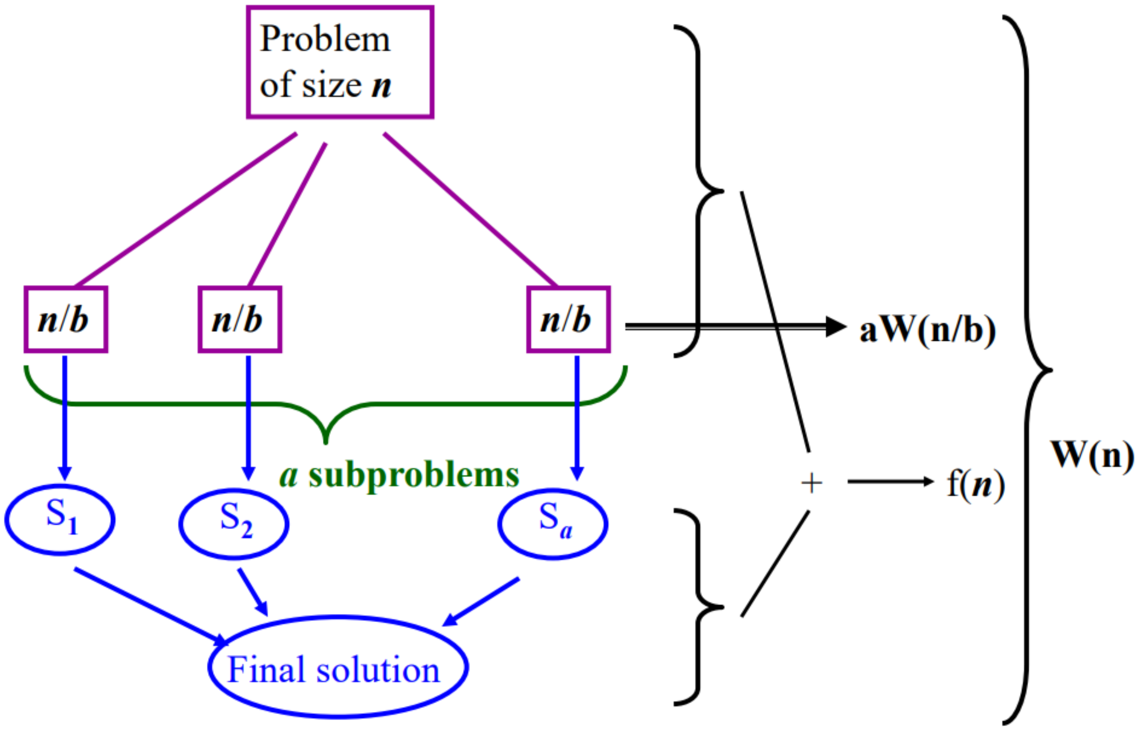
Given the recurrence form

$$W(n) = a \cdot W(\frac{n}{b}) + f(n)$$

where $a \geq 1$ and $b > 1$ are constants, $f(n)$ is a function of n.

- Recurrence describes computational cost of a "divide-and-conquer" algorithm
- $f(n)$: cost of dividing problem and combining results of subproblems
- Usually a problem of size n is divided into subproblems of sizes $\frac{n}{b}$
    - This does not change asymptotic behaviour of recurrence

Visualization of Recurrence Form



Examples of Recurrences

W(n) = 2W(n/2) + 2	Finding the max and min from a sequence
W(n) = W(n/2) + 2	Binary search
W(n) = 3W(n/2) + cn	Multiplying two 2n-bits integers
W(n) = 2W(n/2) + n - 1	Merge sort
W(n) = 7W(n/2) + 15n²/4	Multiplying two nxn matrices

There are 3 main methods used to solve recurrences

Substitution Method

- A "guess and check" kind of strategy.
- Guess form of solution → use Mathematical Induction to prove
- Powerful method (Easier to prove that a certain bound is valid than to compute the bound)
- Only useful when easy to guess form of solution

Mathematical Induction

If $p(a)$ is true and for some integer $k \geq a$, $p(k+1)$ is true whenever $p(k)$ is true, then $p(n)$ is true for all $n \geq a$.

▼ **An example on Worst Case of Merge Sort, where $n = 2^k$**

$W(2) = 1$

$W(n) = 2W(\frac{n}{2}) + n - 1$

Guess $W(n) = O(f(n))$

First Guess: $W(n) = O(n^2)$

Proof by mathematical induction that $W(n) \leq cn^2$:

(1) Base case: $W(2) = 1 \leq 2^2$;

(2) Inductive step: assume that $W(n) = O(n^2)$ for $n \leq 2^k$.
Now consider $n = 2^{k+1}$

$$
\begin{aligned}
W(2^{k+1}) &= 2W(2^k) + 2^{k+1} - 1 \\
&\leq 2 * (2^k)^2 + 2^{k+1} - 1 \\
&= 2 * (2^k)^2 + 2 * 2^k - 1 \\
&\leq 4 * (2^k)^2 \\
&= (2^{k+1})^2
\end{aligned}
$$

i.e. $W(2^{k+1}) \leq (2^{k+1})^2$

A lot is added from step 3 to step 4

Thus $W(n) = O(n^2)$. But is this the best guess?

Second Guess: $W(n) = O(n)$, i.e. $W(n) \leq c \times n$

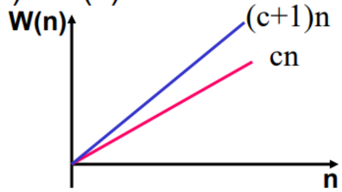
Proof by mathematical induction:

(1) Base case: $W(2) = 1 \le 2c$;

(2) Inductive step: assume that $W(n) = O(n)$ for $n \le 2^k$.

Now consider $n = 2^{k+1}$

$$W(2^{k+1}) = 2W(2^k) + 2^{k+1} - 1$$
$$\le 2 * c * 2^k + 2^{k+1} - 1$$
$$= c * 2^{k+1} + 2^{k+1} - 1$$



Thus $W(2^{k+1}) \le (c+1) * 2^{k+1} - 1$ but we cannot say

$W(2^{k+1}) \le c * 2^{k+1}$ (note: $2^{k+1} - 1 > 0$ for all $k \ge 0$)

Thus $W(n) \ne O(n)$.

Third Guess: $W(n) = O(n \lg n)$

Proof by mathematical induction:

(1) Base case: $W(2) = 1 \le 2\lg 2$;

(2) Inductive step: assume that $W(n) \le n\lg n$ for $n \le 2^k$.

Now consider $n = 2^{k+1}$

$$W(2^{k+1}) = 2W(2^k) + 2^{k+1} - 1$$
$$\le 2 * k * 2^k + 2^{k+1} - 1$$
$$= k * 2^{k+1} + 2^{k+1} - 1$$
$$\le (k + 1) * 2^{k+1}$$

Thus $W(n) = O(n\lg n)$ is a very close upper bound.

What if the base condition does not hold?

Consider the recurrence ($n = 2^k$) :

$W(1) = 1$

$W(n) = 2\,W(n/2) + n - 1$

Prove that $W(n) = O(n\lg n)$:

(1) Base case: $W(1) = 1 > c\lg 1$;

(2) Recall the big-O notation: for $f(n) = O(g(n))$, we need $f(n) <= c * g(n)$ for all $n > n_0$.

(3) Thus to prove $W(n) = O(n\lg n)$, we may use another base case.

- We have $W(2) = 3 < c*2*\lg 2$ for any $c > 1$.
- We can assume that $W(n) \le cn\lg n$ for $n \le 2^k$ then prove $W(2^{k+1}) \le c*(k + 1) * 2^{k+1}$

Then $W(n) = O(n\lg n)$.

What can we say about the general case of n?

The worst case for merge sort :

$W(2) = 1$

$W(n) = W(\lceil n/2 \rceil) + W(\lfloor n/2 \rfloor) + n - 1$

Proof $^+$:

(1) $W(n)$ is a monotonically increasing function. So when n is not a power of 2, that is, $2^k < n < 2^{k+1}$, then $W(2^k) \le W(n) \le W(2^{k+1})$.

(2) We have proved that $W(n) = O(n\lg n)$ for powers of 2, so, $W(2^{k+1}) \le c * (k+1) * 2^{k+1}$.

(3) For any $n < 2^{k+1}$ for some k, $W(n) \le W(2^{k+1})$. Therefore

$W(n) \le c * (k+1) * 2^{k+1} < c * \lg(2n) * (2*n) < 4cn\lg n$

$\boxed{2^k < n, \text{ so } 2^{k+1} < 2n \text{ and } k+1 < \lg(2n)}$

Therefore $W(n) = O(n\lg n)$.

Iteration Method

- Idea is to expand the recurrence by iterating through
- Express it as a summation of terms depending only on n and the initial condition
- Techniques for evaluating summations can be used ot provide bounds on solution
- Leads to lots of algebra

- Focus on how many times the recurrence needs to be iterated to reach boundary condition

▼ **Example on Iteration Method**

W(1) = 1, W(2) = 1, W(3) = 1,

$$W(n) = 3W(\lfloor \frac{n}{4} \rfloor) + n$$

we expand (iterate) it:

$$W(n) = 3W(\lfloor \frac{n}{4} \rfloor) + n$$

$$= 3(3W(\lfloor \frac{n}{4^2} \rfloor) + \lfloor \frac{n}{4} \rfloor) + n$$

$$= 3^2 W(\lfloor \frac{n}{4^2} \rfloor) + 3\lfloor \frac{n}{4} \rfloor + n$$

$$= 3^2(3W(\lfloor \frac{n}{4^3} \rfloor) + \lfloor \frac{n}{4^2} \rfloor) + 3\lfloor \frac{n}{4} \rfloor + n$$

$$= 3^3 W(\lfloor \frac{n}{4^3} \rfloor) + 3^2\lfloor \frac{n}{4^2} \rfloor + 3\lfloor \frac{n}{4} \rfloor + n$$

we need to iterate until we reach one of the boundary conditions, i.e $\lfloor \frac{n}{4^i} \rfloor = 1, 2$ or 3.

E.g. n=64, $4^3 \leq 64 < 4^4$ and $\lfloor \frac{64}{4^3} \rfloor = 1$;

n=255, $4^3 \leq 255 < 4^4$ and $\lfloor \frac{255}{4^3} \rfloor = 3$;

This means if $4^i \leq n < 4^{i+1}$ then $i = \lfloor \log_4 n \rfloor$. So

$$W(n) = 3^i W(a) + 3^{i-1}\lfloor \frac{n}{4^{i-1}} \rfloor + \ldots + 3^2\lfloor \frac{n}{4^2} \rfloor + 3\lfloor \frac{n}{4} \rfloor + n$$

a = 1,2 or 3

$$W(n) = 3^i W(a) + 3^{i-1} \left\lfloor \frac{n}{4^{i-1}} \right\rfloor + \ldots + 3^2 \left\lfloor \frac{n}{4^2} \right\rfloor + 3 \left\lfloor \frac{n}{4} \right\rfloor + n$$

$$\leq 3^{\log_4 n} W(a) + 3^{i-1} \frac{n}{4^{i-1}} + \ldots + 3^2 \frac{n}{4^2} + 3 \frac{n}{4} + n$$

Let $x = 3^{\log_4 n}$ then

$\log_4 x = \log_4 n \log_4 3$ then

$4^{\log_4 x} = 4^{\log_4 n \log_4 3}$ then

$x = n^{\log_4 3}$, i.e. $3^{\log_4 n} = n^{\log_4 3}$

$$\sum_{i=0}^{\infty} \left(\frac{3}{4}\right)^i = 4$$

$$W(n) \leq n^{\log_4 3} + n \sum_{i=0}^{\infty} \left(\frac{3}{4}\right)^i + = O(n)$$

Master Method

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

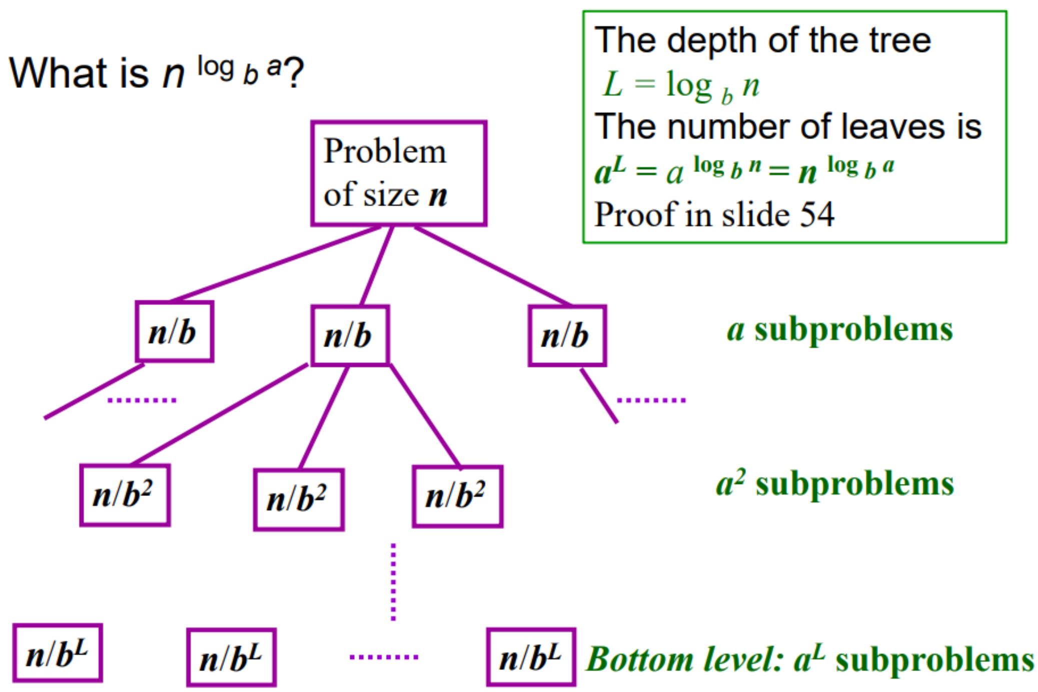
where $a \geq 1$, $b > 1$ and $f(n) = \theta(n^k \log^p n), k \geq 0$

case 1: $\log_b a > k$	case 2: $\log_b a = k$	case 3: $\log_b a < k$
$T(n) = \theta(n^{\log_b a})$	(i) If p > -1 $T(n) = \theta(n^k \log^{p+1} n)$ (ii) If p = -1 $T(n) = \theta(n^k \log(\log n))$ (iii) If p < -1 $T(n) = \theta(n^k)$	(i) If p >= 0 $T(n) = \theta(n^k \log^p n)$ (ii) If p < 0 $T(n) = \theta(n^k)$

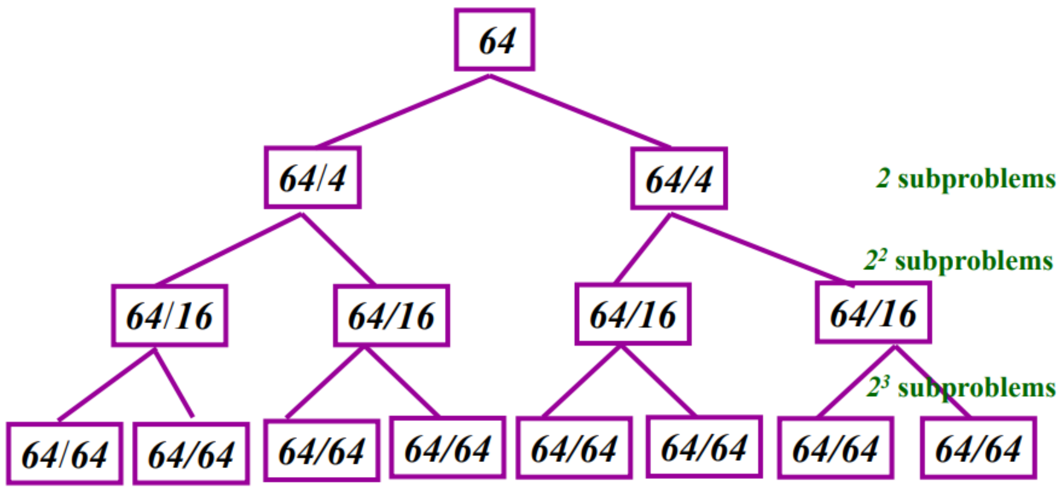
- Provides a "manual" for solving recurrences of the form $W(n) = a \cdot W(\frac{n}{b}) + f(n)$ where $a \geq 1$ and $b > 1$ are constants
- We can determine the asymptotic tight bound in the following three cases

- If $f(n) = O(n^{log_b \cdot a - \epsilon})$ for some constant $\epsilon > 0$, then $W(n) = \Theta(n^{log_b \cdot a})$
- If $f(n) = O(n^{log_b \cdot a}$, then $W(n) = \Theta(n^{log_b \cdot a} \log n)$
- If $f(n) = \Omega(n^{log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $a \cdot f(\frac{n}{b}) \leq c \cdot f(n)$ for some constant $c < 1$ and all sufficiently large n, then $W(n) = \Theta(f(n))$

What is $n^{\log_b a}$



E.g. $\boldsymbol{n} = 64$, $\boldsymbol{a} = 2$, $\boldsymbol{b} = 4$

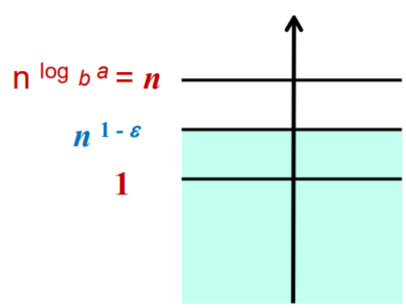


Depth of tree L = $\log_4 64$, Number of leaves = 8 = $2^{\log_4 64} = 64^{\log_4 2}$

$(a^{\log_b n} = n^{\log_b a})$

▼ Examples on Master Method

Examples



Complexity

1) $W(n) = 3W(n/3) + 2$,

so $a = 3$, $b = 3$,

$n^{\log_b a} = n^1$

$f(n) = 2 = \theta(1) = O(n^1)$

We may let $\varepsilon = 0.5$ then we confirm $2 = O(n^{1-0.5})$,

i.e. $f(n) = O(n^{1-\varepsilon})$

$\Rightarrow f(n) = O(n^{\log_b a - \varepsilon})$ (case 1)

thus $W(n) = \theta(n^{\log_b a})$

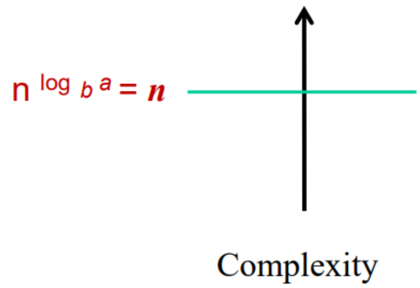
$W(n) = \theta(n)$.

2) $W(n) = 4W(n/4) + n - 1$,

so $a = 4$, $b = 4$,

$n^{\log_b a} = n^1$

$f(n) = n - 1$



We have

$f(n) = n-1$

$= \theta(n^1)$,

$= \theta(n^{\log_b a})$, (case 2)

thus

$W(n) = \theta(n^{\log_b a} \log n)$

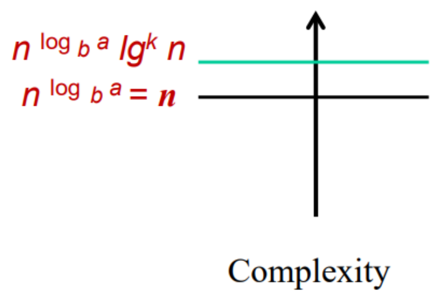
$= \theta(n \log n)$

3) $W(n) = 2W(n/2) + n \lg n$,

so $a = 2$, $b = 2$,

$f(n) = n \lg n$

$n^{\log_b a} = n^1$



We have

$f(n) = \theta(n^1 \lg n)$,

$= \theta(n^{\log_b a} \lg^k n)$, (case 2: $k = 1$)

thus

$W(n) = \theta(n^{\log_b a} \lg^2 n)$

$= \theta(n (\lg n)^2)$

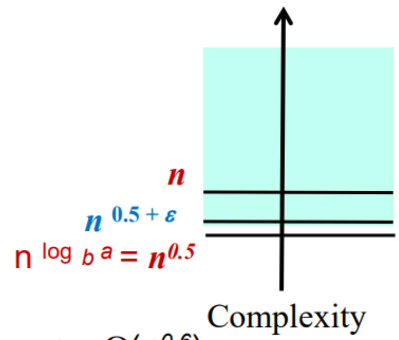
Examples

4) $W(n) = 2W(n/4) + n$,

so $a = 2$, $b = 4$,

$n^{\log_b a} = n^{\log_4 2} = n^{0.5}$

$f(n) = n = \theta(n)$



We may let $\varepsilon = 0.1$ then we have $n = \Omega(n^{0.6})$

i.e. $f(n) = \Omega(n^{\log_b a + \varepsilon})$, and

for all sufficiently large n, we can find a value for c, say, $c = ¾$, to show that $a\,f(n/b) \le c\,f(n)$. (case 3)

$a*f(n/b) = 2*f(n/4) = n/2 \le c*n$

thus $W(n) = \theta(n)$.

There are times where Master Method cannot apply

Example 2: $W(n) = W(n/3) + f(n)$

where $f(n) = \begin{cases} 3n + 2^{3n} & for\ n = 2^i \\ 3n & otherwise \end{cases}$

so $a = 1$, $b = 3$ then $n^{\log_b a} = n^0$

let $\varepsilon = 1$ then $f(n) = \Omega(n^{0+1})$, case 3?

$a\,f(n/b) \le c\,f(n)$ for all sufficiently large n?

When $n = 3 * 2^i$, $a\,f(n/b) = f(2^i) = n + 2^n$, but $cf(n) = c(3n)$

i.e. $a\,f(n/b) > c\,f(n)$. E.g. for n = 6 or greater

So the Master Theorem cannot apply.

Notice that when we want to find the order of a recurrence, the initial conditions are not important. This is because the running costs of the terminating conditions are small constants that do not affect the order.

▼ Iteration Method

1. Unfold/Expand the Recurrence Repeatedly
2. Look for a pattern
3. Generalize the pattern

Example

https://www.youtube.com/watch?v=Ob8SM0fz6p0

▼ Recurrence Relations

Definition. A linear homogeneous recurrence relation of degree k with constant coefficients is a recurrent relation of the form

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k}$$

where $c_1, c_2, \cdots, c_k$ are real constants and $c_k \neq 0$.

Linear: $a_{n-1}, a_{n-2}, \cdots, a_{n-k}$ appear in separate terms and to the first power

Homogeneous: total degree of each term is the same, e.g. no constant term

Constant coefficients: $c_1, c_2, \cdots, c_k$ are fixed real constants that do not depend on n

Degree k: the expression for a_n contains the previous k terms

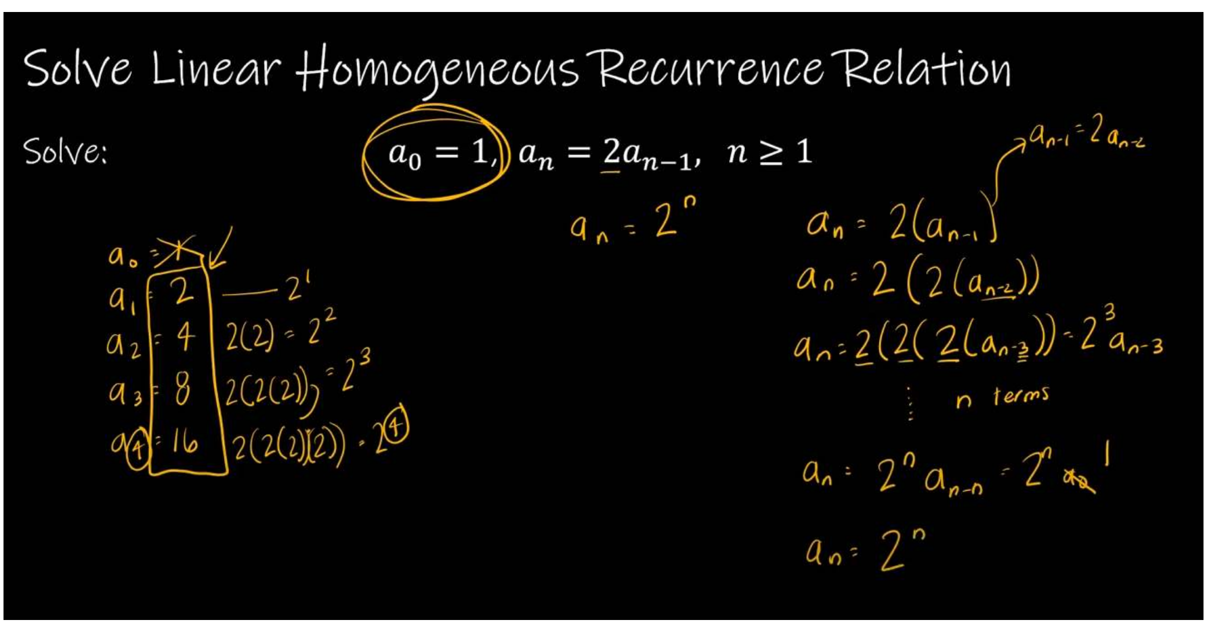
$a_{n-1}, a_{n-2}, \cdots, a_{n-k}, (c_k \neq 0)$

- Examples
  - A *linear homogeneous recurrence relation of degree 2:* $a_n = a_{n-1} + a_{n-2}$
  - A *linear homogeneous recurrence relation of degree 1:* $a_n = 1.04a_{n-1}$
  - A *linear homogeneous recurrence relation of degree 3 :* $a_n = a_{n-3}$
- Non-examples
  - $a_n = a_{n-1} + a_{n-2} + 1$: non-homogeneous
  - $a_n = a_{n-1}a_{n-2}$: not linear
  - $a_n = na_{n-1}$: coefficient not constant

Solving Linear Homogeneous Recurrence Relation

Example: $a_0 = 1, a_n = 2a_{n-1}, n \geq 1$

Geometric Progression Method



Solving Second-Order Linear Homogenous Recurrence Relations

- Write characteristic polynomial (Shift everything to the left, set equal to 0)
- Factor polynomial and find roots
- Determine the form of a_n
    - Case 1: Real and distinct roots $(r_1 \neq r_2) : a_n = A(r_1)^n + B(r_2)^n$
    - Case 2: Double (or triple, etc.) root $(r_1 = r_2) : a_n = A(r_1)^n + Bn(r_2)^n$
    - Case 3: Complex roots $(r_1 \neq r_2)$
- Solve for coefficients

Example: $a_n = 5a_{n-1} - 6a_{n-2}, a_0 = 1, a_1 = 1$

Solve the homogeneous recurrence relation with constant coefficients

1. Write characteristic polynomial (Shift everything to the left, set equal to 0)

   $$a_n - 5a_{n-1} + 6a_{n-2} = 0$$

   a. Replace all a_n with r^n

   $$r^n - 5r^{n-1} + 6r^{n-2} = 0$$

   b. Divide each term by the LEAST degree of r^n (in this case it is r^{n-2})

   $$r^2 - 5r + 6 = 0$$

2. Factor polynomial and find roots

   $(r-3)(r-2) = 0$

3. Determine the form of a_n

   $r = 2, r = 3$ (Two distinct roots) (Form: $a_n = A(r_1)^n + B(r_2)^n$)

   $$a_n = A(2)^n + B(3)^n$$

4. Solve of coefficients (Where $a_0 = 1, a_1 = 1$)

   $n = 0, a_0 = A2^0 + B3^0, 1 = A + B$

   $n = 1, a_1 = A2^1 + B3^1, 1 = 2A + 3B$

   Solve using substitution: $A = 2, B = -1$

5. Sub A and B into $a_n = A(2)^n + B(3)^n$

   $a_n = 2^{n+1} - 3^n$

Solving Second-Order Linear Homogenous Recurrence Relations

- Rewrite it so that it equals 0 (shift everything to the left)
- Replace any initial terms with r^m
- Divide by the least degree of r^m

$$\frac{r^n}{r^{n-2}} = r^{n-(n-2)} = r^2$$

$$a_n - 5a_{n-1} + 6a_{n-2} = 0$$

$$r^n - 5r^{n-1} + 6r^{n-2} = 0$$

$$r^{n-2}(r^2 - 5r + 6) = 0$$

$$r^2 - 5r + 6 = 0$$

Second-Order Linear Homogeneous Recurrence Relations

Solve the linear homogeneous recurrence relation with constant coefficients:

$$a_n = 5a_{n-1} - 6a_{n-2}, \textit{where } a_0 = 1, a_1 = 1$$

2. Factor your characteristic polynomial. Then find the roots.

$$r^2 - 5r + 6 = 0$$

$$(r-2)(r-3) = 0$$

$r - 2 = 0$ $\quad$ $r - 3 = 0$

$r = 2$ $\quad$ $r = 3$

3. Determine the form of a_n.

Two real, distinct roots. Form: $\boxed{a_n = A(r_1)^n + B(r_2)^n}$

$$a_n = A\,2^n + B\,3^n$$

Second-Order Linear Homogeneous Recurrence Relations

Solve the linear homogeneous recurrence relation with constant coefficients:

$$a_n = 5a_{n-1} - 6a_{n-2}, \textit{where } \boxed{a_0 = 1}, \boxed{a_1 = 1}$$

4. Write and solve the system of equations using your initial values.

$$a_n = A2^n + B3^n$$

$n = 0$ $\quad$ $a_0 = A \cdot 2^0 + B \cdot 3^0$

$n = 1$ $\quad$ $a_1 = A \cdot 2^1 + B \cdot 3^1$

$1 = A + B$

$1 = 2A + 3B$

(subst.)

$A = 1 - B$

$1 = 2(1 - B) + 3B$

$1 = 2 - 2B + 3B$

$\boxed{-1 = B}$

$1 = A + (-1)$

$\boxed{2 = A}$

$a_n = \boxed{2 \cdot 2^n} + (-1)3^n$

$\boxed{a_n = 2^{n+1} - 3^n}$

General Case Linear Homogeneous Recurrence Relations

We will use the same steps as we did for second-order recurrence relations for any higher-order recurrence relations:

1. Write the characteristic polynomial (polynomial set equal to zero)
2. Factor the polynomial and find the roots.
3. Determine the form of a_n
4. Solve for the coefficients

Characteristic Root Technique for Repeated Roots.

Suppose the recurrence relation $a_n = \alpha a_{n-1} + \beta a_{n-2}$ has a characteristic polynomial with only one root r. Then the solution to the recurrence relation is

$$a_n = ar^n + bnr^n$$

where a and b are constants determined by the initial conditions.

characteristic roots:

Characteristic Roots.

Given a recurrence relation $a_n + \alpha a_{n-1} + \beta a_{n-2} = 0$, the ***characteristic polynomial*** is

$$x^2 + \alpha x + \beta$$

giving the ***characteristic equation***:

$$x^2 + \alpha x + \beta = 0.$$

If r_1 and r_2 are two distinct roots of the characteristic polynomial (i.e, solutions to the characteristic equation), then the solution to the recurrence relation is

$$a_n = ar_1^n + br_2^n,$$

where a and b are constants determined by the initial conditions.