Counting Steps and Recurrence Relations

Stefan D. Bruda

CS 317, Fall 2024

ANALYZING NON-RECURSIVE ALGORITHMS

- To find the running time all we have to do is count steps, carefully
- Examples:
  - **for** $i = 1$ TO n **do**
    ```
    j ← 1
    while j ≤ i do
        ...
        j ← j + 1
    ```
  - **for** $i = 1$ TO n **do**
    ```
    j ← n
    while j > 1 do
        ...
        j ← j/2
    ```
  - **algorithm** BINSEARCH(x, S, l, h)**:**
    ```
    i ← l
    j ← h
    while i ≤ j do
        m ← (i + j)/2
        if S_m = x then return m
        else if S_m > x then j ← m − 1
        else i ← m + 1
    return −1
    ```

- To find the running time all we have to do is count steps, carefully
- Examples:
  - **for** $i = 1$ TO n **do** $\qquad O(n^2)$
    ```
    for i = 1 TO n do
        j ← 1
        while j ≤ i do
            ...
            j ← j + 1
    ```
  - **for** $i = 1$ TO n **do**
    ```
    for i = 1 TO n do
        j ← n
        while j > 1 do
            ...
            j ← j/2
    ```
  - **algorithm** BINSEARCH(x, S, l, h)**:**
    ```
    algorithm BinSearch(x, S, l, h):
        i ← l
        j ← h
        while i ≤ j do
            m ← (i + j)/2
            if S_m = x then return m
            else if S_m > x then j ← m − 1
            else i ← m + 1
        return −1
    ```

Analyzing non-recursive algorithms

- To find the running time all we have to do is count steps, carefully
- Examples:
  - **for** $i = 1$ TO n **do** — $O(n^2)$

```
for i = 1 TO n do
    j ← 1
    while j ≤ i do
        ...
        j ← j + 1
```

  - **for** $i = 1$ TO n **do** — $O(n \log n)$

```
for i = 1 TO n do
    j ← n
    while j > 1 do
        ...
        j ← j/2
```

  - **algorithm** BINSEARCH(x, S, l, h)**:**

```
algorithm BinSearch(x, S, l, h):
    i ← l
    j ← h
    while i ≤ j do
        m ← (i + j)/2
        if S_m = x then return m
        else if S_m > x then j ← m − 1
        else i ← m + 1
    return −1
```

ANALYZING NON-RECURSIVE ALGORITHMS

- To find the running time all we have to do is count steps, carefully
- Examples:
  - $O(n^2)$
    ```
    for i = 1 TO n do
        j ← 1
        while j ≤ i do
            ...
            j ← j + 1
    ```
  - $O(n \log n)$
    ```
    for i = 1 TO n do
        j ← n
        while j > 1 do
            ...
            j ← j/2
    ```
  - $O(\log n)$
    ```
    algorithm BinSearch(x, S, l, h):
        i ← l
        j ← h
        while i ≤ j do
            m ← (i + j)/2
            if S_m = x then return m
            else if S_m > x then j ← m − 1
            else i ← m + 1
        return −1
    ```

- Counting steps in a recursive algorithm produces a recurrence relation
  - **algorithm** BINSEARCH(x, S, l, h): // $T(n)$

```
if l > h then return -1
else
    m ← (l + h)/2
    if x == S_m then return m
    else if x < S_m then return BinSearch(x, S, l, m−1)   // T(n/2)
    else return BinSearch(x, S, m+1, h)                    // T(n/2)
```

ANALYZING RECURSIVE ALGORITHMS

- Counting steps in a recursive algorithm produces a recurrence relation
  - **algorithm** BINSEARCH(x, S, l, h): // $T(n)$

```
if l > h then return -1
else
    m ← (l + h)/2
    if x == S_m then return m
    else if x < S_m then return BinSearch(x, S, l, m−1)   // T(n/2)
    else return BinSearch(x, S, m+1, h)                   // T(n/2)
```

$$T(n) = \begin{cases} c & n = 1 \\ T(n/2) + c' & n > 1 \end{cases} \qquad \begin{array}{l} T(n) = T(n/2) + 1 \\ T(1) = 1 \end{array}$$

- Counting steps in a recursive algorithm produces a recurrence relation
  - **algorithm** BINSEARCH(x, S, l, h): // $T(n)$

```
if l > h then return -1
else
    m ← (l + h)/2
    if x == S_m then return m
    else if x < S_m then return BINSEARCH(x, S, l, m−1)   // T(n/2)

    else return BINSEARCH(x, S, m+1, h)                   // T(n/2)
```

$$T(n) = \begin{cases} c & n = 1 \\ T(n/2) + c' & n > 1 \end{cases} \qquad T(n) = T(n/2) + 1, \quad T(1) = 1$$

  - **algorithm** MERGESORT(S, l, h): // $T(n)$

```
if l > h then m ← (l + h)/2
  MERGESORT(l, m)          // T(n/2)
MERGESORT(m+1, h)          // T(n/2)
MERGE(l, m, h)             // O(n)
```

- Counting steps in a recursive algorithm produces a recurrence relation
  - **algorithm** BINSEARCH(x, S, l, h): // $T(n)$

```
if l > h then return -1
else
    m ← (l + h)/2
    if x == S_m then return m
    else if x < S_m then return BINSEARCH(x, S, l, m−1)   // T(n/2)
    else return BINSEARCH(x, S, m+1, h)                   // T(n/2)
```

$$T(n) = \begin{cases} c & n = 1 \\ T(n/2) + c' & n > 1 \end{cases}$$

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

  - **algorithm** MERGESORT(S, l, h): // $T(n)$

```
if l > h then m ← (l + h)/2
    MERGESORT(l, m)        // T(n/2)
    MERGESORT(m+1, h)      // T(n/2)
    MERGE(l, m, h)         // O(n)
```

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

- Counting steps in a recursive algorithm produces a recurrence relation
  - **algorithm** BINSEARCH(x, S, l, h): // $T(n)$

```
if l > h then return -1
else
    m ← (l + h)/2
    if x == S_m then return m
    else if x < S_m then return BINSEARCH(x, S, l, m−1)   // T(n/2)
    else return BINSEARCH(x, S, m + 1, h)                 // T(n/2)
```

$$T(n) = \begin{cases} c & n = 1 \\ T(n/2) + c' & n > 1 \end{cases}$$

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

  - **algorithm** MERGESORT(S, l, h): // $T(n)$

```
if l > h then m ← (l + h)/2
    MERGESORT(l, m)           // T(n/2)
    MERGESORT(m + 1, h)       // T(n/2)
    MERGE(l, m, h)            // O(n)
```

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

  - $T(n) = 2T(n-1) + 1$, $T(1) = 1$ (towers of Hanoi)
  - $T(n) = 5T(n-1) - 6T(n-2) + 1$, $T(0) = 5$, $T(1) = 7$

- Technically all the techniques below produce guesses
  - All guesses must be verified by induction
- Induction
  - Calculate a few values until able to make an educated guess

Solving Recurrence Relations

- Technically all the techniques below produce guesses
  - All guesses must be verified by induction
- Induction
  - Calculate a few values until able to make an educated guess
- Forward substitution
  - Obtain $T(n)$ for a few values without performing the calculations
  - See if a series emerge and guess the general form

Solving Recurrence Relations

- Technically all the techniques below produce guesses
  - All guesses must be verified by induction
- Induction
  - Calculate a few values until able to make an educated guess
- Forward substitution
  - Obtain $T(n)$ for a few values without performing the calculations
  - See if a series emerge and guess the general form
- Backward substitution
  - Expand $T(n)$ repeatedly without performing the calculations
  - See if a series emerge and guess the general form

Solving Recurrence Relations

- Technically all the techniques below produce guesses
  - All guesses must be verified by induction
- Induction
  - Calculate a few values until able to make an educated guess
- Forward substitution
  - Obtain $T(n)$ for a few values without performing the calculations
  - See if a series emerge and guess the general form
- Backward substitution
  - Expand $T(n)$ repeatedly without performing the calculations
  - See if a series emerge and guess the general form
- Summing factors
  - Write down the formulae for n, $n-1$, $n-2$, etc. (or n, $n/2$, $n/4$, $\ldots$) and add them up together
  - Simplify the sum, hopefully reaching a
  - See if a series emerge for $T(n)$ and guess the general form

Characteristic equation

Definition (homogeneous linear recurrence)

A recurrence of the form $a_0 t_n + a_1 t_{n-1} + a_2 t_{n-2} + \cdots + a_k t_{n-k} = 0$ where k and a_i are constants is called a homogeneous linear recurrence equation

Definition (characteristic equation)

The characteristic equation for the homogeneous linear recurrence equation
$a_0 t_n + a_1 t_{n-1} + a_2 t_{n-2} + \cdots + a_k t_{n-k} = 0$ is
$a_0 r^k + a_1 r^{k-1} + a_2 r^{k-2} + \cdots + a_k r^0 = 0$

Theorem (solution of a homogeneous linear equation)

Let $a_0 t_n + a_1 t_{n-1} + a_2 t_{n-2} + \cdots + a_k t_{n-k} = 0$ *be a homogeneous linear recurrence equation. If the characteristic equation of this relation has k distinct solutions* $r_1, r_2, \ldots, r_k$, *then the only solution to the recurrence relation is* $t_n = c_1 r_1^n + c_2 r_2^n + \cdots + c_k r_k^n$

Solving recurrence relations using the characteristic equation

- We obtain r from the characteristic equation and then we can determine the constants c_i from the base case(s)

Solving recurrence relations using the characteristic equation

- We obtain r from the characteristic equation and then we can determine the constants c_i from the base case(s)
- Example: $T(n) = 2T(n-1), T(1) = 1$
  - Rewrite inductive case as a linear recurrence: $t_n - 2t_{n-1} = 0$

Solving recurrence relations using the characteristic equation

- We obtain r from the characteristic equation and then we can determine the constants c_i from the base case(s)
- Example: $T(n) = 2T(n-1), T(1) = 1$
  - Rewrite inductive case as a linear recurrence: $t_n - 2t_{n-1} = 0$
  - Characteristic equation: $r - 2 = 0$
  - Solve equation, obtaining $r = 2$
  - Therefore we have $T(n) = c_1 2^n$

Solving recurrence relations using the characteristic equation

- We obtain r from the characteristic equation and then we can determine the constants c_i from the base case(s)
- Example: $T(n) = 2T(n-1)$, $T(1) = 1$
  - Rewrite inductive case as a linear recurrence: $t_n - 2t_{n-1} = 0$
  - Characteristic equation: $r - 2 = 0$
  - Solve equation, obtaining $r = 2$
  - Therefore we have $T(n) = c_1 2^n$
  - From the base case we have $T(1) = c_1 2^1 = 1$ and thus $c_1 = 1/2$
  - Therefore $T(n) = 2^{n-1} = O(2^n)$

SOLVING RECURRENCE RELATIONS USING THE CHARACTERISTIC EQUATION (CONT'D)

- Note in passing: recall that the solutions of the quadratic equation $ax^2 + bx + c = 0$ are

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- Example: $T(n) = T(n-1) + T(n-2)$, $T(0) = T(1) = 1$ (Fibonacci sequence)

Solving recurrence relations using the characteristic equation (cont'd)

- Note in passing: recall that the solutions of the quadratic equation $ax^2 + bx + c = 0$ are

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- Example: $T(n) = T(n-1) + T(n-2)$, $T(0) = T(1) = 1$ (Fibonacci sequence)
  - Characteristic equation: $r^n - r^{n-1} - r^{n-2} = 0$
  - That is, $r^2 - r - 1 = 0$

Solving recurrence relations using the characteristic equation (cont'd)

- Note in passing: recall that the solutions of the quadratic equation $ax^2 + bx + c = 0$ are

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- Example: $T(n) = T(n-1) + T(n-2)$, $T(0) = T(1) = 1$ (Fibonacci sequence)
  - Characteristic equation: $r^n - r^{n-1} - r^{n-2} = 0$
  - That is, $r^2 - r - 1 = 0$
  - Solve as a quadratic equation: $r_{1,2} = (1 \pm \sqrt{5})/2$
  - From the base cases $c_1 + c_2 = 1$ and $c_1(1 + \sqrt{5})/2 + c_2(1 - \sqrt{5})/2 = 1$
  - Therefore $c_{1,2} = (\sqrt{5} \pm 1)/(2\sqrt{5})$

Solving recurrence relations using the characteristic equation (cont'd)

- Note in passing: recall that the solutions of the quadratic equation $ax^2 + bx + c = 0$ are

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- Example: $T(n) = T(n-1) + T(n-2)$, $T(0) = T(1) = 1$ (Fibonacci sequence)
  - Characteristic equation: $r^n - r^{n-1} - r^{n-2} = 0$
  - That is, $r^2 - r - 1 = 0$
  - Solve as a quadratic equation: $r_{1,2} = (1 \pm \sqrt{5})/2$
  - From the base cases $c_1 + c_2 = 1$ and $c_1(1+\sqrt{5})/2 + c_2(1-\sqrt{5})/2 = 1$
  - Therefore $c_{1,2} = (\sqrt{5} \pm 1)/(2\sqrt{5})$
  - That is:

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

  - Not terribly illuminating...

CHARACTERISTIC EQUATION WITH MULTIPLICITY

- A solution of an equation is said to have multiplicity m if it appears m times in the list of solutions to that equation
  - For example the equation $(r-2)(r-5)^3 = 0$ has the solutions $r_1 = 2$ with multiplicity 1 and $r_2 = 5$ with multiplicity 3

Characteristic equation with multiplicity

- A solution of an equation is said to have multiplicity m if it appears m times in the list of solutions to that equation
  - For example the equation $(r-2)(r-5)^3 = 0$ has the solutions $r_1 = 2$ with multiplicity 1 and $r_2 = 5$ with multiplicity 3

Theorem (homogeneous linear recurrence with multiplicity)

Let r be a root of multiplicity m of the characteristic equation for a homogeneous linear recurrence. Then $t_n = n^k r^n$, $0 \leq k < m$ are all solutions to the recurrence and so much be included in the general solution.

- That is, a solution r with multiplicity k will contribute the following to t_n: $c_0 n^0 r^n + c_1 n^1 r^n + \cdots + c_{m-1} n^{m-1} r^n$
- Example: $t_n - 7t_{n-1} + 15t_{n-2} - 9t_{n-3} = 0$, $t_0 = 0$, $t_1 = 1$, $t_2 = 2$

- A solution of an equation is said to have multiplicity m if it appears m times in the list of solutions to that equation
  - For example the equation $(r-2)(r-5)^3 = 0$ has the solutions $r_1 = 2$ with multiplicity 1 and $r_2 = 5$ with multiplicity 3

Theorem (homogeneous linear recurrence with multiplicity)

Let r be a root of multiplicity m of the characteristic equation for a homogeneous linear recurrence. Then $t_n = n^k r^n$, $0 \leq k < m$ are all solutions to the recurrence and so much be included in the general solution.

- That is, a solution r with multiplicity k will contribute the following to t_n: $c_0 n^0 r^n + c_1 n^1 r^n + \cdots + c_{m-1} n^{m-1} r^n$
- Example: $t_n - 7t_{n-1} + 15t_{n-2} - 9t_{n-3} = 0$, $t_0 = 0$, $t_1 = 1$, $t_2 = 2$
  - Characteristic equation: $r^3 - 7r^2 + 15r - 9 = 0$

- A solution of an equation is said to have multiplicity m if it appears m times in the list of solutions to that equation
  - For example the equation $(r-2)(r-5)^3 = 0$ has the solutions $r_1 = 2$ with multiplicity 1 and $r_2 = 5$ with multiplicity 3

Theorem (homogeneous linear recurrence with multiplicity)

Let r be a root of multiplicity m of the characteristic equation for a homogeneous linear recurrence. Then $t_n = n^k r^n$, $0 \le k < m$ are all solutions to the recurrence and so much be included in the general solution.

- That is, a solution r with multiplicity k will contribute the following to t_n: $c_0 n^0 r^n + c_1 n^1 r^n + \cdots + c_{m-1} n^{m-1} r^n$
- Example: $t_n - 7t_{n-1} + 15t_{n-2} - 9t_{n-3} = 0$, $t_0 = 0$, $t_1 = 1$, $t_2 = 2$
  - Characteristic equation: $r^3 - 7r^2 + 15r - 9 = 0$
  - That is, $(r-1)(r-3)^2 = 0$ and so $r_1 = 1$ (multiplicity 1) and $r_2 = 3$ (multiplicity 2)
  - Therefore the general solution is $t_n = c_1 1^n + c_2 3^n + c_3 n 3^n$

- A solution of an equation is said to have multiplicity m if it appears m times in the list of solutions to that equation
  - For example the equation $(r-2)(r-5)^3 = 0$ has the solutions $r_1 = 2$ with multiplicity 1 and $r_2 = 5$ with multiplicity 3

Theorem (homogeneous linear recurrence with multiplicity)

Let r be a root of multiplicity m of the characteristic equation for a homogeneous linear recurrence. Then $t_n = n^k r^n$, $0 \leq k < m$ are all solutions to the recurrence and so much be included in the general solution.

- That is, a solution r with multiplicity k will contribute the following to t_n: $c_0 n^0 r^n + c_1 n^1 r^n + \cdots + c_{m-1} n^{m-1} r^n$
- Example: $t_n - 7t_{n-1} + 15t_{n-2} - 9t_{n-3} = 0, t_0 = 0, t_1 = 1, t_2 = 2$
  - Characteristic equation: $r^3 - 7r^2 + 15r - 9 = 0$
  - That is, $(r-1)(r-3)^2 = 0$ and so $r_1 = 1$ (multiplicity 1) and $r_2 = 3$ (multiplicity 2)
  - Therefore the general solution is $t_n = c_1 1^n + c_2 3^n + c_3 n 3^n$
  - From the base cases we have $c_1 = -1$, $c_2 = 1$, $c_3 = 1/3$
  - Therefore $t_n = 3^n - n3^{n-1} - 1$

NON-HOMOGENEOUS LINEAR RELATIONS

- General form: $a_0 t_n + a_1 t_{n-1} + a_2 t_{n-2} + \cdots + a_k t_{n-k} = f(n)$

NON-HOMOGENEOUS LINEAR RELATIONS

- General form: $a_0 t_n + a_1 t_{n-1} + a_2 t_{n-2} + \cdots + a_k t_{n-k} = f(n)$
  - No known method to solve them
- Special case: $a_0 t_n + a_1 t_{n-1} + a_2 t_{n-2} + \cdots + a_k t_{n-k} = b^n p(n)$, with b constant and $p(n)$ a polynomial in n
  - Can be transformed into a homogeneous linear recurrence

Non-homogeneous linear relations

- General form: $a_0 t_n + a_1 t_{n-1} + a_2 t_{n-2} + \cdots + a_k t_{n-k} = f(n)$
  - No known method to solve them
- Special case: $a_0 t_n + a_1 t_{n-1} + a_2 t_{n-2} + \cdots + a_k t_{n-k} = b^n p(n)$, with b constant and $p(n)$ a polynomial in n
  - Can be transformed into a homogeneous linear recurrence
  - Example: $t_n - 3t_{n-1} = 4^n$, $t_0 = 0$, $t_1 = 4$
  - Replace n with $n-1$: $t_{n-1} - 3t_{n-2} = 4^{n-1}$
  - Divide the original by 4: $1/4 t_n - 3/4 t_{n-1} = 4^{n-1}$
  - Subtract the second version from the first: $1/4 t_n - 7/4 t_{n-1} + 3t_{n-2} = 0$
  - Homogeneous linear recurrence!

Non-homogeneous linear relations

- General form: $a_0 t_n + a_1 t_{n-1} + a_2 t_{n-2} + \cdots + a_k t_{n-k} = f(n)$
  - No known method to solve them
- Special case: $a_0 t_n + a_1 t_{n-1} + a_2 t_{n-2} + \cdots + a_k t_{n-k} = b^n p(n)$, with b constant and $p(n)$ a polynomial in n
  - Can be transformed into a homogeneous linear recurrence
  - Example: $t_n - 3t_{n-1} = 4^n$, $t_0 = 0$, $t_1 = 4$
  - Replace n with $n-1$: $t_{n-1} - 3t_{n-2} = 4^{n-1}$
  - Divide the original by 4: $1/4t_n - 3/4t_{n-1} = 4^{n-1}$
  - Subtract the second version from the first: $1/4t_n - 7/4t_{n-1} + 3t_{n-2} = 0$
  - Homogeneous linear recurrence!

Theorem (Non-homogeneous transformation)

A non-homogeneous linear recurrence of the form $a_0 t_n + a_1 t_{n-1} + a_2 t_{n-2} + \cdots + a_k t_{n-k} = b^n p(n)$ can be transformed into an equivalent homogeneous linear recurrence with the following characteristic equation: $(a_0 r^k + a_1 r^{k-1} + a_2 r^{k-2} + \cdots + a_k r^0)(r-b)^{d+1} = 0$, where d is the degree of $p(n)$

- Two sets of solutions, one from the homogeneous part and the other from the non-homogeneous part

Domain transformation

- Sometimes we do not have a linear recurrence because the indices are nowhere near each other
- We may be able to bring the indices closer using a change of variable

Domain transformation

- Sometimes we do not have a linear recurrence because the indices are nowhere near each other
- We may be able to bring the indices closer using a change of variable
  - Most common change of variable: from n to 2^k
  - Do not forget to change the variable back when done

- Sometimes we do not have a linear recurrence because the indices are nowhere near each other
- We may be able to bring the indices closer using a change of variable
  - Most common change of variable: from n to 2^k
  - Do not forget to change the variable back when done
  - Example: $T(n) = 2T(n/2) + 1$, $T(1) = 0$
    - Would result in $t_n = t_{n/2} + 1$, not linear

- Sometimes we do not have a linear recurrence because the indices are nowhere near each other
- We may be able to bring the indices closer using a change of variable
  - Most common change of variable: from n to 2^k
  - Do not forget to change the variable back when done
  - Example: $T(n) = 2T(n/2) + 1$, $T(1) = 0$
    - Would result in $t_n = t_{n/2} + 1$, not linear
    - However n and $n/2$ are near each other on a logarithmic scale
    - So we let $n = 2^k$ (and so $k = \log n$) and we have: $T(2^k) = 2T(2^{k-1}) + 1$

DOMAIN TRANSFORMATION

- Sometimes we do not have a linear recurrence because the indices are nowhere near each other
- We may be able to bring the indices closer using a change of variable
  - Most common change of variable: from n to 2^k
  - Do not forget to change the variable back when done
  - Example: $T(n) = 2T(n/2) + 1, T(1) = 0$
    - Would result in $t_n = t_{n/2} + 1$, not linear
    - However n and $n/2$ are near each other on a logarithmic scale
    - So we let $n = 2^k$ (and so $k = \log n$) and we have: $T(2^k) = 2T(2^{k-1}) + 1$
    - With $t_k = T(2^k)$ we have: $t_k = 2t_{k-1} + 1$
    - Note that $t_0 = T(2^0) = 0$ and $t_1 = 2t_0 + 1 = 1$

- Sometimes we do not have a linear recurrence because the indices are nowhere near each other
- We may be able to bring the indices closer using a change of variable
  - Most common change of variable: from n to 2^k
  - Do not forget to change the variable back when done
  - Example: $T(n) = 2T(n/2) + 1$, $T(1) = 0$
    - Would result in $t_n = t_{n/2} + 1$, not linear
    - However n and $n/2$ are near each other on a logarithmic scale
    - So we let $n = 2^k$ (and so $k = \log n$) and we have: $T(2^k) = 2T(2^{k-1}) + 1$
    - With $t_k = T(2^k)$ we have: $t_k = 2t_{k-1} + 1$
    - Note that $t_0 = T(2^0) = 0$ and $t_1 = 2t_0 + 1 = 1$
    - We already know how to solve that, and we obtain $t_k = c_1 2^k + c_2 1^k$
    - Solving for t_0 and t_1 we obtain $c_1 = 1$ and $c_2 = -1$
    - So $t_k = 2^k - 1$

- Sometimes we do not have a linear recurrence because the indices are nowhere near each other
- We may be able to bring the indices closer using a change of variable
  - Most common change of variable: from n to 2^k
  - Do not forget to change the variable back when done
  - Example: $T(n) = 2T(n/2) + 1, T(1) = 0$
    - Would result in $t_n = t_{n/2} + 1$, not linear
    - However n and $n/2$ are near each other on a logarithmic scale
    - So we let $n = 2^k$ (and so $k = \log n$) and we have: $T(2^k) = 2T(2^{k-1}) + 1$
    - With $t_k = T(2^k)$ we have: $t_k = 2t_{k-1} + 1$
    - Note that $t_0 = T(2^0) = 0$ and $t_1 = 2t_0 + 1 = 1$
    - We already know how to solve that, and we obtain $t_k = c_1 2^k + c_2 1^k$
    - Solving for t_0 and t_1 we obtain $c_1 = 1$ and $c_2 = -1$
    - So $t_k = 2^k - 1$
    - Finally change the variable back to n by replacing k with $\log n$: $t_n = T(n) = n - 1$

RANGE TRANSFORMATION

- Sometimes we do not have a linear recurrence because terms are combined using multiplication
- We may be able to change multiplication into addition by applying an operation on both sides
  - Indeed, $\log a \times b = \log a + \log b$

Range transformation

- Sometimes we do not have a linear recurrence because terms are combined using multiplication
- We may be able to change multiplication into addition by applying an operation on both sides
  - Indeed, $\log a \times b = \log a + \log b$
  - Example: $t_n = 3t_{n-1}^2, t_0 = 1$
    - Not linear because of t_{n-1}^2

Range transformation

- Sometimes we do not have a linear recurrence because terms are combined using multiplication
- We may be able to change multiplication into addition by applying an operation on both sides
  - Indeed, $\log a \times b = \log a + \log b$
  - Example: $t_n = 3t_{n-1}^2, t_0 = 1$
    - Not linear because of t_{n-1}^2
    - Apply log to convert the exponent into a multiplicative constant: $\log t_n = \log 3 + 2 \log t_{n-1}, \log t_0 = \log 1$
    - Let $b_n = \log t_n$ so we have: $b_n = 2b_{n-1} + \log 3, b_0 = 0, b_1 = 2b_0 + \log 3 = \log 3$
    - Linear recurrence!