DISCRETE MATHEMATICS FOR COMPUTER SCIENCE

Dr. QI WANG

Department of Computer Science and Engineering

Office: Room903, Nanshan iPark A7 Building

Email: wangqi@sustc.edu.cn

The Principle of Mathematical Induction

- **Principle.** (*the Weak Principle of Mathematical Induction*)
  (a) If the statement $P(b)$ is true
  (b) the statement $P(n-1) \to P(n)$ is true for all $n > b$,
  then $P(n)$ is true for all integers $n \geq b$

  (a) – *Basic Step* *Inductive Hypothesis*

  (b) – *Inductive Step* *Inductive Conclusion*

Another Form of Induction

- We may have another form of *direct proof* as follows.

Another Form of Induction

- We may have another form of *direct proof* as follows.
  - ◇ First suppose that we have proof of $P(0)$

Another Form of Induction

- We may have another form of *direct proof* as follows.
  - ◇ First suppose that we have proof of $P(0)$
  - ◇ Next suppose that we have a proof that, $\forall k > 0$,

$$P(0) \wedge P(1) \wedge P(2) \wedge \cdots \wedge P(k-1) \rightarrow P(k)$$

Another Form of Induction

- We may have another form of *direct proof* as follows.
  - ◇ First suppose that we have proof of $P(0)$
  - ◇ Next suppose that we have a proof that, $\forall k > 0$,

$$P(0) \wedge P(1) \wedge P(2) \wedge \cdots \wedge P(k-1) \rightarrow P(k)$$

  - ◇ Then, $P(0)$ implies $P(1)$

$P(0) \wedge P(1)$ implies $P(2)$

$P(0) \wedge P(1) \wedge P(2)$ implies $P(3)$ ...

Another Form of Induction

- We may have another form of *direct proof* as follows.
  - ◇ First suppose that we have proof of $P(0)$
  - ◇ Next suppose that we have a proof that, $\forall k > 0$,

$$P(0) \wedge P(1) \wedge P(2) \wedge \cdots \wedge P(k-1) \rightarrow P(k)$$

  - ◇ Then, $P(0)$ implies $P(1)$

$P(0) \wedge P(1)$ implies $P(2)$

$P(0) \wedge P(1) \wedge P(2)$ implies $P(3)$ ...

  - ◇ Iterating gives us a proof of $P(n)$ for all n

Strong Induction

- **Principle** (*The Strong Principle of Mathematical Induction*)

  (a) If the statement $P(b)$ is true

  (b) for all $n > b$, the statement
  $P(b) \wedge P(b+1) \wedge \cdots \wedge P(n-1) \rightarrow P(n)$ is true.

  then $P(n)$ is true for all integers $n \geq b$.

Example

- Prove that every positive integer is a power of a prime or the product of powers of primes.

Example

- Prove that every positive integer is a power of a prime or the product of powers of primes.
  - Base Step: 1 is a power of a prime number, $1 = 2^0$

Example

- Prove that every positive integer is a power of a prime or the product of powers of primes.

  $\diamond$ Base Step: 1 is a power of a prime number, $1 = 2^0$

  $\diamond$ Inductive Hypothesis: Suppose that every number less than n is a power of a prime or a product of powers of primes.

Example

- Prove that every positive integer is a power of a prime or the product of powers of primes.

  ◇ Base Step: 1 is a power of a prime number, $1 = 2^0$

  ◇ Inductive Hypothesis: Suppose that every number less than n is a power of a prime or a product of powers of primes.

  ◇ Then, if n is not a prime power, it is a product of two smaller numbers, each of which is, by the inductive hypothesis, a power a prime power or a product of powers of primes.

Example

- Prove that every positive integer is a power of a prime or the product of powers of primes.

  ◇ Base Step: 1 is a power of a prime number, $1 = 2^0$

  ◇ Inductive Hypothesis: Suppose that every number less than n is a power of a prime or a product of powers of primes.

  ◇ Then, if n is not a prime power, it is a product of two smaller numbers, each of which is, by the inductive hypothesis, a power a prime power or a product of powers of primes.

  ◇ Thus, by the strong principle of mathematical induction, every positive integer is a power of a prime or a product of powers of primes.

Mathematical Induction

- In practice, we do not usually explicitly distinguish between the weak and strong forms.

Mathematical Induction

- In practice, we do not usually explicitly distinguish between the weak and strong forms.

- In reality, they are equivalent to each other in that the weak form is a special case of the strong form, and the strong form can be derived from the weak form.

Summary

- A *typical* proof by mathematical induction, showing that a statement $P(n)$ is true for all integers $n \geq b$ consists of three steps:

Summary

- A *typical* proof by mathematical induction, showing that a statement $P(n)$ is true for all integers $n \geq b$ consists of three steps:

  1. We show that $P(b)$ is true. – Base Step

Summary

- A *typical* proof by mathematical induction, showing that a statement $P(n)$ is true for all integers $n \geq b$ consists of three steps:

1. We show that $P(b)$ is true. – Base Step

2. We then, $\forall n > b$, show either

$$(*) \qquad P(n-1) \rightarrow P(n)$$

or

$$(**) \qquad P(b) \wedge P(b+1) \wedge \cdots \wedge P(n-1) \rightarrow P(n)$$

Summary

- A *typical* proof by mathematical induction, showing that a statement $P(n)$ is true for all integers $n \geq b$ consists of three steps:

  1. We show that $P(b)$ is true. – Base Step
  2. We then, $\forall n > b$, show either

  $$(*) \qquad P(n-1) \to P(n)$$

  or

  $$(**) \qquad P(b) \wedge P(b+1) \wedge \cdots \wedge P(n-1) \to P(n)$$

  We need to make the inductive hypothesis of either $P(n-1)$ or $P(b) \wedge P(b+1) \wedge \cdots \wedge P(n-1)$.We then use $(*)$ or $(**)$ to derive $P(n)$.

Summary

- A *typical* proof by mathematical induction, showing that a statement $P(n)$ is true for all integers $n \geq b$ consists of three steps:

1. We show that $P(b)$ is true. – Base Step
2. We then, $\forall n > b$, show either

   $$(*) \qquad P(n-1) \rightarrow P(n)$$

   or

   $$(**) \qquad P(b) \wedge P(b+1) \wedge \cdots \wedge P(n-1) \rightarrow P(n)$$

   We need to make the inductive hypothesis of either $P(n-1)$ or $P(b) \wedge P(b+1) \wedge \cdots \wedge P(n-1)$.We then use $(*)$ or $(**)$ to derive $P(n)$.

3. We conclude on the basis of the principle of mathematical induction that $P(n)$ is true for all $n \geq b$.

Recursion

- Recursive computer programs or algorithms often lead to inductive analysis.

Recursion

- Recursive computer programs or algorithms often lead to inductive analysis.

- A classical example of *recursion* is the **Towers of Hanoi** Problem.

Towers of Hanoi

Towers of Hanoi

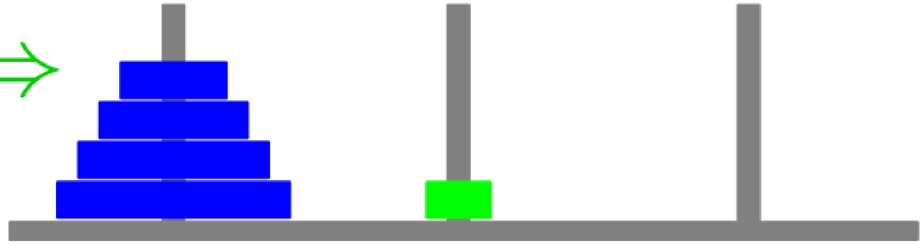
- 3 pegs; n disks of different sizes
- A *legal move* takes a disk from one peg and moves it onto another peg so that it is not on top of a smaller disk
- **Problem**: Find a (efficient) way to move all of the disks from one peg to another

Towers of Hanoi

Towers of Hanoi

legal move

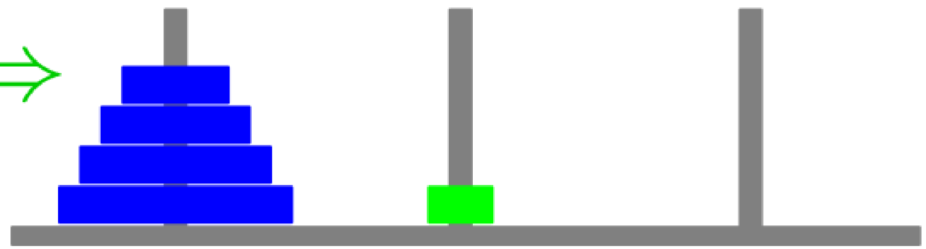
$\Rightarrow$



Towers of Hanoi

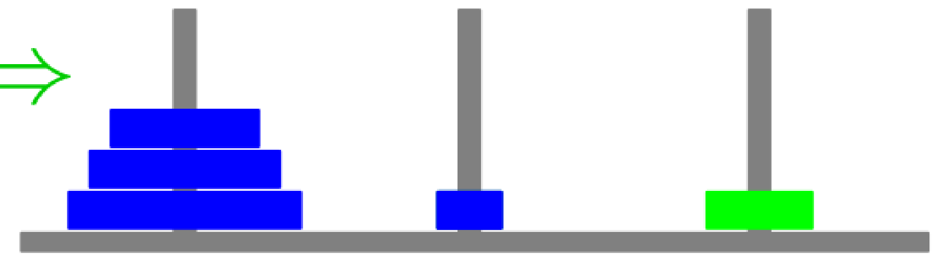
legal move

$\Rightarrow$



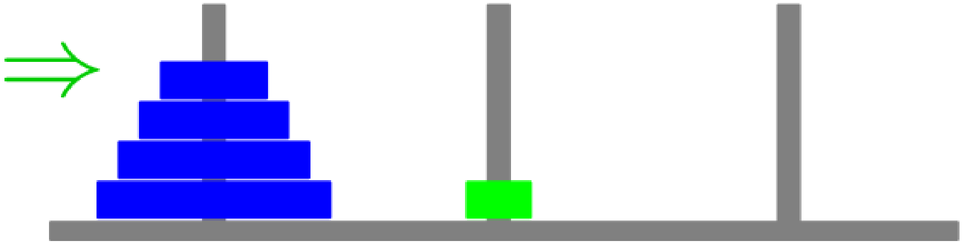
legal move

$\Rightarrow$

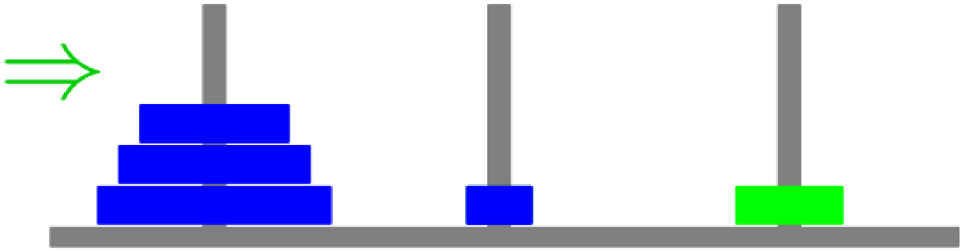


Towers of Hanoi

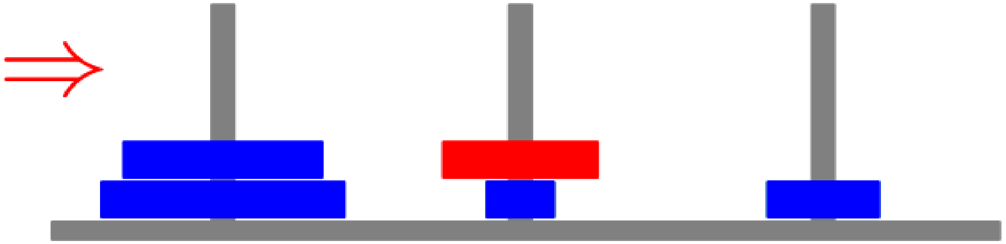
legal move



legal move

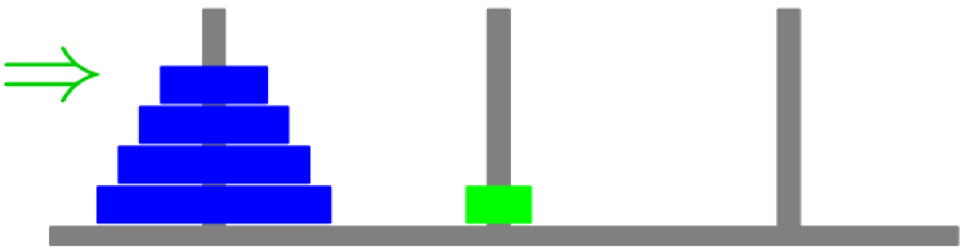


not legal

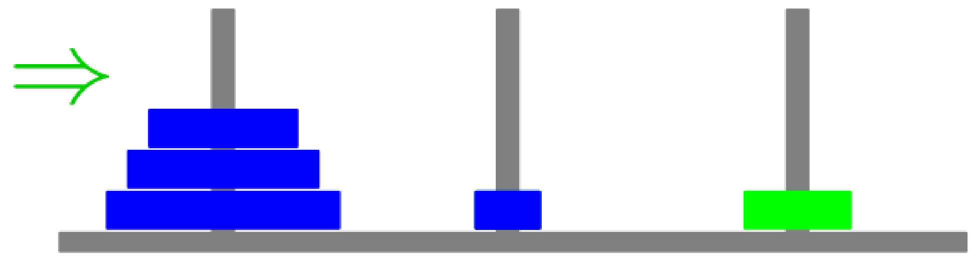


Towers of Hanoi

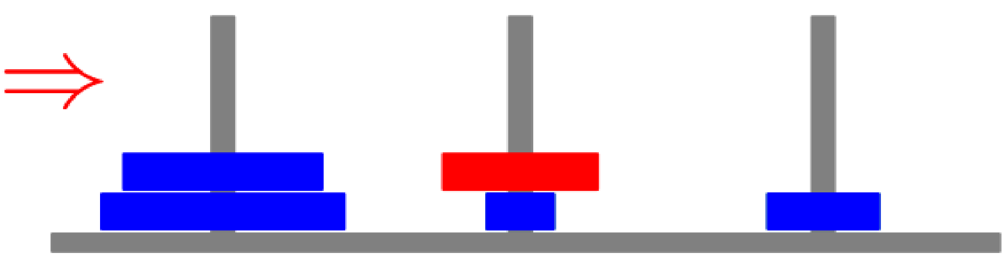
legal move



legal move



not legal



legal move

Towers of Hanoi

- **Problem:** Start with n disks on leftmost peg

Towers of Hanoi

- **Problem:** Start with n disks on leftmost peg

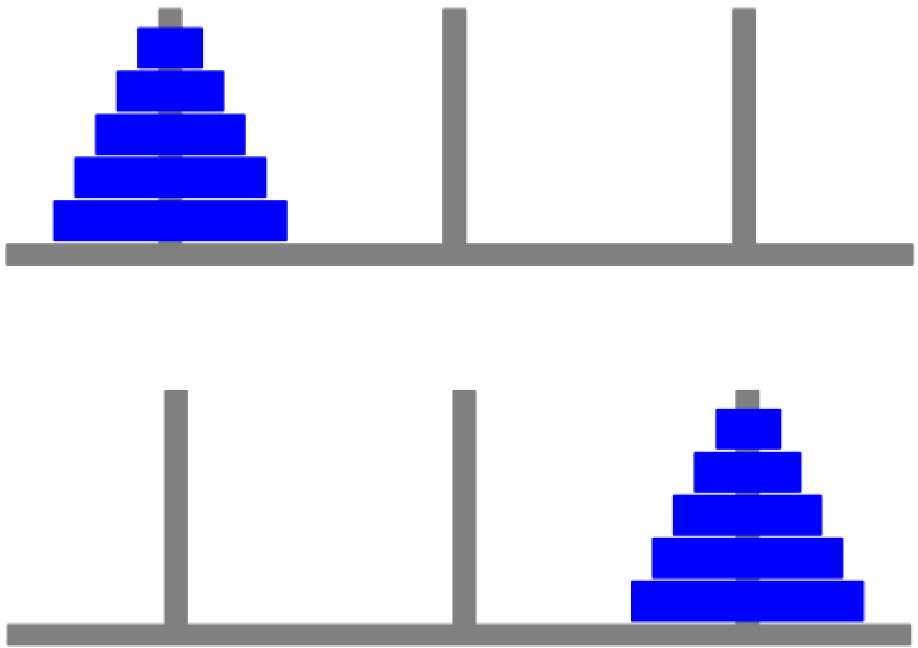
  using only legal moves

Towers of Hanoi

- **Problem:** Start with n disks on leftmost peg

  using only legal moves

  move all disks to rightmost peg.

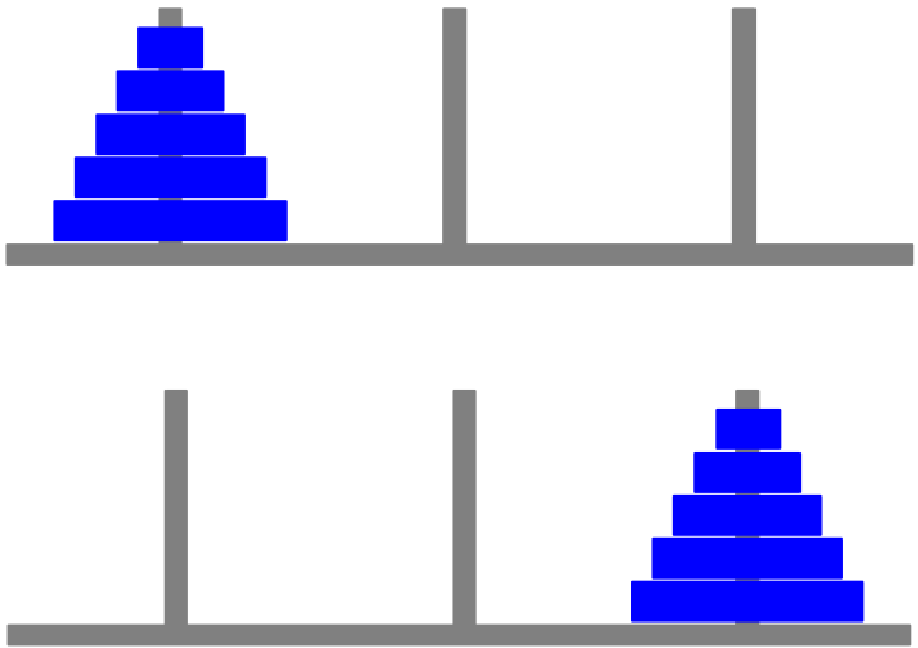


Towers of Hanoi

- **Problem:** Start with n disks on leftmost peg

  using only legal moves

  move all disks to rightmost peg.



Given $i, j \in \{1,2,3\}$, let
$\overline{\{i,j\}} = \{1,2,3\} - \{i\} - \{j\}$,
i.e., $\overline{\{1,2\}} = \{3\}$, $\overline{\{1,3\}} = \{2\}$,
$\overline{\{2,3\}} = \{1\}$.

Towers of Hanoi

- General solution

Towers of Hanoi

- General solution

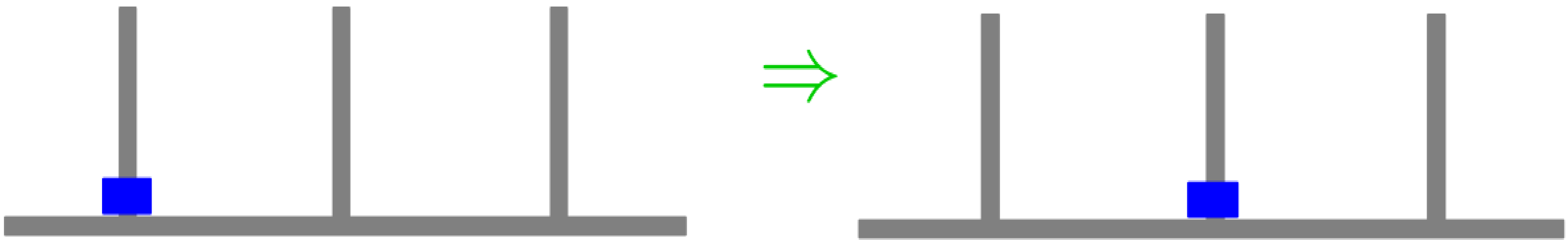
  Recursion Base:

  If $n = 1$, moving one disk from i to j is easy. Just move it.

- General solution

  Recursion Base:

  If $n = 1$, moving one disk from i to j is easy. Just move it.



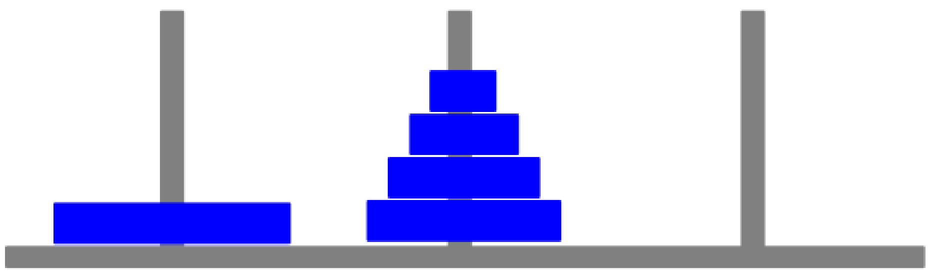
Towers of Hanoi

To move $n > 1$ disks from i to j

Towers of Hanoi

To move $n > 1$ disks from i to j

1)

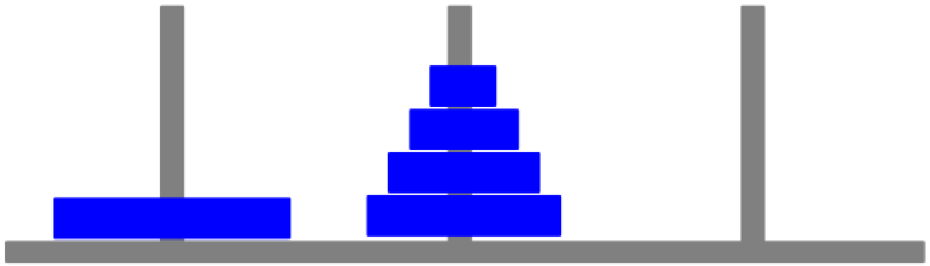


move top $n-1$ disks from i to $\overline{\{i,j\}}$

Towers of Hanoi

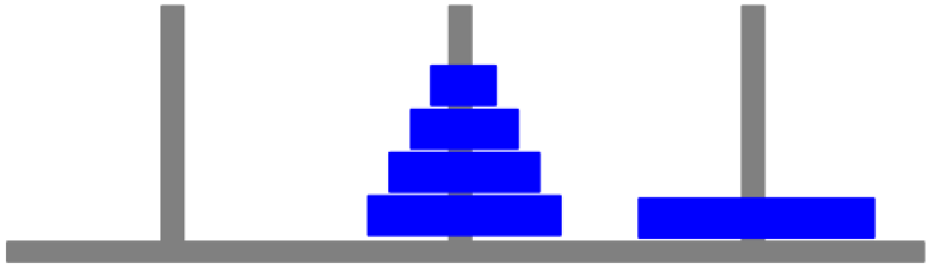
To move $n > 1$ disks from i to j

1)



move top $n-1$ disks from i to $\overline{\{i,j\}}$

2)

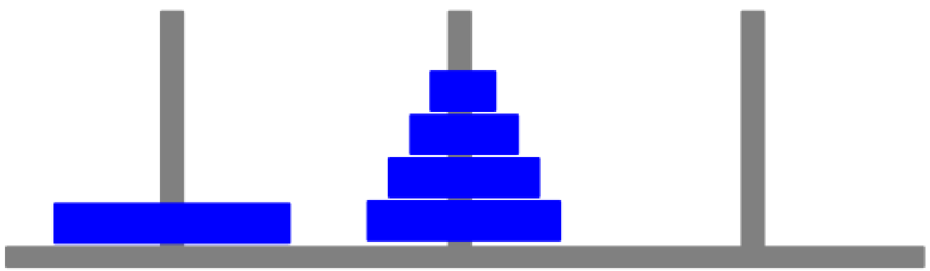


move largest disk from i to j

Towers of Hanoi

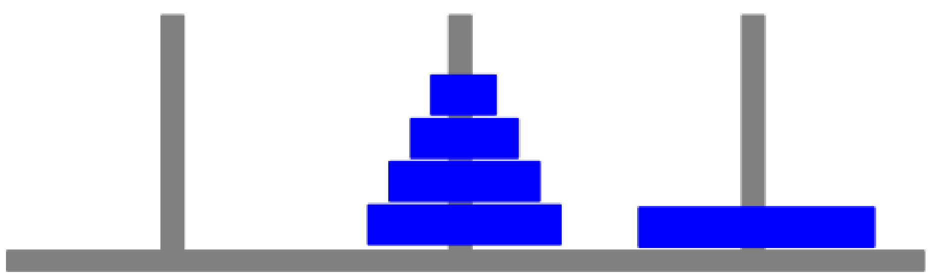
To move $n > 1$ disks from i to j

1)



move top $n-1$ disks from i to $\overline{\{i,j\}}$

2)



move largest disk from i to j

3)

move top $n-1$ disks from $\overline{\{i,j\}}$ to j

Towers of Hanoi

```java
public class Hanoi
{

    public void move(int n, char a, char b, char c)
    {
        if (n == 1)
            System.out.println("plate " + n + " from " + a + " to " + c);
        else
        {
            move(n-1,a,c,b);
            System.out.println("plate " + n + " from " + a + " to " + c);
            move(n-1,b,a,c);
        }

    }

```

Towers of Hanoi

To move n disks from i to j

i) move top $n-1$ disks from i to $\overline{\{i,j\}}$

ii) move largest disk from i to j

iii) move top $n-1$ disks from $\overline{\{i,j\}}$ to j

Towers of Hanoi

- To prove Correctness of solution, we are implicitly using induction

To move n disks from i to j

i) move top $n-1$ disks from i to $\overline{\{i,j\}}$

ii) move largest disk from i to j

iii) move top $n-1$ disks from $\overline{\{i,j\}}$ to j

Towers of Hanoi

- To prove Correctness of solution, we are implicitly using induction

- $p(n)$ is statement that algorithm is correct for n

To move n disks from i to j

i) move top $n-1$ disks from i to $\overline{\{i,j\}}$

ii) move largest disk from i to j

iii) move top $n-1$ disks from $\overline{\{i,j\}}$ to j

Towers of Hanoi

- To prove Correctness of solution, we are implicitly using induction

- $p(n)$ is statement that algorithm is correct for n

To move n disks from i to j

i) move top $n-1$ disks from i to $\overline{\{i,j\}}$

ii) move largest disk from i to j

iii) move top $n-1$ disks from $\overline{\{i,j\}}$ to j

- $p(1)$ is statement that algorithm works for $n = 1$ disks, which is obviously true

Towers of Hanoi

- To prove Correctness of solution, we are implicitly using induction

- $p(n)$ is statement that algorithm is correct for n

To move n disks from i to j

i) move top $n-1$ disks from i to $\overline{\{i,j\}}$

ii) move largest disk from i to j

iii) move top $n-1$ disks from $\overline{\{i,j\}}$ to j

- $p(1)$ is statement that algorithm works for $n=1$ disks, which is obviously true

- $p(n-1) \to p(n)$ is *recursion* statement that

  if our algorithm works for $n-1$ disks, then we can build a correct solution for n disks

Towers of Hanoi

- Running time

  $M(n)$ is number of disk moves needed for n disks

To move n disks from i to j

i) move top $n-1$ disks from i to $\overline{\{i,j\}}$

ii) move largest disk from i to j

iii) move top $n-1$ disks from $\overline{\{i,j\}}$ to j

Towers of Hanoi

- Running time

$M(n)$ is number of disk moves needed for n disks

To move n disks from i to j

i) move top $n-1$ disks from i to $\overline{\{i,j\}}$

ii) move largest disk from i to j

iii) move top $n-1$ disks from $\overline{\{i,j\}}$ to j

$M(1) = 1$

if $n > 1$, then $M(n) = 2M(n-1) + 1$

Towers of Hanoi

- We saw that $M(1) = 1$ and that
- $M(n) = 2M(n-1) + 1$ for $n > 1$

Towers of Hanoi

- We saw that $M(1) = 1$ and that
- $M(n) = 2M(n-1) + 1$ for $n > 1$

- Iterating the recurrence gives

$$M(1) = 1,\ M(2) = 3,\ M(3) = 7,$$
$$M(4) = 15,\ M(5) = 31,\ \ldots$$

Towers of Hanoi

- We saw that $M(1) = 1$ and that
- $M(n) = 2M(n-1) + 1$ for $n > 1$

- Iterating the recurrence gives

$$M(1) = 1,\ M(2) = 3,\ M(3) = 7,$$
$$M(4) = 15,\ M(5) = 31,\ \ldots$$

- We *guess* that $M(n) = 2^n - 1$

Towers of Hanoi

- We saw that $M(1) = 1$ and that
- $M(n) = 2M(n-1) + 1$ for $n > 1$

- Iterating the recurrence gives

  $$M(1) = 1,\ M(2) = 3,\ M(3) = 7,$$
  $$M(4) = 15,\ M(5) = 31,\ \ldots$$

- We *guess* that $M(n) = 2^n - 1$

  We'll prove this by induction

Towers of Hanoi

- We saw that $M(1) = 1$ and that
- $M(n) = 2M(n-1) + 1$ for $n > 1$

- Iterating the recurrence gives

  $M(1) = 1$, $M(2) = 3$, $M(3) = 7$,

  $M(4) = 15$, $M(5) = 31$, ...

- We *guess* that $M(n) = 2^n - 1$

  We'll prove this by induction

  Later, we'll also see how to solve without guessing

Towers of Hanoi

- Formally, given

$$M(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2M(n-1) + 1 & \text{otherwise} \end{cases}$$

We show that $M(n) = 2^n - 1$.

Towers of Hanoi

- Formally, given

$$M(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2M(n-1)+1 & \text{otherwise} \end{cases}$$

We show that $M(n) = 2^n - 1$.

Proof. (by induction)

The base case $n = 1$ is true, since $2^1 - 1 = 1$.

For the inductive step, assume that $M(n-1) = 2^{n-1} - 1$ for $n > 1$.

Towers of Hanoi

- Formally, given

$$M(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2M(n-1)+1 & \text{otherwise} \end{cases}$$

We show that $M(n) = 2^n - 1$.

Proof. (by induction)

The base case $n = 1$ is true, since $2^1 - 1 = 1$.

For the inductive step, assume that $M(n-1) = 2^{n-1} - 1$ for $n > 1$.

Then $M(n) = 2M(n-1) + 1 = 2(2^{n-1} - 1) + 1 = 2^n - 1$

Towers of Hanoi

- Note that we used induction twice.

Towers of Hanoi

- Note that we used induction twice.
- The first time was to derive correctness of algorithm and the recurrence

$$M(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2M(n-1)+1 & \text{otherwise} \end{cases}$$

Towers of Hanoi

- Note that we used induction twice.
- The first time was to derive correctness of algorithm and the recurrence

$$M(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2M(n-1)+1 & \text{otherwise} \end{cases}$$

- The second time was to derive the closed form solution $M(n) = 2^n - 1$ of the recurrence.

Recurrences

- A *recurrence equation* or *recurrence* for a function defined on the set of integers $\geq b$ is one that tells us how to compute the nth value from some or all the first $n-1$ values.

Recurrences

- A *recurrence equation* or *recurrence* for a function defined on the set of integers $\geq b$ is one that tells us how to compute the nth value from some or all the first $n-1$ values.

  To completely specify a function on the basis of a recurrence, we have to give the *initial condition(s)* (a.k.a. the *base case(s)*) for the recurrence.

Recurrences

- A *recurrence equation* or *recurrence* for a function defined on the set of integers $\geq b$ is one that tells us how to compute the nth value from some or all the first $n-1$ values.

  To completely specify a function on the basis of a recurrence, we have to give the *initial condition(s)* (a.k.a. the *base case(s)*) for the recurrence.

$$M(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2M(n-1)+1 & \text{otherwise} \end{cases}$$

Towers of Hanoi

Fibonacci Sequence

$$F(n) = \begin{cases} 1 & \text{if } n = 0, 1 \\ F(n-1)+F(n-2) & \text{otherwise} \end{cases}$$

- **Example 2**: Let $S(n)$ be the number of subsets of a set of size n. What is the formula for $S(n)$?

  The empty set, of size $n = 0$ has only one subset (itself), so $S(0) = 1$.

  It is not difficult to see that

  $S(1) = 2$, $S(2) = 4$, $S(3) = 8$

- **Example 2**: Let $S(n)$ be the number of subsets of a set of size n. What is the formula for $S(n)$?

  The empty set, of size $n = 0$ has only one subset (itself), so $S(0) = 1$.

  It is not difficult to see that

  $S(1) = 2$, $S(2) = 4$, $S(3) = 8$

  We "guess" that $S(n) = 2^n$. But, in order to prove formula, we'll need to think recursively.

Recurrences

- Consider the eight subsets of $\{1, 2, 3\}$:

  $\emptyset, \{1\}, \{2\}, \{1, 2\}, \{3\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}$

Recurrences

- Consider the eight subsets of $\{1, 2, 3\}$:

$$\emptyset, \{1\}, \{2\}, \{1, 2\}, \{3\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}$$

$$\begin{array}{cccc} \emptyset & \{1\} & \{2\} & \{1, 2\} \\ \{3\} & \{1, 3\} & \{2, 3\} & \{1, 2, 3\} \end{array}$$

Recurrences

- Consider the eight subsets of $\{1,2,3\}$:

$$\emptyset, \{1\}, \{2\}, \{1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}$$

$$\begin{array}{cccc} \emptyset & \{1\} & \{2\} & \{1,2\} \\ \{3\} & \{1,3\} & \{2,3\} & \{1,2,3\} \end{array}$$

First four subsets are exactly the subsets of $\{1,2\}$, while second four are the subsets of $\{1,2\}$ with 3 added into each.

- Consider the eight subsets of $\{1,2,3\}$:

  $\emptyset, \{1\}, \{2\}, \{1,2\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}$

$$\begin{array}{cccc} \emptyset & \{1\} & \{2\} & \{1,2\} \\ \{3\} & \{1,3\} & \{2,3\} & \{1,2,3\} \end{array}$$

  First four subsets are exactly the subsets of $\{1,2\}$, while second four are the subsets of $\{1,2\}$ with 3 added into each.

  So, we get a subset of $\{1,2,3\}$ either by taking a subset of $\{1,2\}$ or by adjoining 3 to a subset of $\{1,2\}$.

Recurrences

- Consider the eight subsets of $\{1, 2, 3\}$:

  $\emptyset, \{1\}, \{2\}, \{1, 2\}, \{3\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}$

  $$\begin{array}{cccc} \emptyset & \{1\} & \{2\} & \{1,2\} \\ \{3\} & \{1,3\} & \{2,3\} & \{1,2,3\} \end{array}$$

  First four subsets are exactly the subsets of $\{1, 2\}$, while second four are the subsets of $\{1, 2\}$ with 3 added into each.

  So, we get a subset of $\{1, 2, 3\}$ either by taking a subset of $\{1, 2\}$ or by adjoining 3 to a subset of $\{1, 2\}$.

  This suggests that the recurrence for the number of subsets of an n-element set $\{1, 2, \ldots, n\}$ is

  $$S(n) = \begin{cases} 1 & \text{if } n = 0 \\ 2S(n-1) & \text{if } n \geq 1 \end{cases}$$

Recurrences

- **Proof.** of correctness of this recurrence

Recurrences

- **Proof.** of correctness of this recurrence

  The subsets of $\{1, 2, \ldots, n\}$ can be partitioned according to whether or not they contain element n.

Recurrences

- **Proof.** of correctness of this recurrence

  The subsets of $\{1, 2, \ldots, n\}$ can be partitioned according to whether or not they contain element n.

  Each subset S containing n can be constructed in a unique fashion by adding n to the subset $S - \{n\}$ not containing n.

  Each subset S not containing n can be constructed by removing n from the unique set $S \cup \{n\}$ containing n.

Recurrences

- **Proof.** of correctness of this recurrence

  The subsets of $\{1, 2, \ldots, n\}$ can be partitioned according to whether or not they contain element n.

  Each subset S containing n can be constructed in a unique fashion by adding n to the subset $S - \{n\}$ not containing n.

  Each subset S not containing n can be constructed by removing n from the unique set $S \cup \{n\}$ containing n.

  So, the number of subsets containing n is exactly the same as the number of subsets not containing n.

Recurrences

- **Proof.** of correctness of this recurrence

  The subsets of $\{1, 2, \ldots, n\}$ can be partitioned according to whether or not they contain element n.

  Each subset S containing n can be constructed in a unique fashion by adding n to the subset $S - \{n\}$ not containing n.

  Each subset S not containing n can be constructed by removing n from the unique set $S \cup \{n\}$ containing n.

  So, the number of subsets containing n is exactly the same as the number of subsets not containing n.

  Thus, if $n > 1$, then $S(n) = 2S(n-1)$.

Recurrences

- **Proof.** of correctness of this recurrence

The subsets of $\{1, 2, \ldots, n\}$ can be partitioned according to whether or not they contain element n.

Each subset S containing n can be constructed in a unique fashion by adding n to the subset $S - \{n\}$ not containing n.

Each subset S not containing n can be constructed by removing n from the unique set $S \cup \{n\}$ containing n.

So, the number of subsets containing n is exactly the same as the number of subsets not containing n.

Thus, if $n > 1$, then $S(n) = 2S(n-1)$.

Proof by induction is easy.

Iterating a Recurrence

- Let $T(n) = rT(n-1) + a$,

  where r and a are constants.

Iterating a Recurrence

- Let $T(n) = rT(n-1) + a$,

  where r and a are constants.

  Find a recurrence that expresses

  $T(n)$ in terms of $T(n-2)$
  $T(n)$ in terms of $T(n-3)$
  $T(n)$ in terms of $T(n-4)$
  $\vdots$

Iterating a Recurrence

- Let $T(n) = rT(n-1) + a$,

  where r and a are constants.

  Find a recurrence that expresses

  $T(n)$ in terms of $T(n-2)$
  $T(n)$ in terms of $T(n-3)$
  $T(n)$ in terms of $T(n-4)$
  $\vdots$

  Can we generalize this to find a closed form solution?

Iterating a Recurrence

- Note that $T(n) = rT(n-1) + a$ implies that

  $\forall i < n,\ T(n-i) = rT((n-i)-1)) + a$

Iterating a Recurrence

- Note that $T(n) = rT(n-1) + a$ implies that

  $\forall i < n,\ T(n-i) = rT((n-i)-1)) + a$

  Then, we have

$$
\begin{aligned}
T(n) &= rT(n-1) + a \\
&= r(rT(n-2) + a) + a \\
&= r^2 T(n-2) + ra + a \\
&= r^2(rT(n-3) + a) + ra + a \\
&= r^3 T(n-3) + r^2 a + ra + a \\
&= r^3(rT(n-4) + a) + r^2 a + ra + a \\
&= r^4 T(n-4) + r^3 a + r^2 a + ra + a.
\end{aligned}
$$

Iterating a Recurrence

- Note that $T(n) = rT(n-1) + a$ implies that

  $$\forall i < n,\ T(n-i) = rT((n-i)-1)) + a$$

  Then, we have

$$\begin{aligned}
T(n) &= rT(n-1) + a \\
&= r(rT(n-2) + a) + a \\
&= r^2 T(n-2) + ra + a \\
&= r^2(rT(n-3) + a) + ra + a \\
&= r^3 T(n-3) + r^2 a + ra + a \\
&= r^3(rT(n-4) + a) + r^2 a + ra + a \\
&= r^4 T(n-4) + r^3 a + r^2 a + ra + a.
\end{aligned}$$

  Guess $T(n) = r^n T(0) + a \sum_{i=0}^{n-1} r^i$

Iterating a Recurrence

- The method we used to guess the solution is called *iterating the recurrence*, because we repeatedly (iteratively) use the recurrence.

Iterating a Recurrence

- The method we used to guess the solution is called *iterating the recurrence*, because we repeatedly (iteratively) use the recurrence.

  Another approach is to iterate from the "bottom-up" instead of "top-down".

Iterating a Recurrence

- The method we used to guess the solution is called *iterating the recurrence*, because we repeatedly (iteratively) use the recurrence.

  Another approach is to iterate from the "bottom-up" instead of "top-down".

  $$\begin{aligned}
  T(0) &= b \\
  T(1) &= rT(0) + a = rb + a \\
  T(2) &= rT(1) + a = r(rb + a) + a = r^2b + ra + a \\
  T(3) &= rT(2) + a = r^3b + r^2a + ra + a
  \end{aligned}$$

Iterating a Recurrence

- The method we used to guess the solution is called *iterating the recurrence*, because we repeatedly (iteratively) use the recurrence.

  Another approach is to iterate from the "bottom-up" instead of "top-down".

  $$\begin{aligned}T(0) &= b\\ T(1) &= rT(0) + a = rb + a\\ T(2) &= rT(1) + a = r(rb + a) + a = r^2 b + ra + a\\ T(3) &= rT(2) + a = r^3 b + r^2 a + ra + a\end{aligned}$$

  This would lead to the same guess

  $$T(n) = r^n b + a \sum_{i=0}^{n-1} r^i.$$

Formula of Recurrences

- **Theorem** If $T(n) = rT(n-1) + a$, $T(0) = b$, and $r \neq 1$, then

$$T(n) = r^n b + a\frac{1 - r^n}{1 - r}$$

for all nonnegative integers n.

- **Theorem** If $T(n) = rT(n-1) + a$, $T(0) = b$, and $r \neq 1$, then

$$T(n) = r^n b + a\frac{1 - r^n}{1 - r}$$

for all nonnegative integers n.

Proof by induction

The base case:

$$T(0) = r^0 b + a\frac{1 - r^0}{1 - r} = b.$$

So the formula is true when $n = 0$.

Now assume that $n > 0$ and

$$T(n-1) = r^{n-1} b + a\frac{1 - r^{n-1}}{1 - r}.$$

- **Proof by induction**

$$
\begin{aligned}
T(n) &= rT(n-1)+a \\
&= r\left(r^{n-1}b + a\frac{1-r^{n-1}}{1-r}\right)+a \\
&= r^n b + \frac{ar-ar^n}{1-r}+a \\
&= r^n b + \frac{ar-ar^n+a-ar}{1-r} \\
&= r^n b + a\frac{1-r^n}{1-r}.
\end{aligned}
$$

Formula of Recurrences

- **Theorem** If $T(n) = rT(n-1) + a$, $T(0) = b$, and $r \neq 1$, then

$$T(n) = r^n b + a\frac{1 - r^n}{1 - r}$$

for all nonnegative integers n.

Formula of Recurrences

- **Theorem** If $T(n) = rT(n-1) + a$, $T(0) = b$, and $r \neq 1$, then

$$T(n) = r^n b + a\frac{1-r^n}{1-r}$$

for all nonnegative integers n.

Example:
$T(n) = 3T(n-1) + 2$ with $T(0) = 5$

Formula of Recurrences

- **Theorem** If $T(n) = rT(n-1) + a$, $T(0) = b$, and $r \neq 1$, then

$$T(n) = r^n b + a\frac{1-r^n}{1-r}$$

for all nonnegative integers n.

Example:
$T(n) = 3T(n-1) + 2$ with $T(0) = 5$

Plugging $r = 3, a = 2, b = 5$ in the formula, gives

$$T(n) = 3^n \cdot 5 + 2\frac{1-3^n}{1-3} = 3^n \cdot 6 - 1$$

First-Order Linear Recurrences

- A recurrence of the form $T(n) = f(n)T(n-1) + g(n)$ is called a *first-order linear recurrence*.

First-Order Linear Recurrences

- A recurrence of the form $T(n) = f(n)T(n-1) + g(n)$ is called a *first-order linear recurrence*.
  - ◇ First Order because it only depends upon going back one step, i.e., $T(n-1)$

First-Order Linear Recurrences

- A recurrence of the form $T(n) = f(n)T(n-1) + g(n)$ is called a *first-order linear recurrence*.

  ◇ First Order because it only depends upon going back one step, i.e., $T(n-1)$

  If it depends upon $T(n-2)$, it would be a second-order recurrence, e.g., $T(n) = T(n-1) + 2T(n-2)$.

First-Order Linear Recurrences

- A recurrence of the form $T(n) = f(n)T(n-1) + g(n)$ is called a *first-order linear recurrence*.

  ◇ First Order because it only depends upon going back one step, i.e., $T(n-1)$

  If it depends upon $T(n-2)$, it would be a second-order recurrence, e.g., $T(n) = T(n-1) + 2T(n-2)$.

  ◇ Linear because $T(n-1)$ only appears to the first power.

First-Order Linear Recurrences

- A recurrence of the form $T(n) = f(n)T(n-1) + g(n)$ is called a *first-order linear recurrence*.

  ◇ First Order because it only depends upon going back one step, i.e., $T(n-1)$

  If it depends upon $T(n-2)$, it would be a second-order recurrence, e.g., $T(n) = T(n-1) + 2T(n-2)$.

  ◇ Linear because $T(n-1)$ only appears to the first power.

  Something like $T(n) = (T(n-1))^2 + 3$ would be a non-linear first-order recurrence relation.

First-Order Linear Recurrences

- $T(n) = f(n)T(n-1) + g(n)$

First-Order Linear Recurrences

- $T(n) = f(n)T(n-1) + g(n)$

  When $f(n)$ is a constant, say r, the general solution is almost as easy as we derived before. Iterating the recurrence gives

$$
\begin{aligned}
T(n) &= rT(n-1) + g(n) \\
&= r(rT(n-2) + g(n-1)) + g(n) \\
&= r^2 T(n-2) + rg(n-1) + g(n) \\
&= r^3 T(n-3) + r^2 g(n-2) + rg(n-1) + g(n) \\
&\vdots \\
&= r^n T(0) + \sum_{i=0}^{n-1} r^i g(n-i)
\end{aligned}
$$

First-Order Linear Recurrences

- **Theorem** For any positive constants a and r, and any function g defined on nonnegative integers, the solution to the first-order linear recurrence

$$T(n) = \begin{cases} rT(n-1) + g(n) & \text{if } n > 0 \\ a & \text{if } n = 0 \end{cases}$$

is

$$T(n) = r^n a + \sum_{i=1}^{n} r^{n-i} g(i).$$

First-Order Linear Recurrences

- **Theorem** For any positive constants a and r, and any function g defined on nonnegative integers, the solution to the first-order linear recurrence

$$T(n) = \begin{cases} rT(n-1) + g(n) & \text{if } n > 0 \\ a & \text{if } n = 0 \end{cases}$$

is

$$T(n) = r^n a + \sum_{i=1}^{n} r^{n-i} g(i).$$

Proof by induction

Examples

- Solve $T(n) = 4T(n-1) + 2^n$ with $T(0) = 6$

Examples

- Solve $T(n) = 4T(n-1) + 2^n$ with $T(0) = 6$

$$
\begin{aligned}
T(n) &= 6 \cdot 4^n + \sum_{i=1}^{n} 4^{n-i} \cdot 2^i \\
&= 6 \cdot 4^n + 4^n \sum_{i=1}^{n} 4^{-i} \cdot 2^i \\
&= 6 \cdot 4^n + 4^n \sum_{i=1}^{n} (\frac{1}{2})^i \\
&= 6 \cdot 4^n + (1 - \frac{1}{2^n}) \cdot 4^n \\
&= 7 \cdot 4^n - 2^n.
\end{aligned}
$$

Examples

- Solve $T(n) = 3T(n-1) + n$ with $T(0) = 10$

Examples

- Solve $T(n) = 3T(n-1) + n$ with $T(0) = 10$

$$
\begin{aligned}
T(n) &= 10 \cdot 3^n + \sum_{i=1}^{n} 3^{n-i} \cdot i \\
&= 10 \cdot 3^n + 3^n \sum_{i=1}^{n} i \cdot 3^{-i}
\end{aligned}
$$

Examples

- Solve $T(n) = 3T(n-1) + n$ with $T(0) = 10$

$$\begin{aligned} T(n) &= 10 \cdot 3^n + \sum_{i=1}^{n} 3^{n-i} \cdot i \\ &= 10 \cdot 3^n + 3^n \sum_{i=1}^{n} i \cdot 3^{-i} \end{aligned}$$

Theorem. For any real number $x \neq 1$,

$$\sum_{i=1}^{n} i x^i = \frac{n x^{n+2} - (n+1) x^{n+1} + x}{(1-x)^2}.$$

Examples

- Solve $T(n) = 3T(n-1) + n$ with $T(0) = 10$

$$
\begin{aligned}
T(n) &= 10 \cdot 3^n + \sum_{i=1}^{n} 3^{n-i} \cdot i \\
&= 10 \cdot 3^n + 3^n \sum_{i=1}^{n} i \cdot 3^{-i} \\
&= 10 \cdot 3^n + 3^n \left(-\frac{3}{2}(n+1)3^{-(n+1)} - \frac{3}{4}3^{-(n+1)} + \frac{3}{4} \right) \\
&= \frac{43}{4}3^n - \frac{n+1}{2} - \frac{1}{4}.
\end{aligned}
$$

Next Lecture

- recurrence, more counting ...