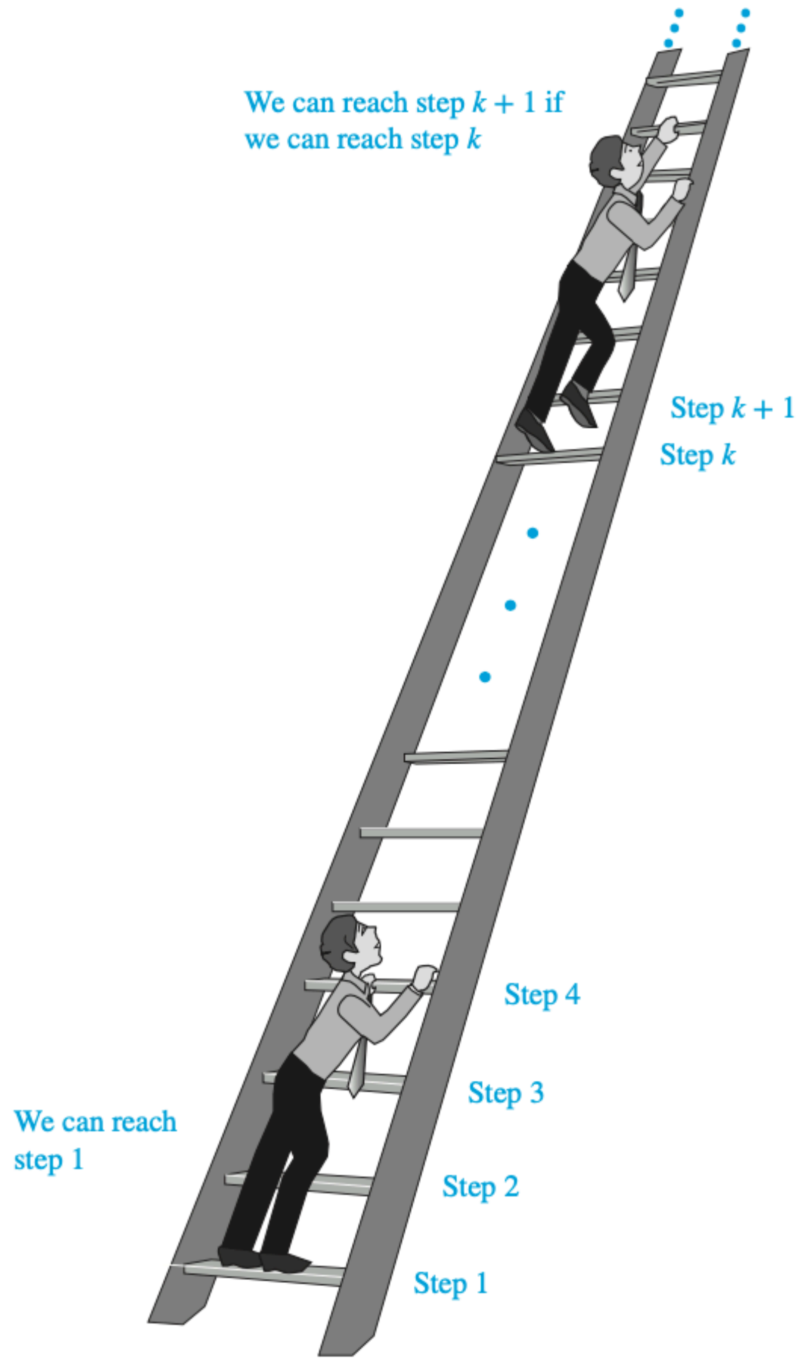
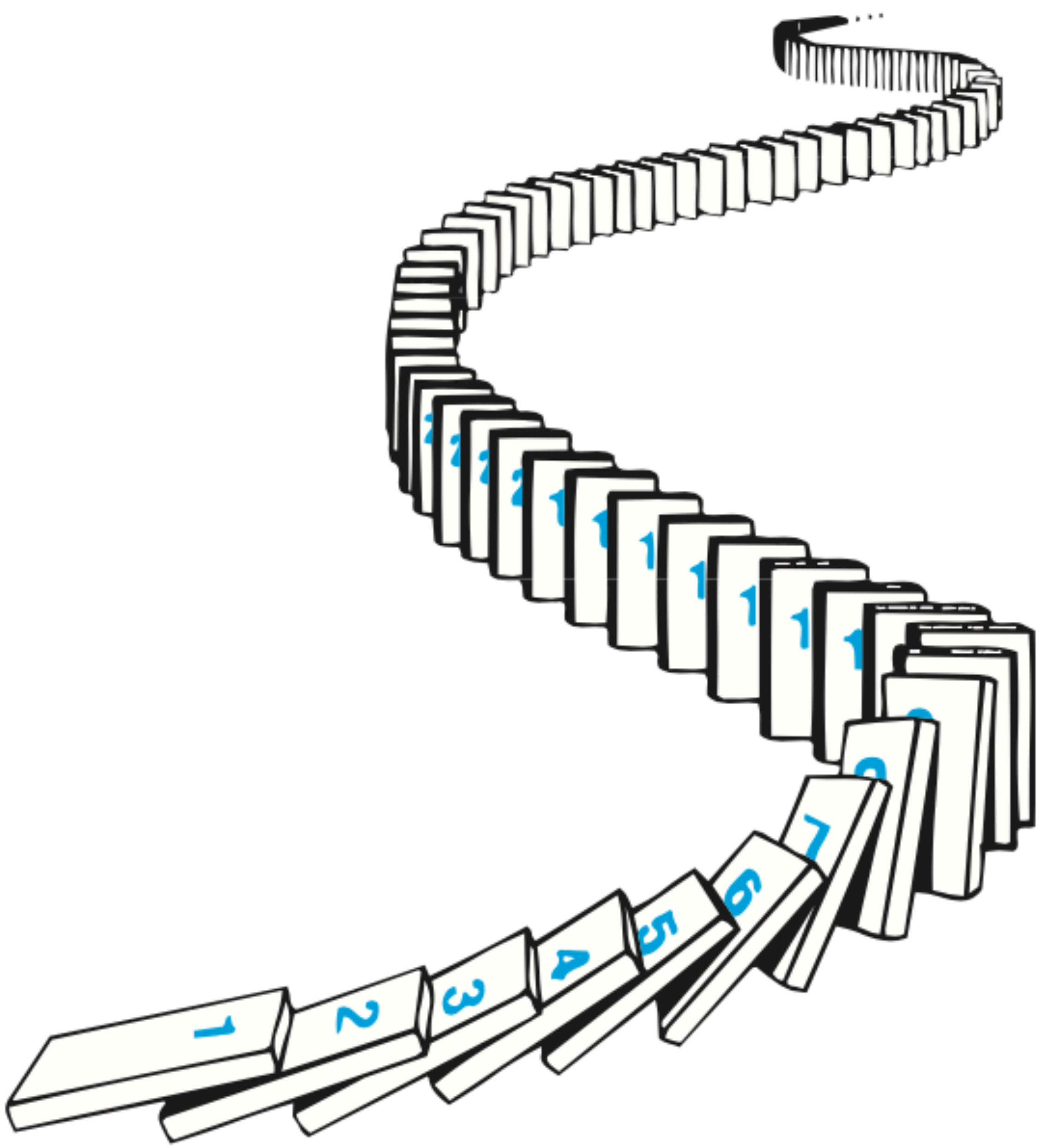
06 Induction and Recursion

CS201 Discrete Mathematics

Instructor: Shan Chen

Mathematical Induction





Mathematical Induction

Principle of Mathematical Induction

- Let $P(n)$ be a predicate, i.e., $P(n)$ is either true or false when n is a specific number.

- **Principle of Mathematical Induction:** To prove that $P(n)$ is true for all positive integers $n \in \boldsymbol{Z}^+$, we complete two steps:
  - **Basis step:** prove $P(1)$ is true
  - **Inductive step:** prove $\forall k \in \boldsymbol{Z}^+, P(k) \rightarrow P(k+1)$ is true

    * *"P(k) is true" is called the inductive hypothesis (IH)* 归纳假设

- **Q:** Why this principle is valid?
  - Proof by contradiction: Assume $P(n)$ is false for some integer $n \geq 1$, then the set S of all positive integer n such that $P(n)$ is false is not empty. Let m be the smallest integer in S. * *why such m exists?* We have $m \geq 2$ as $P(1)$ is true. However, since $P(m-1)$ is true and $P(m-1) \rightarrow P(m)$ is true, $P(m)$ must be true, contradiction!

Principle of Mathematical Induction

- **Principle of Mathematical Induction:** To prove that $P(n)$ is true for all positive integers $n \in \mathbf{Z}^+$, we complete two steps:
  - **Basis step:** prove $P(1)$ is true
  - **Inductive step:** prove $\forall k \in \mathbf{Z}^+, P(k) \rightarrow P(k+1)$ is true
    ** "P(k) is true" is called the inductive hypothesis (IH) 归纳假设*

- **Well-Ordering Principle**: every nonempty subset of $\mathbf{Z}^+$ has a least/minimum element. ** this is an axiom 公理*
  - This principle is equivalent to mathematical induction.
    ** the proof is left as an assignment problem*
  - This also means mathematical induction can be generalized from $\mathbf{Z}^+$ to any well-ordered set S, e.g., $\mathbf{N}$, $\{n \in \mathbf{Z} \mid n \geq b\}$, etc.

Example

- Show that $1 + 2 + \cdots + n = n(n + 1)/2$ for any positive integer n.

- Proof by (mathematical) induction:
  - Let $P(n)$ be the predicate that the sum of the first n positive integers is equal to $n(n + 1)/2$.
  - **Basis step:** $P(1)$ is true, because $1 = 1(1 + 1)/2$.
  - **Inductive step:** From the inductive hypothesis, i.e., $P(k)$ is true for an arbitrary positive integer k, we need to show that $P(k + 1)$ is true, i.e., $1 + 2 + \cdots + k + 1 = (k + 1)((k + 1) + 1)/2$.

    $$1 + 2 + \cdots + k + (k + 1) = k(k + 1)/2 + k + 1$$
    $$= (k(k + 1) + 2(k + 1))/2 = (k + 1)(k + 2)/2 = (k + 1)((k + 1) + 1)/2$$

  - By mathematical induction, we know that $P(n)$ is true for all positive integers n. That is, we have proven that $1 + 2 + \cdots + n = n(n + 1)/2$ holds for all positive integers n.

Exercise (3 mins)

- For any positive integer n, $1 + 3 + 5 + \cdots + (2n - 1) = ?$ Prove it.

- **Principle of Mathematical Induction:** To prove that $P(n)$ is true for all positive integers $n \in \mathbf{Z}^+$, we complete two steps:
  - **Basis step:** prove $P(1)$ is true
  - **Inductive step:** prove $\forall k \in \mathbf{Z}^+, P(k) \to P(k + 1)$ is true

    * *"P(k) is true" is called the inductive hypothesis (IH) 归纳假设*

Exercise *(3 mins)*

- For any positive integer n, $1 + 3 + 5 + \cdots + (2n - 1) = ?$ Prove it.
- Guess $1 + 3 + 5 + \cdots + (2n - 1) = n^2$
  - $1 + 3 = 4,\ 1 + 3 + 5 = 9, \ldots$
- Proof by induction: (Let $P(n)$ be $1 + 3 + 5 + \cdots + (2n - 1) = n^2$.)
  - **Basis step:** $P(1)$ is true, because $1 = 1^2$.
  - **Inductive step:** From the inductive hypothesis, i.e., $P(k)$ is true for an arbitrary positive integer k, we need to show that $P(k + 1)$ is true, i.e., $1 + 3 + \cdots + 2(k + 1) - 1 = (k + 1)^2$.

$$1 + 3 + \cdots + 2k - 1 + 2(k + 1) - 1 = k^2 + 2(k + 1) - 1$$
$$= k^2 + 2k + 2 - 1 = k^2 + 2k + 1 = (k + 1)^2$$

  - By mathematical induction, we know that $P(n)$ is true for all positive integers n. That is, we proved that $1 + 3 + 5 + \cdots + (2n - 1) = n^2$ for all positive integers n.

SUSTech

Exercise *(3 mins)*

- Prove that for any integer $n \geq 2$, $2^{n+1} \geq n^2 + 3$

- **Principle of Mathematical Induction:** To prove that $P(n)$ is true for all positive integers $n \in \boldsymbol{Z}^+$, we complete two steps:
  - **Basis step:** prove $P(1)$ is true
  - **Inductive step:** prove $\forall k \in \boldsymbol{Z}^+, P(k) \rightarrow P(k + 1)$ is true
    ** "P(k) is true" is called the inductive hypothesis (IH)* 归纳假设

- **Well-Ordering Principle**: every nonempty subset of $\boldsymbol{Z}^+$ has a least/minimum element. ** this is an axiom* 公理
  - This also means mathematical induction can be generalized from $\boldsymbol{Z}^+$ to any well-ordered set S, e.g., $\boldsymbol{N}$, $\{n \in \boldsymbol{Z} \mid n \geq b\}$, etc.

Exercise (3 mins)

- Prove that for any integer $n \geq 2$, $2^{n+1} \geq n^2 + 3$
- Proof by induction:
  - Let $P(n)$ be $2^{n+1} \geq n^2 + 3$.
  - **Basis step:** $P(2)$ is true, because $2^{2+1} = 8 \geq 7 = 2^2 + 3$.
  - **Inductive step:** From the inductive hypothesis, i.e., $P(k)$ is true for an arbitrary integer $k \geq 2$, we need to show that $P(k + 1)$ is true:

$$\begin{aligned} 2^{(k+1)+1} = 2 \cdot 2^{k+1} \geq 2(k^2 + 3) = 2k^2 + 6 &= (k + 1)^2 - 2k - 1 + k^2 + 6 \\ &= (k + 1)^2 + (k - 1)^2 + 4 \geq (k + 1)^2 + 3 \end{aligned}$$

  - By mathematical induction, $P(n)$ is true for all integers $n \geq 2$.

SUSTech

Another Form of Induction

- Consider another form of mathematical induction as follows:
  - First suppose that we have a proof that $P(1)$ is true.
  - Next suppose that we have a proof that
    $\forall k \geq 1, P(1) \wedge P(2) \wedge \cdots \wedge P(k) \rightarrow P(k+1)$
  - Then,
    $P(1) \rightarrow P(2)$
    $P(1) \wedge P(2) \rightarrow P(3)$
    $P(1) \wedge P(2) \wedge P(3) \rightarrow P(4)$
    …
  - Iterating gives us a proof of $P(n)$ for all n.

Strong Induction

- **Second principle of mathematical induction:** To prove that $P(n)$ is true for all positive integers n, we complete two steps:
  - **Basis step:** prove $P(1)$ is true
  - **Inductive step:** prove $\forall k \in \mathbf{Z}^+, P(1) \wedge \cdots \wedge P(k) \rightarrow P(k+1)$ is true
    * *here "$P(1) \wedge P(2) \wedge \cdots \wedge P(k)$ is true" is the inductive hypothesis (IH)*

- This is called **strong induction** or **complete induction**, while the previous principle is called **weak** or **incomplete** induction.

- In practice, strong induction is often easier to apply than its weak form, because the inductive hypothesis is stronger.

- However, these two forms of induction are actually equivalent.
  - *the proof is left as an assignment problem*

Example

- **Theorem:** Every positive integer is a power of a prime or the product of powers of primes.

- Proof by strong induction:
  - *P(n)*: "*n* is a power of a prime or the product of powers of primes"
  - **Basis step:** *P(1)* is true, as *1* is a power of a prime number, $1 = 2^0$.
  - **Inductive step:**

    Inductive hypothesis: *P(m)* is true for every *m* that $1 \leq m \leq k$, i.e., *m* is a power of a prime or a product of powers of primes.

    If $k + 1$ is a prime power, $P(k + 1)$ is true. Otherwise, $k + 1$ must be a composite, i.e., a product of two smaller positive integers, each of which is, by the inductive hypothesis, a power of a prime or the product of powers of primes. Therefore, $P(k + 1)$ is true.
  - Finally, by strong induction, *P(n)* is true for all positive integers.

Mathematical Induction Summary

- A typical proof by induction, showing that $P(n)$ is true for all integers $n \geq b$, consists of three steps:
  - **Basis step:** prove $P(b)$ is true
  - **Inductive step:** prove one of the following

    $\forall k \geq b, P(k) \rightarrow P(k+1)$ is true ***OR***

    $\forall k \geq b, P(b) \wedge \cdots \wedge P(k) \rightarrow P(k+1)$ is true
  - **Conclusion:** based on the (second) principle of mathematical induction, we conclude that $P(n)$ is true for all $n \geq b$.

- The assumption "$P(k)$ is true" ***OR*** "$P(1) \wedge P(2) \wedge \cdots \wedge P(k)$ *is true*" is called the inductive hypothesis (IH).
  - IH is used as premises to prove the conclusion "$P(k+1)$ *is true*".

Recursion

Recursion

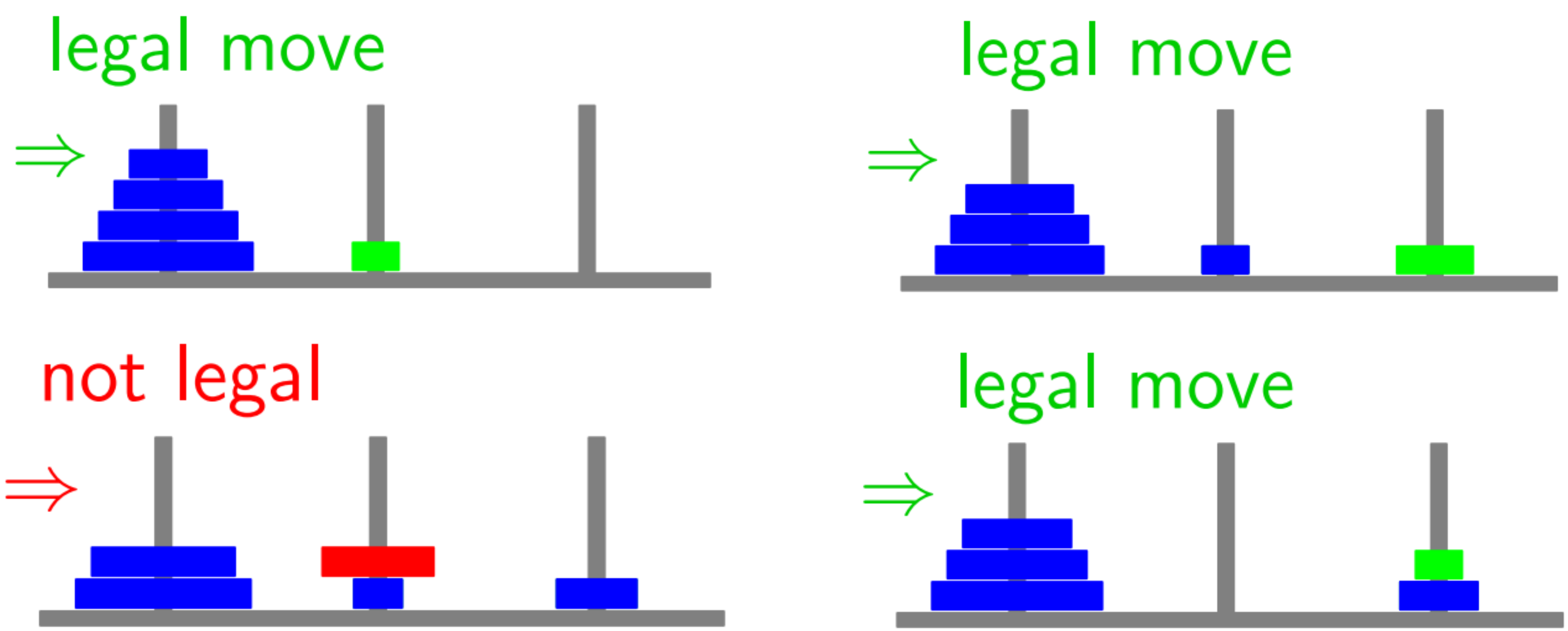
- **Recursion:** a method of solving a computational problem where its solution depends on solutions to smaller instances of the same problem.
- Recursive computer programs or algorithms often lead to inductive analysis.
- A classical example of recursion is the Towers of Hanoi puzzle.

Example: Towers of Hanoi Puzzle

- **Problem:** Find an efficient way to move all of the disks from one peg to another, using only legal moves.
  - Consider *3* pegs and *n* disks of different sizes.
  - What is a legal move?

Example: Towers of Hanoi Puzzle

- **Problem:** Find an efficient way to move all of the disks from one peg to another, using only legal moves.
  - Consider *3* pegs and *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.



SUSTech

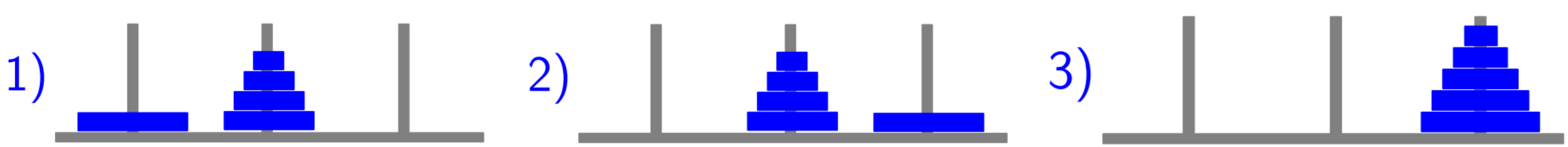
Towers of Hanoi: Solution

- **Problem:** Find an efficient way to move all of the disks from one peg to another, using only legal moves.

- **Solution** by recursion: * *very similar to mathematical induction*
  - **Basis step:** If $n = 1$, moving one disk from one to another is easy.

  - **Recursive step:** If $n > 1$, we need 3 steps (e.g., to move n disks from peg 1 to peg 3):

n – 1 from 1 to 2   *largest from 1 to 3*   *n – 1 from 2 to 3*



SUSTech

Towers of Hanoi: Solution

- **Problem:** Find an efficient way to move all of the disks from one peg to another, using only legal moves.
- **Solution** by recursion: * *very similar to mathematical induction*

```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);
        }
    }

```

move n disks from peg a to peg c using peg b

1. move n – 1 disks from a to b using c

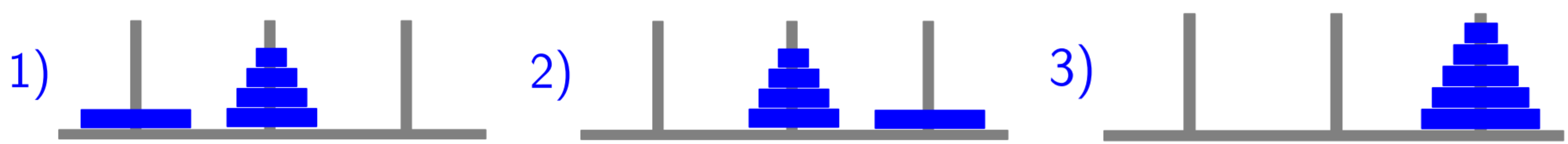
2. move the largest disk from a to c

3. move n – 1 disks from b to c using a

Towers of Hanoi: Correctness

- **Problem:** Find an efficient way to move all of the disks from one peg to another, using only legal moves.

- **Proof of correctness** (of solution) by induction:
  - Let *P(n)* be the predicate that the solution is correct for *n*.
  - **Basis step:** *P(1)* is obviously true, i.e., the solution is correct when there is only one disk.
  - **Inductive step:** From the inductive hypothesis, i.e., *P(k)* is true for an arbitrary positive integer *k*, we need to show that *P(k + 1)* is true. That is, if our solution works for *k* disks, then we can build a correct solution for *k + 1* disks, which is true by the recursive step:



  - By mathematical induction, *P(n)* is true for all positive integer *n*.

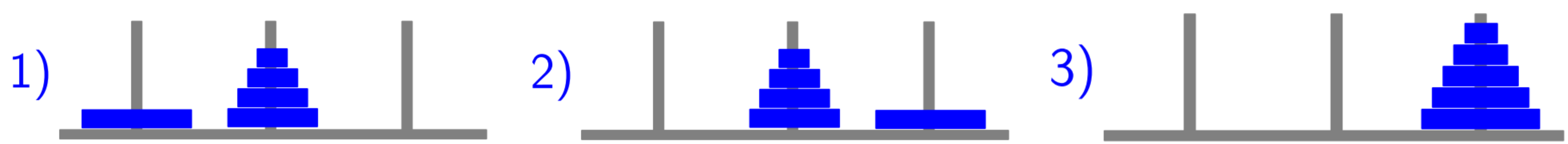
Towers of Hanoi: Running Time

- **Problem:** Find an efficient way to move all of the disks from one peg to another, using only legal moves.

- **Solving the running time:** * *number of disk moves* $M(n) = ?$
  - **Basis step:** If $n = 1$, moving one disk from one to another is easy.

$M(1) = 1$

  - **Recursive step:** If $n > 1$, we need three steps:



$$M(n) = 2M(n-1) + 1 \text{ for } n > 1$$

Towers of Hanoi: Running Time

- **Problem:** Find an efficient way to move all of the disks from one peg to another, using only legal moves.

- **Solving the running time:** * *number of disk moves $M(n) = ?$*

  $M(1) = 1 \qquad M(n) = 2M(n-1) + 1$ for $n > 1$

  - Iterating the above function gives:

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

  - We can guess that $M(n) = 2^n - 1$ and prove it by induction:

    Let $P(n)$ denote the above equation.

    Basis step: $P(1)$ is true, because $M(1) = 1 = 2^1 - 1$.

    Inductive step: Assume $P(k)$ is true for $k \geq 1$, i.e., $M(k) = 2^k - 1$.

    Then $P(k + 1)$ is true: $M(k + 1) = 2M(k) + 1 = 2(2^k - 1) + 1 = 2^{k+1} - 1$

    By mathematical induction, $P(n)$ is true for all positive n.

Towers of Hanoi: Summary

- **Problem:** Find an efficient way to move all of the disks from one peg to another, using only legal moves.
- **Solution** by recursion: * *very similar to mathematical induction*
  - **Basis step:** If $n = 1$, moving one disk from one to another is easy.
  - **Recursive step:** If $n > 1$, we need 3 steps (e.g., to move n disks from peg 1 to peg 3):

    $n - 1$ from 1 to 2 *largest from 1 to 3* *$n - 1$ from 2 to 3*
- Note that we applied induction twice:
  - first time to prove correctness of the solution
  - second time to derive the closed-form running time

Recurrence Relations

Recurrence Relations

- A **recurrence relation** or **recurrence** for a function defined on the set of integers ≥ b tells us how to compute the n-th value from some or all of the first $n - 1$ values.
- To completely specify a function defined by a recurrence, we have to also give the initial condition(s) (as known as the base case(s)) for the recurrence.
- Example: running time for Towers of Hanoi

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

Example

- Let $S(n)$ be the number of subsets of a set of size n. We already learned that $S(n) = 2^n$, but now let us think about this recursively:
  - Consider the subsets of $\{1, 2, 3\}$:

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

  - The top 4 subsets are exactly the subsets of $\{1, 2\}$, while the bottom 4 subsets 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 adding 3 to a subset of $\{1, 2\}$.
  - This suggests that the recurrence relation for the number of subsets of an n-element set $\{1, 2, \ldots, n\}$ is

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

SUSTech

Example

- Let $S(n)$ be the number of subsets of a set of size n.
- **Proof of correctness** for the recurrence:

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

  - It is clear that $S(0) = 1$, as only the empty set $\varnothing$ contains 0 element.
  - For $n \geq 1$, w.l.o.g., consider the subsets of $\{1, 2, \ldots, n\}$. They can be partitioned according to whether or not they contain the element n.

    Each subset S containing n can be uniquely constructed from the subset $S - \{n\}$ not containing n. Each subset S not containing n can be uniquely constructed from the subset $S \cup \{n\}$ containing n. Therefore, the number of subsets containing n is equal to the number of subsets not containing n, so we have $S(n) = 2S(n - 1)$.

- **Proof of the closed form:** $S(n) = 2^n$ * *left as an exercise*

Iterating a Recurrence

- Let $T(n) = rT(n - 1) + a$, where r and a are constants.
- Find a recurrence relation that expresses

  $T(n)$ in terms of $T(n - 2)$

  $T(n)$ in terms of $T(n - 3)$

  $T(n)$ in terms of $T(n - 4)$

  ...
- Can we generalize this to find a closed-form solution to $T(n)$?

SUSTech

Iterating a Recurrence

- Note that $T(n) = rT(n-1) + a$ implies that for any $n - i > 0$:

$$T(n-i) = rT((n-i)-1)) + a$$

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

SUSTech

Iterating a Recurrence

- The technique we used to guess the closed formula of *T(n)* is called **iteration**, because we iteratively use the recurrence.
- The approach we just used is called **backward substitution**, because we began with *T(n)* and iterated to express it in terms of falling terms of the sequence until we found it in terms of *T(0)*.
- The other similar approach is called **forward substitution**, which iterates from *T(0)* to *T(n)*.
  - E.g., $T(n) = rT(n - 1) + a$, where r and a are constants.

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

  - This leads to the same guess: $T(n) = r^n T(0) + a \sum_{i=0}^{n-1} r^i$

SUSTech

Closed Formula of Recurrences

- **Theorem:** If $T(n) = rT(n-1) + a$, $T(0) = b$, and $r \neq 1$, then for all non-negative integers n, we have:

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

- Proof by induction:
  - **Basis step:** The formula is true for $n = 0$: $T(0) = r^0 b + a\frac{1-r^0}{1-r} = b.$
  - **Inductive step:**

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

Closed Formula of Recurrences

- **Theorem:** If $T(n) = rT(n-1) + a$, $T(0) = b$, and $r \neq 1$, then for all non-negative integers n, we have:

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

- Example: $T(n) = 3T(n-1) + 2,\ T(0) = 5$
  - Plugging $r = 3,\ a = 2,\ b = 5$ in the formula, we have:

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

Exercise (3 mins)

- Solve $T(n) = rT(n-1) + g(n)$ with $T(0) = a$ and constant $r \neq 0$.
  Hint: write T(n) in terms of r, n, T(0) and g(i).

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

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

SUSTech

Exercise (3 mins)

- Solve $T(n) = rT(n-1) + g(n)$ with $T(0) = a$ and constant $r \neq 0$.
  Hint: write T(n) in terms of r, n, T(0) and g(i).

- Solution:

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

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

First-Order Linear Recurrences

- **Theorem:** For any constants a and $r \neq 0$, and any function g defined on positive integers, the solution to the recurrence

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

is

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

  - *the proof (by induction) is left as an exercise*

- A recurrence relation of the form $T(n) = f(n)T(n-1) + g(n)$ is called a **first-order linear** recurrence.
  - **first order:** $T(n)$ depends upon going back one step, i.e., $T(n-1)$
    e.g., $T(n) = T(n-1) + 2T(n-2)$ is a second-order recurrence
  - **linear:** the $T(n-1)$ only appears to the first power.
    e.g., $T(n) = (T(n-1))^2 + 3$ is a non-linear first-order recurrence

Exercise (3 mins)

- Solve $T(n) = 4T(n-1) + 2^n$ $(n > 0)$ with $T(0) = 6$
  Hint: write T(n) in terms of 4^n and 2^n

- **Theorem:** For any constants a and $r \neq 0$, and any function g defined on positive integers, the solution to the recurrence

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

is

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

SUSTech

Exercise (3 mins)

- Solve $T(n) = 4T(n-1) + 2^n$ $(n > 0)$ with $T(0) = 6$
  Hint: write $T(n)$ in terms of 4^n and 2^n

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

Divide-and-Conquer Recurrences

Divide and Conquer

- **Divide and conquer (D&C):** recursively breaks down a problem into two or more sub-problems of the same or related type, until these become simple enough to be solved directly; the solutions to the sub-problems are then combined to give a solution to the original problem.

- Divide-and-conquer recurrence relations are usually of the form:

$$T(n) = \begin{cases} \text{something given} & \text{if } n \leq n_0 \\ r \cdot T(n/m) + g(n) & \text{if } n > n_0 \end{cases}$$

Example: Binary Search

- **Problem:** Someone has chosen an integer x between 1 and n. We need to find this secret x.

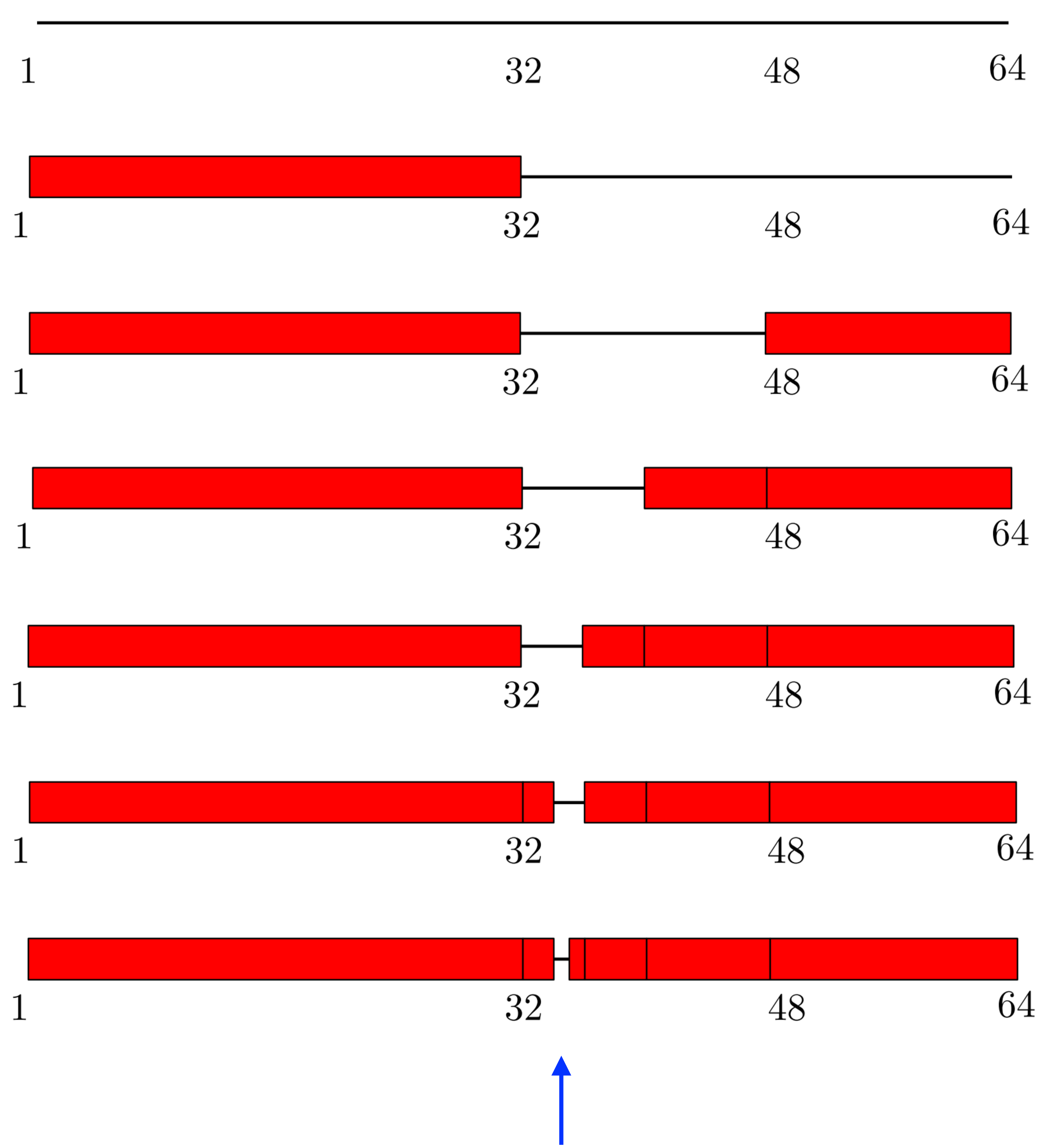
- We only need to ask two types of questions:
  - Is x greater than k?
  - Is x equal to k?

- **Strategy:** We first always ask greater than questions, at each step halving our search range, until the range contains only one number, then we ask a final equal to question.

Binary Search: Demonstration

- Example: $n = 64$, $x = 35$

$x > 32?$	*Yes*
$x > 48?$	*No*
$x > 40?$	*No*
$x > 36?$	*No*
$x > 34?$	*Yes*
$x > 35?$	*No*
$x = 35?$	*Yes*



Binary Search: Running Time

- **D&C:** Each guess reduces the size of the problem to only half as big, then we can (recursively) conquer this smaller problem.

- When *n is a power of 2*, the number of comparisons *T(n)* in a binary search on *{1, 2, …, n}* satisfies

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

- Proof of correctness (by strong induction) of the running time:
  - **Basis step** *(n = 1)*: only one “equal to” comparison is needed
  - **Inductive step** *(n ≥ 2)*: one “great than” comparison + time to perform binary search on the remaining *n/2* terms

SUSTech

Binary Search: Running Time

- **Solving the recurrence:**

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

- Solve it by **iteration** (e.g., **backward substitution**):

$$\begin{aligned}
T(n) &= T\left(\tfrac{n}{2}\right) + 1 && = \left(T\left(\tfrac{n}{2^2}\right) + 1\right) + 1 \\
&= T\left(\tfrac{n}{2^2}\right) + 2 && = \left(T\left(\tfrac{n}{2^3}\right) + 1\right) + 2 \\
&= T\left(\tfrac{n}{2^3}\right) + 3 \\
&\quad\vdots \qquad \vdots \\
&= T\left(\tfrac{n}{2^i}\right) + i \\
&\quad\vdots \qquad \vdots \\
&= T\left(\tfrac{n}{2^{\log_2 n}}\right) + \log_2 n = 1 + \log_2 n
\end{aligned}$$

terminate when $i = log_2\, n$

Binary Search: Summary

- **Problem:** Someone has chosen an integer x between 1 and n. We need to find the chosen x.
- **D&C:** Each guess reduces the size of the problem to only half as big, then we can (recursively) conquer this smaller problem.
- **Running time:** When n is a power of 2, the number of comparisons $T(n)$ in a binary search on $\{1, 2, \ldots, n\}$ satisfies

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

- **Solving the recurrence** by iteration: $T(n) = 1 + \log_2 n$
  - **Note:** Technically, we still need to use induction to prove the above closed formula is correct. Practically, we almost never explicitly perform this step, since it is obvious how the induction would work.

Iterating D&C Recurrences

- **Example 1:**

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

- This corresponds to solving a problem of size *n*, by either

  (i) solving *2* subproblems of size *n/2*

  (ii) doing *n* units of additional work

  or using *T(1)* work for the case of *n = 1*.

  ** this is exactly how Merge Sort (from an algorithm course) works*

- How to solve the recurrence?

Iterating D&C Recurrences

- **Example 1:**

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

- Solve it by **iteration**: * *assume n is a power of 2*

$$\begin{aligned}
T(n) = 2T\left(\tfrac{n}{2}\right) + n \quad &= 2\left(2T\left(\tfrac{n}{4}\right) + \tfrac{n}{2}\right) + n \\
= 4T\left(\tfrac{n}{4}\right) + 2n \quad &= 4\left(2T\left(\tfrac{n}{8}\right) + \tfrac{n}{4}\right) + 2n \\
= 8T\left(\tfrac{n}{8}\right) + 3n & \\
\vdots \qquad \vdots & \\
= 2^i T\left(\tfrac{n}{2^i}\right) + in & \\
\vdots \qquad \vdots & \\
= 2^{\log_2 n} T\left(\tfrac{n}{2^{\log_2 n}}\right) + (\log_2 n)n &= nT(1) + n\log_2 n
\end{aligned}$$

Iterating D&C Recurrences

- **Example 2:**

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

- Solve it by **iteration**: * *assume n is a power of 2*

$$\begin{aligned}
T(n) &= T\left(\frac{n}{2}\right) + n \\
&= T\left(\frac{n}{2^2}\right) + \frac{n}{2} + n \\
&= T\left(\frac{n}{2^3}\right) + \frac{n}{2^2} + \frac{n}{2} + n \\
&\quad \vdots \qquad \vdots \\
&= T\left(\frac{n}{2^i}\right) + \frac{n}{2^{i-1}} + \cdots + \frac{n}{2^2} + \frac{n}{2} + n \\
&\quad \vdots \qquad \vdots \\
&= T\left(\frac{n}{2^{\log_2 n}}\right) + \frac{n}{2^{\log_2 n - 1}} + \cdots + \frac{n}{2^2} + \frac{n}{2} + n \\
&= 1 + 2 + 2^2 + \cdots + \frac{n}{2^2} + \frac{n}{2} + n = 2n - 1
\end{aligned}$$

Exercise (3 mins)

- Solve this recurrence by iteration:

$$T(n) = \begin{cases} 1 & \text{if } n < 3 \\ 3T(n/3) + n & \text{if } n \geq 3 \end{cases}$$

- **Example 2:**

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

- Solve it by iteration:

$$\begin{aligned} T(n) &= T\left(\tfrac{n}{2}\right) + n \\ &= T\left(\tfrac{n}{2^2}\right) + \tfrac{n}{2} + n \\ &= T\left(\tfrac{n}{2^3}\right) + \tfrac{n}{2^2} + \tfrac{n}{2} + n \\ &\ \ \vdots \qquad \vdots \\ &= T\left(\tfrac{n}{2^i}\right) + \tfrac{n}{2^{i-1}} + \cdots + \tfrac{n}{2^2} + \tfrac{n}{2} + n \\ &\ \ \vdots \qquad \vdots \\ &= T\left(\tfrac{n}{2^{\log_2 n}}\right) + \tfrac{n}{2^{\log_2 n - 1}} + \cdots + \tfrac{n}{2^2} + \tfrac{n}{2} + n \\ &= 1 + 2 + 2^2 + \cdots + \tfrac{n}{2^2} + \tfrac{n}{2} + n = 2n - 1 \end{aligned}$$

Exercise (3 mins)

- Solve this recurrence by iteration:

$$T(n) = \begin{cases} 1 & \text{if } n < 3 \\ 3T(n/3) + n & \text{if } n \geq 3 \end{cases}$$

- Solution:

$$\begin{aligned} T(n) = 3T\left(\frac{n}{3}\right) + n \quad & = 3\left(3T\left(\frac{n}{3^2}\right) + \frac{n}{3}\right) + n \\ = 3^2T\left(\frac{n}{3^2}\right) + 2n \quad & = 3^2\left(3T\left(\frac{n}{3^3}\right) + \frac{n}{3^2}\right) + 2n \\ = 3^3T\left(\frac{n}{3^3}\right) + 3n & \\ \vdots \quad \vdots & \\ = 3^iT\left(\frac{n}{3^i}\right) + in & \\ \vdots \quad \vdots & \\ = 3^{\log_3 n}T\left(\frac{n}{3^{\log_3 n}}\right) + n\log_3 n \quad & = n + n\log_3 n \end{aligned}$$

Iterating D&C Recurrences

- **Example 3:**

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 4T(n/2) + n & \text{if } n \geq 2 \end{cases}$$

- Solve it by **iteration**: ** assume n is a power of 2*

$$\begin{aligned}
T(n) &= 4T\left(\frac{n}{2}\right) + n &&= 4\left(4T\left(\frac{n}{2^2}\right) + \frac{n}{2}\right) + n \\
&= 4^2 T\left(\frac{n}{2^2}\right) + \frac{4}{2}n + n &&= 4^2\left(4T\left(\frac{n}{2^3}\right) + \frac{n}{2^2}\right) + \frac{4}{2}n + n \\
&= 4^3 T\left(\frac{n}{2^3}\right) + \frac{4^2}{2^2}n + \frac{4}{2}n + n \\
&\vdots \\
&= 4^i T\left(\frac{n}{2^i}\right) + \frac{4^{i-1}}{2^{i-1}}n + \cdots + \frac{4^2}{2^2}n + n \\
&\vdots \\
&= 4^{\log_2 n} T\left(\frac{n}{2^{\log_2 n}}\right) + \frac{4^{\log_2 n - 1}}{2^{\log_2 n - 1}}n + \cdots + \frac{4}{2}n + n = 2n^2 - n
\end{aligned}$$

SUSTech

Three Different Behaviors

- Compare the iteration for the following recurrences ($T(1) = 1$):
  - $T(n) = T(n / 2) + n$ $\quad T(n) = 2n - 1 = \Theta(n)$
  - $T(n) = 2T(n / 2) + n$ $\quad T(n) = n + n \log_2 n = \Theta(n \log n)$
  - $T(n) = 4T(n / 2) + n$ $\quad T(n) = 2n^2 - n = \Theta(n^2)$
  - All three recurrences iterate $\log_2 n$ times. The size of subproblem in next iteration is half the size in the preceding iteration level.

- **Theorem:** Consider a recurrence relation $T(n) = aT(n / 2) + n$ whenever $n = 2^k$, where $a \geq 1$ and $T(1) = \Theta(1)$. Then we have the following Θ bounds on the solution:
  - If $1 \leq a < 2$, then $T(n) = \Theta(n)$. ** left as an assignment problem*
  - If $a = 2$, then $T(n) = \Theta(n \log n)$. ** already proved in Example 1*
  - If $a > 2$, then $T(n) = \Theta(n^{\log_2 a})$. ** now let us prove this*

SUSTech

Three Different Behaviors

- For $T(n) = aT(n / 2) + n$ and $n = 2^k$, where $a \geq 1$ and $T(1) = \Theta(1)$. Prove that "If $a > 2$, then $T(n) = \Theta(n^{\log_2 a})$."
- Iterating the recurrence as in **Example 3** gives:

$$T(n) = a^i T\left(\frac{n}{2^i}\right) + \left(\frac{a^{i-1}}{2^{i-1}} + \frac{a^{i-2}}{2^{i-2}} + \cdots \frac{a}{2} + 1\right) n$$

$$= a^{\log_2 n} T(1) + n \sum_{i=0}^{\log_2 n - 1} \left(\frac{a}{2}\right)^i$$

- For $r > 1$, the sum $1 + r + r^2 + \cdots + r^n$ grows as fast as its largest item r^n. * *think why?* Therefore, for $a/2 > 1$, we have:

$$n \sum_{i=0}^{\log_2 n - 1} \left(\frac{a}{2}\right)^i = n\Theta\left((a/2)^{\log_2 n - 1}\right)$$

Three Different Behaviors

- For $T(n) = aT(n / 2) + n$ and $n = 2^k$, where $a \geq 1$ and $T(1) = \Theta(1)$. Prove that "If $a > 2$, then $T(n) = \Theta(n^{\log_2 a})$."

- Iterating $T(n) = aT(n/2) + n$ gives:

$$T(n) = a^{\log_2 n} T(1) + n\Theta((a/2)^{\log_2 n - 1})$$

- Now, for $a > 2$, we have:

$$n\left(\frac{a}{2}\right)^{\log_2 n - 1} = \frac{2}{a} \cdot \frac{n \cdot a^{\log_2 n}}{2^{\log_2 n}} = \frac{2}{a} \cdot \frac{n \cdot a^{\log_2 n}}{n} = \frac{2}{a} \cdot a^{\log_2 n}$$

$$\boxed{a^{\log_2 n} = \left(2^{\log_2 a}\right)^{\log_2 n} = \left(2^{\log_2 n}\right)^{\log_2 a} = n^{\log_2 a}}$$

- Finally: $T(n) = a^{\log_2 n} T(1) + n\Theta((a/2)^{\log_2 n - 1}) = \Theta\left(n^{\log_2 a}\right)$

Master Theorem

- **Theorem:** Consider a recurrence relation $T(n) = aT(n / 2) + n$ whenever $n = 2^k$, where $a \geq 1$ and $T(1) = \Theta(1)$. Then we have the following big Θ bounds on the solution:
  - If $1 \leq a < 2$, then $T(n) = \Theta(n)$. ** this proof is left as an exercise*
  - If $a = 2$, then $T(n) = \Theta(n \log n)$. ** already proved in Example 2*
  - If $a > 2$, then $T(n) = \Theta(n^{\log_2 a})$. ** just proved*

- **Master Theorem:** For a recurrence relation $T(n) = aT(n / b) + cn^d$ whenever $n = b^k$, where $a \geq 1, c > 0, d \geq 0,$ integer $b \geq 2,$ and $T(1) = \Theta(1)$, we have the following big Θ bounds on the solution:
  - If $1 \leq a < b^d$, then $T(n) = \Theta(n^d)$.
  - If $a = b^d$, then $T(n) = \Theta(n^d \log n)$.
  - If $a > b^d$, then $T(n) = \Theta(n^{\log_b a})$.

07 Counting

To be continued…

Assignment 4

- Deadline for Assignment 4: Nov 29