Department of Computer Science and Engineering
计算机科学与工程系

CS215 DISCRETE MATH

Dr. Qi Wang

Department of Computer Science and Engineering
Office: Room413, CoE South Tower
Email: wangqi@sustech.edu.cn

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

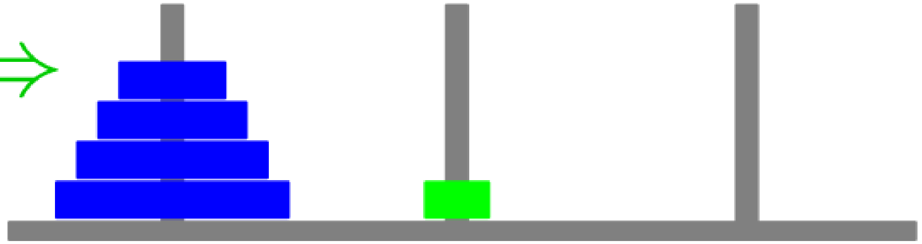
Towers of Hanoi

- 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 an (efficient) way to move all of the disks from one peg to another

Towers of Hanoi

Towers of Hanoi

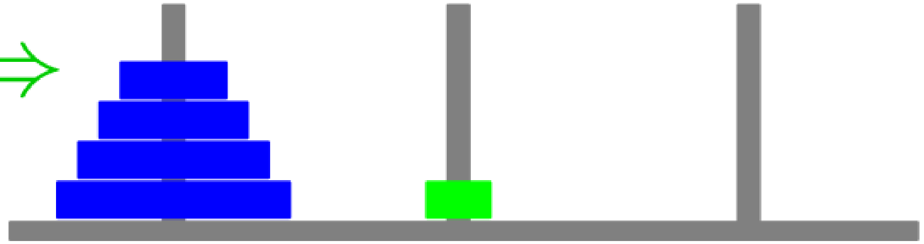
legal move

$\Rightarrow$ 

Towers of Hanoi

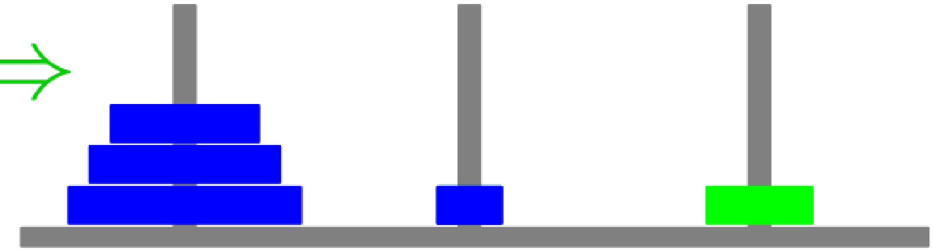
legal move

$\Rightarrow$



legal move

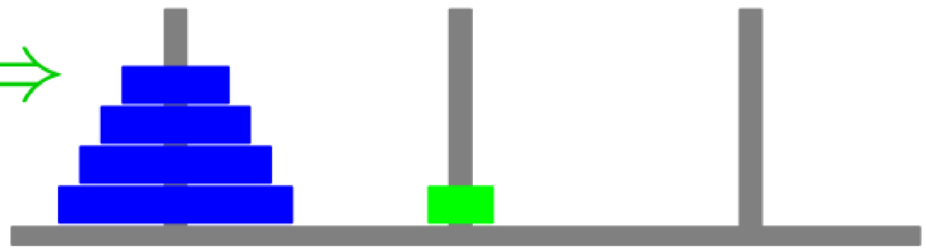
$\Rightarrow$



Towers of Hanoi

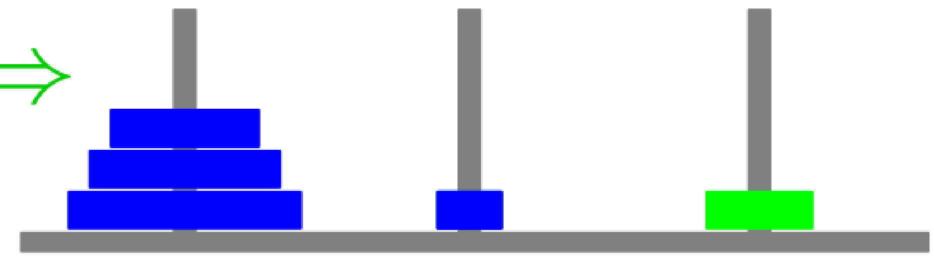
legal move

$\Rightarrow$



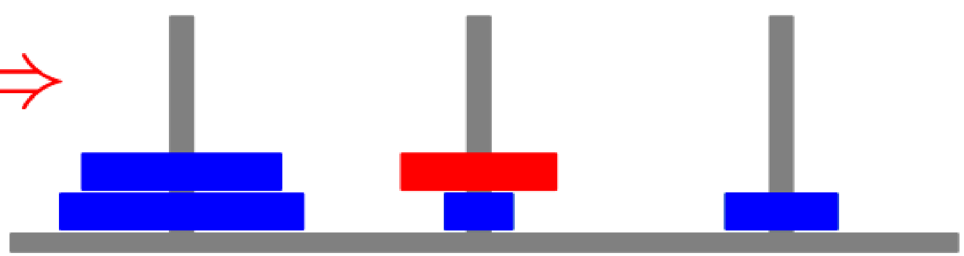
legal move

$\Rightarrow$



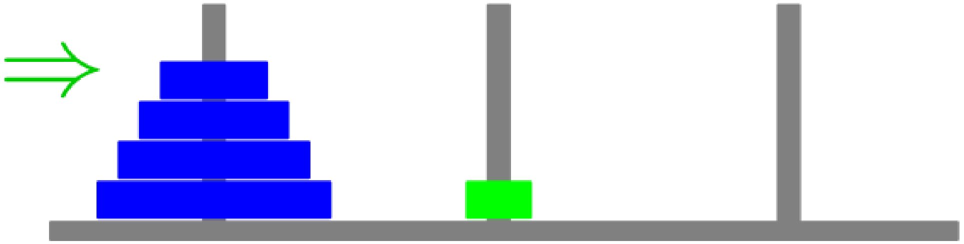
not legal

$\Rightarrow$

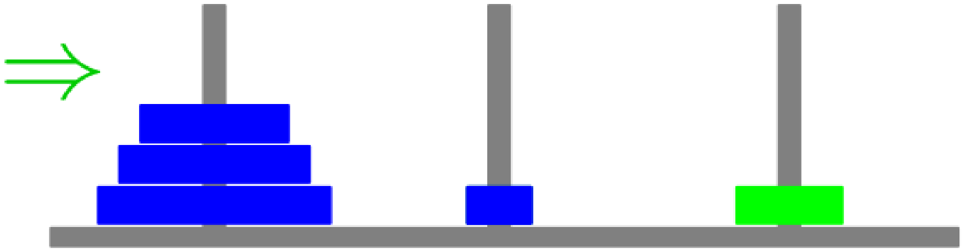


Towers of Hanoi

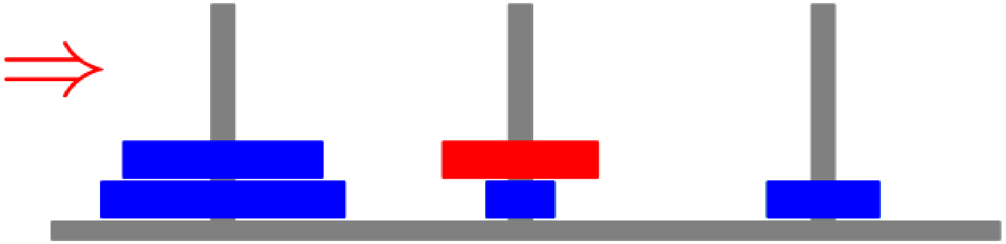
legal move



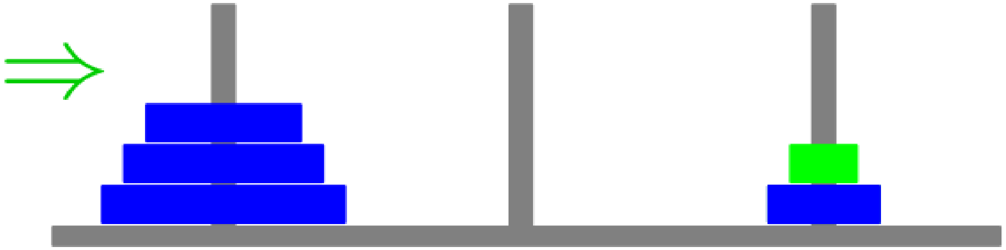
legal move



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

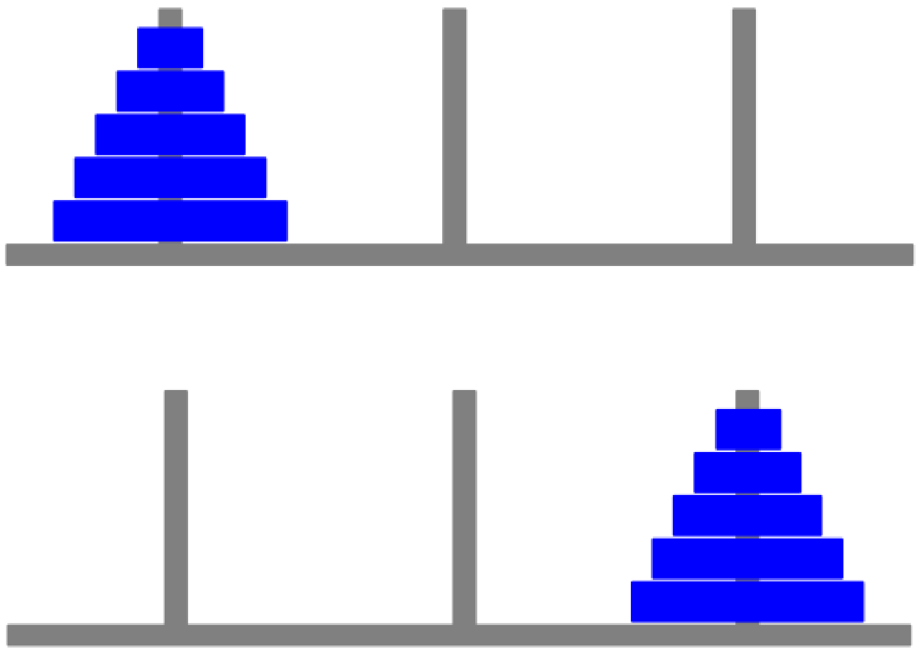
  using only legal moves

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:

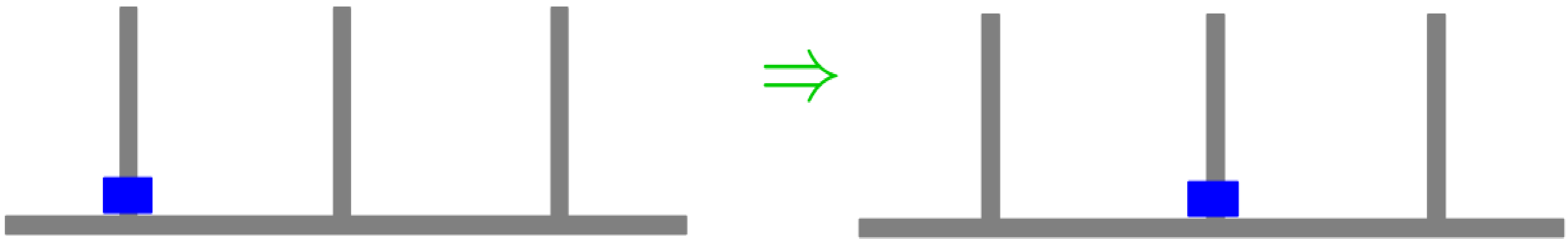
  If $n = 1$, moving one disk from i to j is easy. Just move it.

Towers of Hanoi

- General solution

  Recursion Base:

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



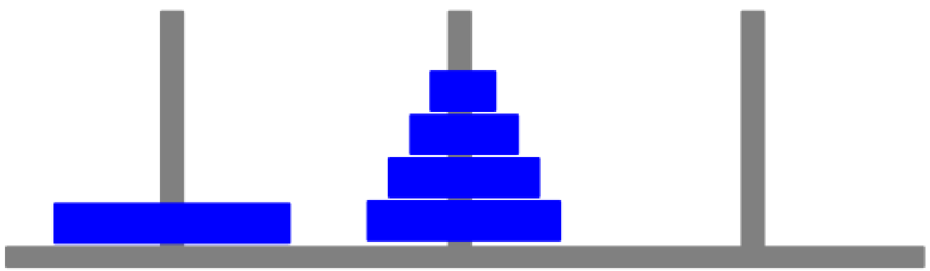
Towers of Hanoi

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

Towers of Hanoi

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

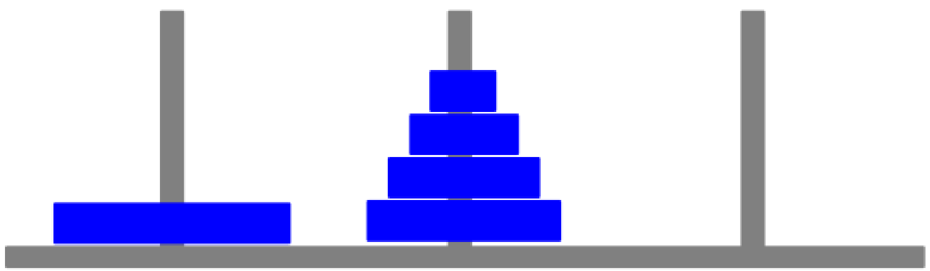
1)



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

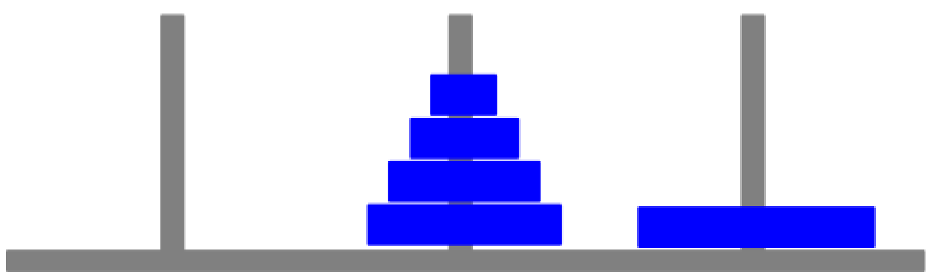
To move $n > 1$ disks from i to j

1)



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

2)

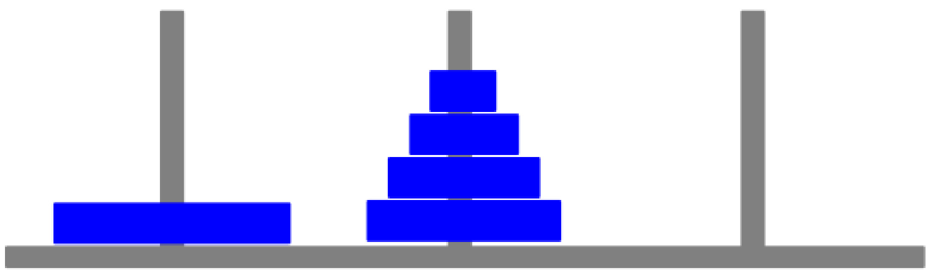


move largest disk from i to j

Towers of Hanoi

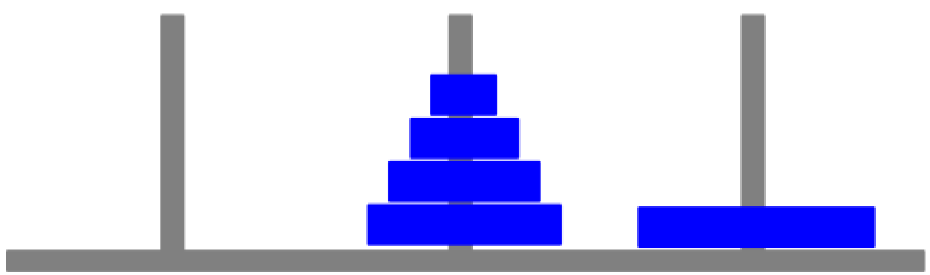
To move $n > 1$ disks from i to j

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 $M(n) = 2^n - 1$.

- 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 $\{1, 2, 3\}$:

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

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

- 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 $\{1, 2\}$.

- 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

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

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

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

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

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

  The subsets of $\{1, 2, \ldots, n\}$ can be partitioned according to whether or not they contain the 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.** of correctness of this recurrence

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

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

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".

  $$T(0) = b$$
  $$T(1) = rT(0) + a = rb + a$$
  $$T(2) = rT(1) + a = r(rb + a) + a = r^2 b + ra + a$$
  $$T(3) = rT(2) + a = r^3 b + r^2 a + ra + a$$

- 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**

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

- **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 $x \neq 1$,

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

Examples

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

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

Growth Rates of Solutions to Recurrences

- Divide and conquer algorithms

- Iterating recurrences

- Three different behaviors

Divide and conquer algorithms

- We just analyzed recurrences of the form

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

Divide and conquer algorithms

- We just analyzed recurrences of the form

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

These corresponded to the analysis of recursive algorithms in which a problem of size n is solved by recursively solving a problem of size $n-1$.

Divide and conquer algorithms

- We just analyzed recurrences of the form

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

These corresponded to the analysis of recursive algorithms in which a problem of size n is solved by recursively solving a problem of size $n-1$.

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

Divide and conquer algorithms

- We just analyzed recurrences of the form

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

These corresponded to the analysis of recursive algorithms in which a problem of size n is solved by recursively solving a problem of size $n-1$.

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

We will now look at recurrences of the form

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

Binary Search

- Someone has chosen a number x between 1 and n. We need to discover x.

Binary Search

- Someone has chosen a number x between 1 and n. We need to discover x.

  We are only allowed to ask two types of questions:

Binary Search

- Someone has chosen a number x between 1 and n. We need to discover x.

  We are only allowed to ask two types of questions:

  - ◇ Is x greater than k?
  - ◇ Is x equal to k?

Binary Search

- Someone has chosen a number x between 1 and n.
  We need to discover x.

  We are only allowed to ask two types of questions:

  - ◇ Is x greater than k?
  - ◇ Is x equal to k?

  Our strategy will be to always ask greater than questions, at each step halving our search range, until the range only contains one number, when we ask a final equal to question.

Binary Search Example

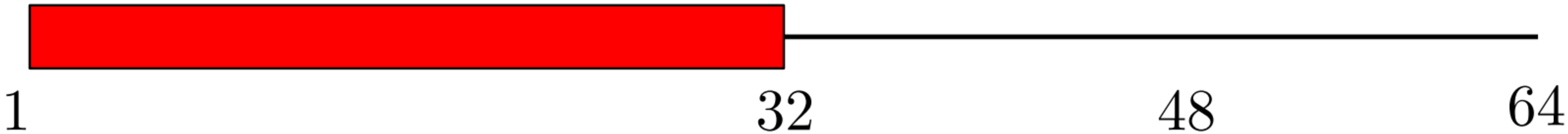
1 32 48 64

Binary Search Example

1 32 48 64

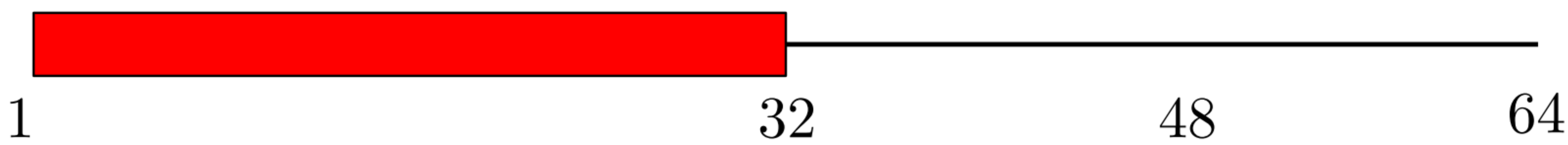
Is $x > 32$?

Binary Search Example



Is $x > 32$? Answer: Yes

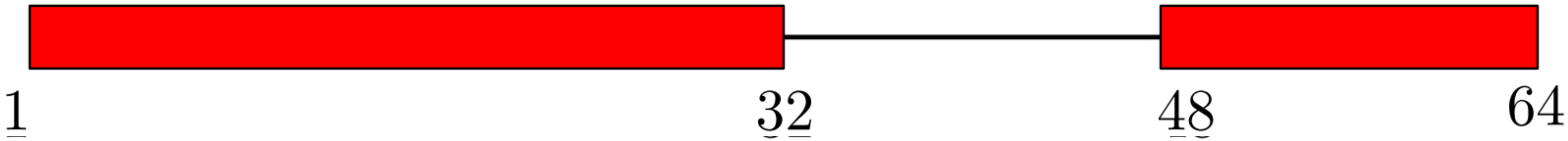
Binary Search Example



Is $x > 32$?   Answer: Yes

Is $x > 48$?

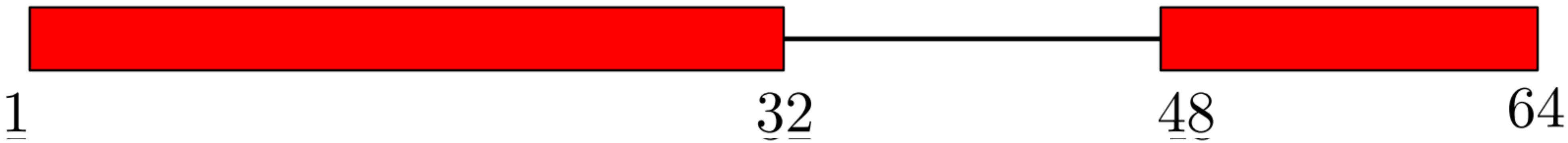
Binary Search Example



Is $x > 32$? Answer: Yes

Is $x > 48$? Answer: No

Binary Search Example

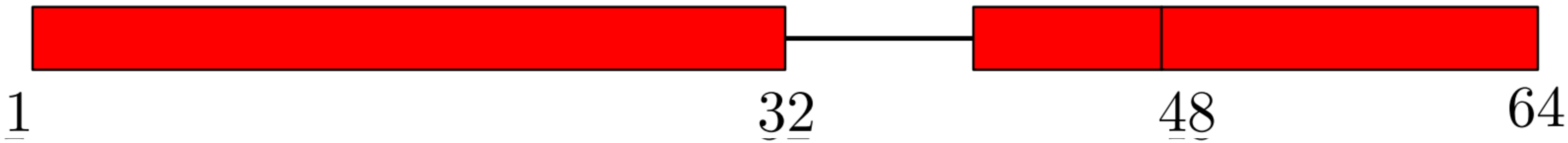


Is $x > 32?$ Answer: Yes

Is $x > 48?$ Answer: No

Is $x > 40?$

Binary Search Example

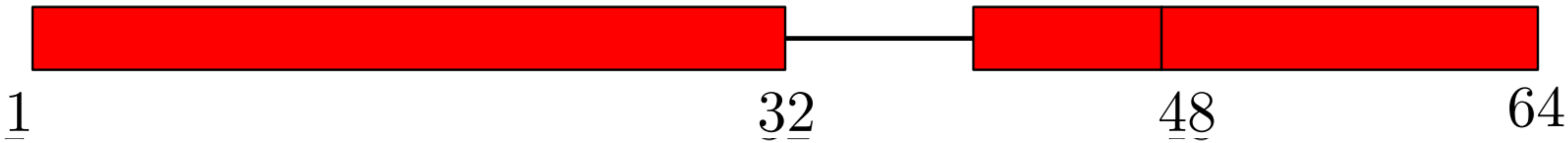


Is $x > 32$? Answer: Yes

Is $x > 48$? Answer: No

Is $x > 40$? Answer: No

Binary Search Example



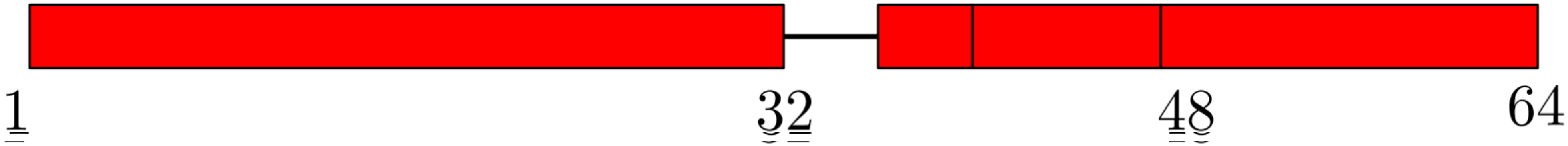
Is $x > 32?$ Answer: Yes

Is $x > 48?$ Answer: No

Is $x > 40?$ Answer: No

Is $x > 36?$

Binary Search Example



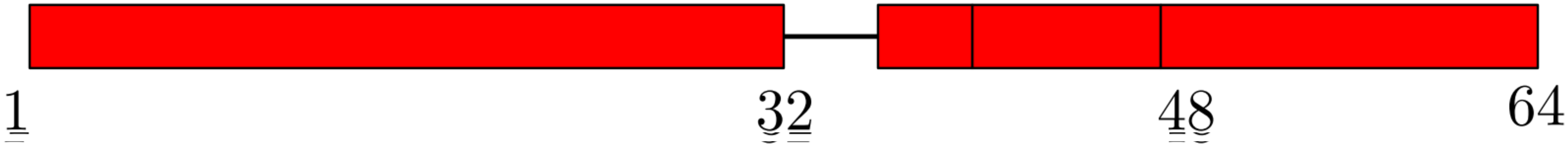
Is $x > 32$? Answer: Yes

Is $x > 48$? Answer: No

Is $x > 40$? Answer: No

Is $x > 36$? Answer: No

Binary Search Example



Is $x > 32$? Answer: Yes

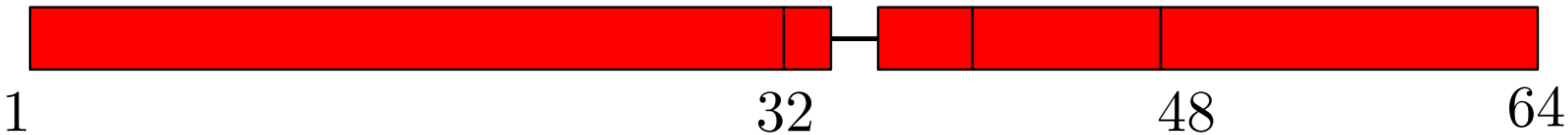
Is $x > 48$? Answer: No

Is $x > 40$? Answer: No

Is $x > 36$? Answer: No

Is $x > 34$?

Binary Search Example



Is $x > 32$? Answer: Yes

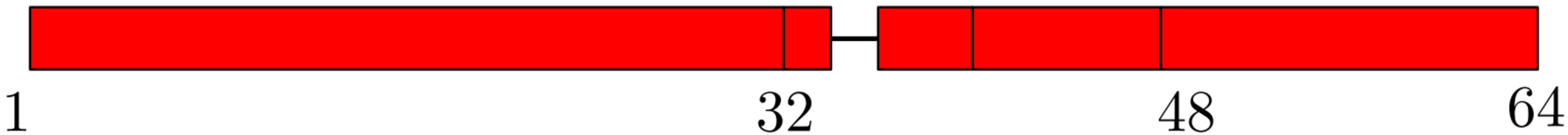
Is $x > 48$? Answer: No

Is $x > 40$? Answer: No

Is $x > 36$? Answer: No

Is $x > 34$? Answer: Yes

Binary Search Example



Is $x > 32$? Answer: Yes

Is $x > 48$? Answer: No

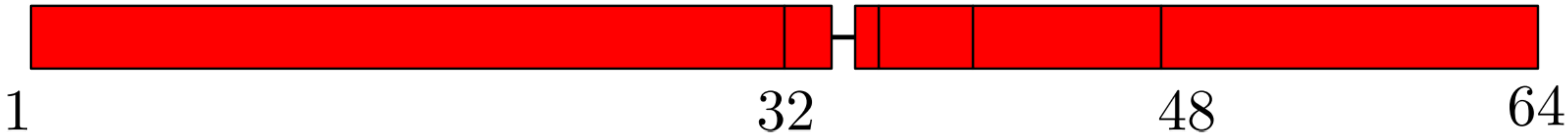
Is $x > 40$? Answer: No

Is $x > 36$? Answer: No

Is $x > 34$? Answer: Yes

Is $x > 35$?

Binary Search Example



Is $x > 32$? Answer: Yes

Is $x > 48$? Answer: No

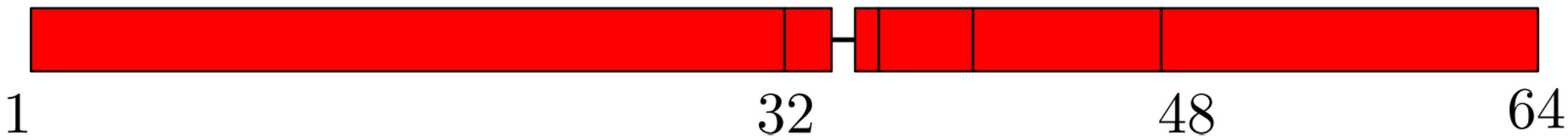
Is $x > 40$? Answer: No

Is $x > 36$? Answer: No

Is $x > 34$? Answer: Yes

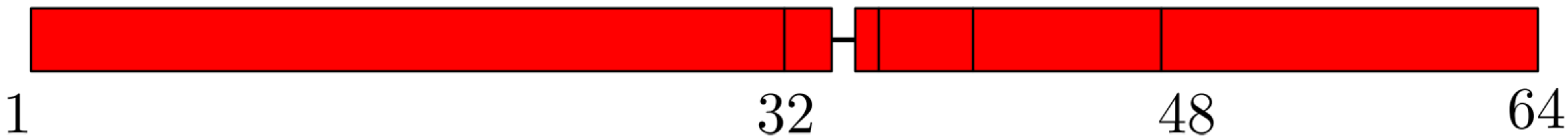
Is $x > 35$? Answer: No

Binary Search Example



Is $x > 32$?	Answer: Yes
Is $x > 48$?	Answer: No
Is $x > 40$?	Answer: No
Is $x > 36$?	Answer: No
Is $x > 34$?	Answer: Yes
Is $x > 35$?	Answer: No
Is $x = 35$?	

Binary Search Example



Is $x > 32$? Answer: Yes

Is $x > 48$? Answer: No

Is $x > 40$? Answer: No

Is $x > 36$? Answer: No

Is $x > 34$? Answer: Yes

Is $x > 35$? Answer: No

Is $x = 35$? Answer: BINGO!

Binary Search Example

- Method: Each guess reduces the problem to one in which the range is only half as big.

Binary Search Example

- Method: Each guess reduces the problem to one in which the range is only half as big.

  This divides the original problem into one that is only half as big; we can now (recursively) conquer this smaller problem.

Binary Search Example

- Method: Each guess reduces the problem to one in which the range is only half as big.

  This divides the original problem into one that is only half as big; we can now (recursively) conquer this smaller problem.

  Note: When n is a power of 2, $T(n)$, the number of questions in a binary search on $[1, n]$, satisfies

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

Binary Search Example

- Method: Each guess reduces the problem to one in which the range is only half as big.

  This divides the original problem into one that is only half as big; we can now (recursively) conquer this smaller problem.

  Note: When n is a power of 2, $T(n)$, the number of questions in a binary search on $[1, n]$, satisfies

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

  This can also be proved inductively, similar to the tower of Hanoi recurrence.

Binary Search Example

- $T(n)$: number of questions in a binary search on $[1, n]$

Binary Search Example

- $T(n)$: number of questions in a binary search on $[1, n]$

  Assume: n is a power of 2. Give recurrence for $T(n)$

Binary Search Example

- $T(n)$: number of questions in a binary search on $[1, n]$

  Assume: n is a power of 2.   Give recurrence for $T(n)$

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

Binary Search Example

- $T(n)$: number of questions in a binary search on $[1, n]$

  Assume: n is a power of 2. Give recurrence for $T(n)$

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

  Number of questions needed for binary search on n items is:

Binary Search Example

- $T(n)$: number of questions in a binary search on $[1, n]$

  Assume: n is a power of 2. Give recurrence for $T(n)$

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

  Number of questions needed for binary search on n items is:

  first step

  $+$

  time to perform binary search on the remaining $n/2$ items

Binary Search Example

- $T(n)$: number of questions in a binary search on $[1, n]$

  Assume: n is a power of 2.   Give recurrence for $T(n)$

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

  Number of questions needed for binary search on n items is:

  first step

  $+$

  time to perform binary search on the remaining $n/2$ items

  Base case (1 item): $T(1) = 1$ to ask: "Is the number k?"

Binary Search Example

- $$(*) \qquad T(n) = \begin{cases} C_1 & \text{if } n = 1 \\ T(\lceil n/2 \rceil) + C_2 & \text{if } n \geq 2 \end{cases}$$

  For simplicity, we will (usually) assume that n is a power of 2 (or sometimes 3 or 4) and also often that constants such as C_1, C_2 are 1. This will let us replace a recurrence such as (*) by one such as (**).

Binary Search Example

- $$(*) \qquad T(n) = \begin{cases} C_1 & \text{if } n = 1 \\ T(\lceil n/2 \rceil) + C_2 & \text{if } n \geq 2 \end{cases}$$

  For simplicity, we will (usually) assume that n is a power of 2 (or sometimes 3 or 4) and also often that constants such as C_1, C_2 are 1. This will let us replace a recurrence such as (*) by one such as (**).

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

Binary Search Example

- $$(*) \qquad T(n) = \begin{cases} C_1 & \text{if } n = 1 \\ T(\lceil n/2 \rceil) + C_2 & \text{if } n \geq 2 \end{cases}$$

  For simplicity, we will (usually) assume that n is a power of 2 (or sometimes 3 or 4) and also often that constants such as C_1, C_2 are 1. This will let us replace a recurrence such as (*) by one such as (**).

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

  In practice, the solution of (*) will be very close to that of (**) (this can be proved mathematically). Hence, we can restrict attention to (**).

Growth Rates of Solutions to Recurrences

- Divide and conquer algorithms
- Iterating recurrences
- Three different behaviors

Iterating Recurrences: Example 1

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

Iterating Recurrences: Example 1

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

This corresponds to solving a problem of size n, by

(i) solving 2 subproblems of size $n/2$ and
(ii) doing n units of additional work

or using $T(1)$ work for "bottom" case of $n = 1$

Iterating Recurrences: Example 1

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

  This corresponds to solving a problem of size n, by

  (i) solving 2 subproblems of size $n/2$ and
  (ii) doing n units of additional work

  or using $T(1)$ work for "bottom" case of $n = 1$

  In the course "Analysis of Algorithms", this is exactly how Mergesort works.

Iterating Recurrences: Example 1

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

  This corresponds to solving a problem of size n, by

  (i) solving 2 subproblems of size $n/2$ and
  (ii) doing n units of additional work

  or using $T(1)$ work for "bottom" case of $n = 1$

  In the course "Analysis of Algorithms", this is exactly how Mergesort works.

  We now see how to solve (*) by algebraically iterating the recurrence.

Iterating Recurrences: Example 1

- Algebraically iterating the recurrence

$$T(n) = 2T\left(\frac{n}{2}\right) + n$$

Iterating Recurrences: Example 1

- Algebraically iterating the recurrence

  Assume that n is a power of 2

$$T(n) = 2T\left(\frac{n}{2}\right) + n \qquad = 2\left(2T\left(\frac{n}{4}\right) + \frac{n}{2}\right) + n$$

Iterating Recurrences: Example 1

- Algebraically iterating the recurrence

  Assume that n is a power of 2

$$T(n) = 2T\left(\frac{n}{2}\right) + n \qquad = 2\left(2T\left(\frac{n}{4}\right) + \frac{n}{2}\right) + n$$

$$= 4T\left(\frac{n}{4}\right) + 2n \qquad = 4\left(2T\left(\frac{n}{8}\right) + \frac{n}{4}\right) + 2n$$

Iterating Recurrences: Example 1

- Algebraically iterating the recurrence

  Assume that n is a power of 2

$$
\begin{aligned}
{\color{blue} T(n) = 2T\left(\tfrac{n}{2}\right) + n} \quad &= 2\left(2T\left(\tfrac{n}{4}\right) + \tfrac{n}{2}\right) + n \\
= 4T\left(\tfrac{n}{4}\right) + 2n \quad &= 4\left(2T\left(\tfrac{n}{8}\right) + \tfrac{n}{4}\right) + 2n \\
= 8T\left(\tfrac{n}{8}\right) + 3n &
\end{aligned}
$$

Iterating Recurrences: Example 1

- Algebraically iterating the recurrence

  Assume that n is a power of 2

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

Iterating Recurrences: Example 1

- Algebraically iterating the recurrence

  Assume that n is a power of 2

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

End when $i = \log_2 n$

Iterating Recurrences: Example 1

- Algebraically iterating the recurrence

  Assume that n is a power of 2

$$T(n) = 2T\left(\frac{n}{2}\right) + n \qquad = 2\left(2T\left(\frac{n}{4}\right) + \frac{n}{2}\right) + n$$

$$= 4T\left(\frac{n}{4}\right) + 2n \qquad = 4\left(2T\left(\frac{n}{8}\right) + \frac{n}{4}\right) + 2n$$

$$= 8T\left(\frac{n}{8}\right) + 3n$$

$$\vdots \qquad \vdots$$

$$= 2^i T\left(\frac{n}{2^i}\right) + in$$

End when $i = \log_2 n$

$$\vdots \qquad \vdots$$

$$= 2^{\log_2 n} T\left(\frac{n}{2^{\log_2 n}}\right) + (\log_2 n)n$$

$$= nT(1) + n\log_2 n$$

Iterating Recurrences: Example 1

- We just iterated the recurrence to derive that the solution to

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

is $nT(1) + n\log_2 n$.

Iterating Recurrences: Example 1

- We just iterated the recurrence to derive that the solution to

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

is $nT(1) + n \log_2 n$.

Note: Technically, we still need to use **induction** to prove that our solution is correct. Practically, we never explicitly perform this step, since it is obvious how the induction would work.

Iterating Recurrences: Example 2

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

Iterating Recurrences: Example 2

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

$$T(n) = T\left(\frac{n}{2}\right) + 1$$

Iterating Recurrences: Example 2

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

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

Iterating Recurrences: Example 2

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

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

$$= T\left(\frac{n}{2^2}\right) + 2$$

Iterating Recurrences: Example 2

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

$$\begin{aligned} T(n) &= T\left(\frac{n}{2}\right) + 1 & &= \left(T\left(\frac{n}{2^2}\right) + 1\right) + 1 \\ &= T\left(\frac{n}{2^2}\right) + 2 & &= \left(T\left(\frac{n}{2^3}\right) + 1\right) + 2 \end{aligned}$$

Iterating Recurrences: Example 2

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

$$\begin{aligned} T(n) &= T\left(\frac{n}{2}\right) + 1 & &= \left(T\left(\frac{n}{2^2}\right) + 1\right) + 1 \\ &= T\left(\frac{n}{2^2}\right) + 2 & &= \left(T\left(\frac{n}{2^3}\right) + 1\right) + 2 \\ &= T\left(\frac{n}{2^3}\right) + 3 & & \end{aligned}$$

Iterating Recurrences: Example 2

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

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

Iterating Recurrences: Example 2

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

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

Iterating Recurrences: Example 2

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

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

Iterating Recurrences: Example 3

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

Iterating Recurrences: Example 3

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

$$\begin{aligned} T(n) &= T\left(\frac{n}{2}\right) + n \\ &= T\left(\frac{n}{2^2}\right) + \frac{n}{2} + n \end{aligned}$$

Iterating Recurrences: Example 3

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

$$\begin{aligned} T(n) &= T\left(\frac{n}{2}\right) + n \\ &= T\left(\frac{n}{2^2}\right) + \frac{n}{2} + n \\ &= T\left(\frac{n}{2^3}\right) + \frac{n}{2^2} + \frac{n}{2} + n \end{aligned}$$

Iterating Recurrences: Example 3

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

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

Iterating Recurrences: Example 3

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

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

Iterating Recurrences: Example 3

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

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

Iterating Recurrences: Example 3

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

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

Iterating Recurrences: Example 4

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

Iterating Recurrences: Example 4

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

$$T(n) = 3T\left(\frac{n}{3}\right) + n$$

Iterating Recurrences: Example 4

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

$$T(n) = 3T\left(\frac{n}{3}\right) + n \qquad = 3\left(3T\left(\frac{n}{3^2}\right) + \frac{n}{3}\right) + n$$

Iterating Recurrences: Example 4

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

$$T(n) = 3T\left(\frac{n}{3}\right) + n \qquad = 3\left(3T\left(\frac{n}{3^2}\right) + \frac{n}{3}\right) + n$$

$$= 3^2 T\left(\frac{n}{3^2}\right) + 2n$$

Iterating Recurrences: Example 4

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

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

Iterating Recurrences: Example 4

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

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

Iterating Recurrences: Example 4

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

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

Iterating Recurrences: Example 4

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

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

Iterating Recurrences: Example 4

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

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

Iterating Recurrences: Example 5

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

Iterating Recurrences: Example 5

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

$$T(n) = 4T\left(\frac{n}{2}\right) + n$$

Iterating Recurrences: Example 5

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

$$T(n) = 4T\left(\frac{n}{2}\right) + n \qquad = 4\left(4T\left(\frac{n}{2^2}\right) + \frac{n}{2}\right) + n$$

Iterating Recurrences: Example 5

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

$$T(n) = 4T\left(\frac{n}{2}\right) + n \qquad = 4\left(4T\left(\frac{n}{2^2}\right) + \frac{n}{2}\right) + n$$

$$= 4^2 T\left(\frac{n}{2^2}\right) + \frac{4}{2}n + n$$

Iterating Recurrences: Example 5

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

$$T(n) = 4T\left(\frac{n}{2}\right) + n \qquad = 4\left(4T\left(\frac{n}{2^2}\right) + \frac{n}{2}\right) + n$$

$$= 4^2 T\left(\frac{n}{2^2}\right) + \frac{4}{2}n + n \quad = 4^2\left(4T\left(\frac{n}{2^3}\right) + \frac{n}{2^2}\right) + \frac{4}{2}n + n$$

Iterating Recurrences: Example 5

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

$$T(n) = 4T\left(\frac{n}{2}\right) + n \qquad = 4\left(4T\left(\frac{n}{2^2}\right) + \frac{n}{2}\right) + n$$

$$= 4^2T\left(\frac{n}{2^2}\right) + \frac{4}{2}n + n \quad = 4^2\left(4T\left(\frac{n}{2^3}\right) + \frac{n}{2^2}\right) + \frac{4}{2}n + n$$

$$= 4^3T\left(\frac{n}{2^3}\right) + \frac{4^2}{2^2}n + \frac{4}{2}n + n$$

Iterating Recurrences: Example 5

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

$$T(n) = 4T\left(\frac{n}{2}\right) + n \qquad = 4\left(4T\left(\frac{n}{2^2}\right) + \frac{n}{2}\right) + n$$

$$= 4^2 T\left(\frac{n}{2^2}\right) + \frac{4}{2}n + n \quad = 4^2\left(4T\left(\frac{n}{2^3}\right) + \frac{n}{2^2}\right) + \frac{4}{2}n + n$$

$$= 4^3 T\left(\frac{n}{2^3}\right) + \frac{4^2}{2^2}n + \frac{4}{2}n + n$$

$$\vdots \qquad \vdots$$

$$= 4^i T\left(\frac{n}{2^i}\right) + \frac{4^{i-1}}{2^{i-1}}n + \cdots + \frac{4^2}{2^2}n + n$$

Iterating Recurrences: Example 5

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

$$T(n) = 4T\left(\frac{n}{2}\right) + n \qquad = 4\left(4T\left(\frac{n}{2^2}\right) + \frac{n}{2}\right) + n$$

$$= 4^2T\left(\frac{n}{2^2}\right) + \frac{4}{2}n + n \quad = 4^2\left(4T\left(\frac{n}{2^3}\right) + \frac{n}{2^2}\right) + \frac{4}{2}n + n$$

$$= 4^3T\left(\frac{n}{2^3}\right) + \frac{4^2}{2^2}n + \frac{4}{2}n + n$$

$$\vdots \qquad \vdots$$

$$= 4^iT\left(\frac{n}{2^i}\right) + \frac{4^{i-1}}{2^{i-1}}n + \cdots + \frac{4^2}{2^2}n + n$$

$$\vdots \qquad \vdots$$

$$= 4^{\log_2 n}T\left(\frac{n}{2^{\log_2 n}}\right) + \frac{4^{\log_2 n-1}}{2^{\log_2 n-1}}n + \cdots + \frac{4}{2}n + n$$

Iterating Recurrences: Example 5

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

$$T(n) = 4T\left(\frac{n}{2}\right) + n \qquad = 4\left(4T\left(\frac{n}{2^2}\right) + \frac{n}{2}\right) + n$$

$$= 4^2 T\left(\frac{n}{2^2}\right) + \frac{4}{2}n + n \quad = 4^2\left(4T\left(\frac{n}{2^3}\right) + \frac{n}{2^2}\right) + \frac{4}{2}n + n$$

$$= 4^3 T\left(\frac{n}{2^3}\right) + \frac{4^2}{2^2}n + \frac{4}{2}n + n$$

$$\vdots \qquad \vdots$$

$$= 4^i T\left(\frac{n}{2^i}\right) + \frac{4^{i-1}}{2^{i-1}}n + \cdots + \frac{4^2}{2^2}n + n$$

$$\vdots \qquad \vdots$$

$$= 4^{\log_2 n} T\left(\frac{n}{2^{\log_2 n}}\right) + \frac{4^{\log_2 n - 1}}{2^{\log_2 n - 1}}n + \cdots + \frac{4}{2}n + n$$

$$= 2n^2 - n$$

Three Different Behaviors

- Compare the iteration for the recurrences

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

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

$$T(n) = 4T(n/2) + n$$

Three Different Behaviors

- Compare the iteration for the recurrences

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

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

$$T(n) = 4T(n/2) + n$$

◇ all three recurrences iterate $\log_2 n$ times

◇ in each case, size of subproblem in next iteration is half the size in the preceding iteration level

Three Different Behaviors

- **Theorem** Suppose that we have a recurrence of the form
$$T(n) = aT(n/2) + n,$$
where a is a positive integer and $T(1)$ is nonnegative. Then we have the following big Θ bounds on the solution:

    1. If $a < 2$, then $T(n) = \Theta(n)$.
    2. If $a = 2$, then $T(n) = \Theta(n \log n)$.
    3. If $a > 2$, then $T(n) = \Theta(n^{\log_2 a})$

Three Different Behaviors

- **Theorem** Suppose that we have a recurrence of the form
$$T(n) = aT(n/2) + n,$$
where a is a positive integer and $T(1)$ is nonnegative. Then we have the following big Θ bounds on the solution:

  1. If $a < 2$, then $T(n) = \Theta(n)$.
  2. If $a = 2$, then $T(n) = \Theta(n \log n)$.
  3. If $a > 2$, then $T(n) = \Theta(n^{\log_2 a})$

Proof

We already proved Case 1 when $a = 1$ in Example 3.
(will not prove it for $1 < a < 2$)
We already proved Case 2 in Example 1.
We will now prove Case 3.

Iterating Recurrences

- $T(n) = aT(n/2) + n$, where $a > 2$. Assume that $n = 2^i$.

Iterating Recurrences

- $T(n) = aT(n/2) + n$, where $a > 2$. Assume that $n = 2^i$.

  Iterating as in Example 5 gives

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

Iterating Recurrences

- $T(n) = aT(n/2) + n$, where $a > 2$. Assume that $n = 2^i$.

  Iterating as in Example 5 gives

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

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

Work at "bottom"

Iterated Work

Total work

- The total work is

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

Total work

- The total work is

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

Since $a > 2$, the geometric series is Θ of the largest term.

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

Total work

- n times the largest term in the geometric series is

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

Total work

- n times the largest term in the geometric series is

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

Notice that

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

Total work

- n times the largest term in the geometric series is

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

Notice that

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

So the total work is

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

Total work

- n times the largest term in the geometric series is

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

  Notice that

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

  So the total work is

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

$$\Theta\left(n^{\log_2 a}\right) \qquad \Theta\left(n^{\log_2 a}\right)$$

Example 5 Recap

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

Example 5 Recap

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

$a = 4$, so the Theorem says that

$$T(n) = \Theta\left(n^{\log_2 a}\right) = \Theta\left(n^{\log_2 4}\right) = \Theta(n^2)$$

Example 5 Recap

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

$a = 4$, so the Theorem says that

$$T(n) = \Theta\left(n^{\log_2 a}\right) = \Theta\left(n^{\log_2 4}\right) = \Theta(n^2)$$

This matches with the exact answer of $2n^2 - n$.

Three Different Behaviors

- **Theorem** Suppose that we have a recurrence of the form
$$T(n) = aT(n/2) + n,$$
where a is a positive integer and $T(1)$ is nonnegative. Then we have the following big Θ bounds on the solution:

  1. If $a < 2$, then $T(n) = \Theta(n)$.
  2. If $a = 2$, then $T(n) = \Theta(n \log n)$.
  3. If $a > 2$, then $T(n) = \Theta(n^{\log_2 a})$

The Master Theorem

- **Theorem** Suppose that we have a recurrence of the form
$$T(n) = aT(n/b) + cn^d,$$
where a is a positive integer, $b \geq 1$, c, d are real numbers with c positive and d nonnegative, and $T(1)$ is nonnegative. Then we have the following big Θ bounds on the solution:

1. If $a < b^d$, then $T(n) = \Theta(n^d)$.
2. If $a = b^d$, then $T(n) = \Theta(n^d \log n)$.
3. If $a > b^d$, then $T(n) = \Theta(n^{\log_b a})$

Next Lecture

- counting ...