

# Recurrences I

## COMS10018 - Algorithms

Dr Christian Konrad

## **Algorithmic Design Principle: Divide-and-conquer**

## Algorithmic Design Principle: Divide-and-conquer

- 1 **Divide** the problem into a number of subproblems that are smaller instances of the same problem

## Algorithmic Design Principle: Divide-and-conquer

- ➊ **Divide** the problem into a number of subproblems that are smaller instances of the same problem
- ➋ **Conquer** the subproblems by solving them recursively (if subproblems have constant size, solve them *directly*)

## Algorithmic Design Principle: Divide-and-conquer

- ① **Divide** the problem into a number of subproblems that are smaller instances of the same problem
- ② **Conquer** the subproblems by solving them recursively (if subproblems have constant size, solve them *directly*)
- ③ **Combine** the solutions to the subproblems into the solution for the original problem

## Algorithmic Design Principle: Divide-and-conquer

- ➊ **Divide** the problem into a number of subproblems that are smaller instances of the same problem
- ➋ **Conquer** the subproblems by solving them recursively (if subproblems have constant size, solve them *directly*)
- ➌ **Combine** the solutions to the subproblems into the solution for the original problem

## Examples

## Algorithmic Design Principle: Divide-and-conquer

- 1 **Divide** the problem into a number of subproblems that are smaller instances of the same problem
- 2 **Conquer** the subproblems by solving them recursively (if subproblems have constant size, solve them *directly*)
- 3 **Combine** the solutions to the subproblems into the solution for the original problem

## Examples

Quicksort,

## Algorithmic Design Principle: Divide-and-conquer

- ➊ **Divide** the problem into a number of subproblems that are smaller instances of the same problem
- ➋ **Conquer** the subproblems by solving them recursively (if subproblems have constant size, solve them *directly*)
- ➌ **Combine** the solutions to the subproblems into the solution for the original problem

## Examples

Quicksort, Mergesort,



## Algorithmic Design Principle: Divide-and-conquer

- ➊ **Divide** the problem into a number of subproblems that are smaller instances of the same problem
- ➋ **Conquer** the subproblems by solving them recursively (if subproblems have constant size, solve them *directly*)
- ➌ **Combine** the solutions to the subproblems into the solution for the original problem

## Examples

Quicksort, Mergesort, maximum subarray algorithm,

## Algorithmic Design Principle: Divide-and-conquer

- ➊ **Divide** the problem into a number of subproblems that are smaller instances of the same problem
- ➋ **Conquer** the subproblems by solving them recursively (if subproblems have constant size, solve them *directly*)
- ➌ **Combine** the solutions to the subproblems into the solution for the original problem

## Examples

Quicksort, Mergesort, maximum subarray algorithm, binary search,

## Algorithmic Design Principle: Divide-and-conquer

- 1 **Divide** the problem into a number of subproblems that are smaller instances of the same problem
- 2 **Conquer** the subproblems by solving them recursively (if subproblems have constant size, solve them *directly*)
- 3 **Combine** the solutions to the subproblems into the solution for the original problem

## Examples

Quicksort, Mergesort, maximum subarray algorithm, binary search, FAST-PEAK-FINDING, ...

# Example: Mergesort

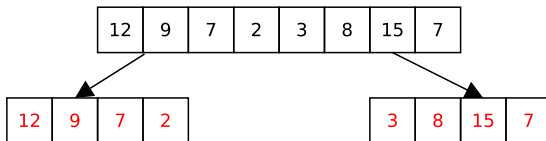
## Recall: Mergesort

# Example: Mergesort

## Recall: Mergesort

### 1 Divide

Split input array  $A$  of length  $n$  into subarrays  $A_1 = A[0, \lfloor n/2 \rfloor]$  and  $A_2 = A[\lfloor n/2 \rfloor + 1, n - 1]$



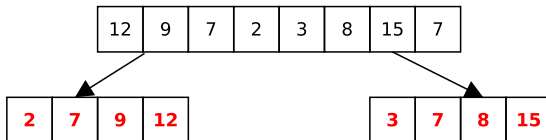
# Example: Mergesort

## Recall: Mergesort

① **Divide**  $A \rightarrow A_1$  and  $A_2$

② **Conquer**

Sort  $A_1$  and  $A_2$  recursively using the same algorithm

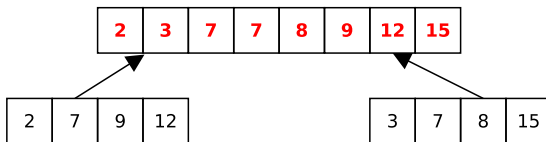


# Example: Mergesort

## Recall: Mergesort

- 1 **Divide**  $A \rightarrow A_1$  and  $A_2$
- 2 **Conquer** Solve  $A_1$  and  $A_2$
- 3 **Combine**

Combine sorted subarrays  $A_1$  and  $A_2$  and obtain sorted array  $A$

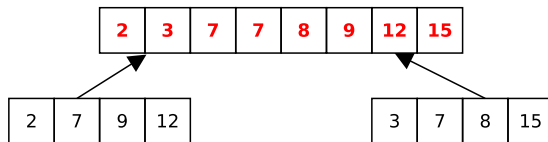


# Example: Mergesort

## Recall: Mergesort

- 1 **Divide**  $A \rightarrow A_1$  and  $A_2$
- 2 **Conquer** Solve  $A_1$  and  $A_2$
- 3 **Combine**

Combine sorted subarrays  $A_1$  and  $A_2$  and obtain sorted array  $A$



**Runtime:** (assuming that  $n$  is a power of 2)

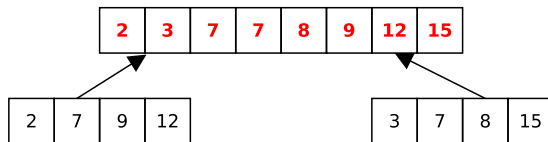


# Example: Mergesort

## Recall: Mergesort

- 1 **Divide**  $A \rightarrow A_1$  and  $A_2$
- 2 **Conquer** Solve  $A_1$  and  $A_2$
- 3 **Combine**

Combine sorted subarrays  $A_1$  and  $A_2$  and obtain sorted array  $A$



**Runtime:** (assuming that  $n$  is a power of 2)

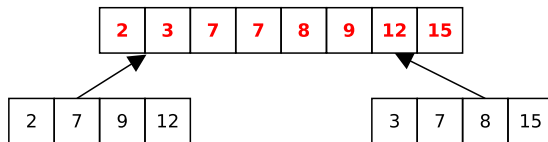
$$T(1) = O(1)$$

# Example: Mergesort

## Recall: Mergesort

- 1 **Divide**  $A \rightarrow A_1$  and  $A_2$
- 2 **Conquer** Solve  $A_1$  and  $A_2$
- 3 **Combine**

Combine sorted subarrays  $A_1$  and  $A_2$  and obtain sorted array  $A$



**Runtime:** (assuming that  $n$  is a power of 2)

$$T(1) = O(1)$$

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

# How to solve Recurrences?

## Recurrences

# How to solve Recurrences?

## Recurrences

- Divide-and-Conquer algorithms naturally lead to recurrences

# How to solve Recurrences?

## Recurrences

- Divide-and-Conquer algorithms naturally lead to recurrences
- How can we *solve* them?

# How to solve Recurrences?

## Recurrences

- Divide-and-Conquer algorithms naturally lead to recurrences
- How can we *solve* them? Often only interested in asymptotic upper bounds

# How to solve Recurrences?

## Recurrences

- Divide-and-Conquer algorithms naturally lead to recurrences
- How can we *solve* them? Often only interested in asymptotic upper bounds

## Methods for solving recurrences

# How to solve Recurrences?

## Recurrences

- Divide-and-Conquer algorithms naturally lead to recurrences
- How can we *solve* them? Often only interested in asymptotic upper bounds

## Methods for solving recurrences

- Substitution method



# How to solve Recurrences?

## Recurrences

- Divide-and-Conquer algorithms naturally lead to recurrences
- How can we *solve* them? Often only interested in asymptotic upper bounds

## Methods for solving recurrences

- Substitution method  
guess solution, verify, induction

# How to solve Recurrences?

## Recurrences

- Divide-and-Conquer algorithms naturally lead to recurrences
- How can we *solve* them? Often only interested in asymptotic upper bounds

## Methods for solving recurrences

- Substitution method  
guess solution, verify, induction
- Recursion-tree method (previously seen for Mergesort and maximum subarray problem)

# How to solve Recurrences?

## Recurrences

- Divide-and-Conquer algorithms naturally lead to recurrences
- How can we *solve* them? Often only interested in asymptotic upper bounds

## Methods for solving recurrences

- Substitution method  
guess solution, verify, induction
- Recursion-tree method (previously seen for Mergesort and maximum subarray problem)  
may have plenty of awkward details, provides good guess that can be verified with substitution method

# How to solve Recurrences?

## Recurrences

- Divide-and-Conquer algorithms naturally lead to recurrences
- How can we *solve* them? Often only interested in asymptotic upper bounds

## Methods for solving recurrences

- Substitution method  
guess solution, verify, induction
- Recursion-tree method (previously seen for Mergesort and maximum subarray problem)  
may have plenty of awkward details, provides good guess that can be verified with substitution method
- ~~Master theorem~~

# How to solve Recurrences?

## Recurrences

- Divide-and-Conquer algorithms naturally lead to recurrences
- How can we *solve* them? Often only interested in asymptotic upper bounds

## Methods for solving recurrences

- Substitution method  
guess solution, verify, induction
- Recursion-tree method (previously seen for Mergesort and maximum subarray problem)  
may have plenty of awkward details, provides good guess that can be verified with substitution method
- ~~Master theorem~~  
very powerful, cannot always be applied

# The Substitution Method

## **The Substitution Method**

# The Substitution Method

## The Substitution Method

- 1 Guess the form of the solution

# The Substitution Method

## The Substitution Method

- 1 Guess the form of the solution
- 2 Use mathematical induction to find the constants and show that the solution works



# The Substitution Method

## The Substitution Method

- 1 Guess the form of the solution
- 2 Use mathematical induction to find the constants and show that the solution works
- 3 Method provides an upper bound on the recurrence

# The Substitution Method

## The Substitution Method

- 1 Guess the form of the solution
- 2 Use mathematical induction to find the constants and show that the solution works
- 3 Method provides an upper bound on the recurrence

**Example** (suppose  $n$  is always a power of two)

# The Substitution Method

## The Substitution Method

- 1 Guess the form of the solution
- 2 Use mathematical induction to find the constants and show that the solution works
- 3 Method provides an upper bound on the recurrence

**Example** (suppose  $n$  is always a power of two)

$$T(1) = O(1)$$

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

# The Substitution Method

## The Substitution Method

- 1 Guess the form of the solution
- 2 Use mathematical induction to find the constants and show that the solution works
- 3 Method provides an upper bound on the recurrence

**Example** (suppose  $n$  is always a power of two)

$$T(1) \leq c_1$$

$$T(n) \leq 2T(n/2) + c_2n$$

Eliminate  $O$ -notation in recurrence

# The Substitution Method

## The Substitution Method

- 1 Guess the form of the solution
- 2 Use mathematical induction to find the constants and show that the solution works
- 3 Method provides an upper bound on the recurrence

**Example** (suppose  $n$  is always a power of two)

$$T(1) \leq c_1$$

$$T(n) \leq 2T(n/2) + c_2n$$

Eliminate  $O$ -notation in recurrence

**Step 1. Guess good upper bound**

# The Substitution Method

## The Substitution Method

- 1 Guess the form of the solution
- 2 Use mathematical induction to find the constants and show that the solution works
- 3 Method provides an upper bound on the recurrence

**Example** (suppose  $n$  is always a power of two)

$$T(1) \leq c_1$$

$$T(n) \leq 2T(n/2) + c_2n$$

Eliminate  $O$ -notation in recurrence

**Step 1. Guess good upper bound**

$$T(n) \leq Cn \log n$$

# The Substitution Method (2)

## Step 2. Substitute into the Recurrence

# The Substitution Method (2)

## Step 2. Substitute into the Recurrence

- Assume that our guess  $T(n) \leq Cn \log n$  is correct for every  $n' < n$



# The Substitution Method (2)

## Step 2. Substitute into the Recurrence

- Assume that our guess  $T(n) \leq Cn \log n$  is correct for every  $n' < n$
- Corresponds to induction step of a proof by induction

# The Substitution Method (2)

## Step 2. Substitute into the Recurrence

- Assume that our guess  $T(n) \leq Cn \log n$  is correct for every  $n' < n$
- Corresponds to induction step of a proof by induction

$$T(n)$$

# The Substitution Method (2)

## Step 2. Substitute into the Recurrence

- Assume that our guess  $T(n) \leq Cn \log n$  is correct for every  $n' < n$
- Corresponds to induction step of a proof by induction

$$T(n) \leq 2T(n/2) + c_2n$$

# The Substitution Method (2)

## Step 2. Substitute into the Recurrence

- Assume that our guess  $T(n) \leq Cn \log n$  is correct for every  $n' < n$
- Corresponds to induction step of a proof by induction

$$T(n) \leq 2T(n/2) + c_2n \leq 2C\frac{n}{2}\log\left(\frac{n}{2}\right) + c_2n$$

# The Substitution Method (2)

## Step 2. Substitute into the Recurrence

- Assume that our guess  $T(n) \leq Cn \log n$  is correct for every  $n' < n$
- Corresponds to induction step of a proof by induction

$$\begin{aligned} T(n) &\leq 2T(n/2) + c_2n \leq 2C\frac{n}{2}\log\left(\frac{n}{2}\right) + c_2n \\ &= Cn(\log(n) - \log(2)) + c_2n \end{aligned}$$

# The Substitution Method (2)

## Step 2. Substitute into the Recurrence

- Assume that our guess  $T(n) \leq Cn \log n$  is correct for every  $n' < n$
- Corresponds to induction step of a proof by induction

$$\begin{aligned} T(n) &\leq 2T(n/2) + c_2n \leq 2C\frac{n}{2}\log\left(\frac{n}{2}\right) + c_2n \\ &= Cn(\log(n) - \log(2)) + c_2n \\ &= Cn \log n - Cn + c_2n \end{aligned}$$

# The Substitution Method (2)

## Step 2. Substitute into the Recurrence

- Assume that our guess  $T(n) \leq Cn \log n$  is correct for every  $n' < n$
- Corresponds to induction step of a proof by induction

$$\begin{aligned} T(n) &\leq 2T(n/2) + c_2n \leq 2C\frac{n}{2}\log\left(\frac{n}{2}\right) + c_2n \\ &= Cn(\log(n) - \log(2)) + c_2n \\ &= Cn \log n - Cn + c_2n \leq Cn \log n, \end{aligned}$$

if we chose  $C \geq c_2$ .

# The Substitution Method (2)

## Step 2. Substitute into the Recurrence

- Assume that our guess  $T(n) \leq Cn \log n$  is correct for every  $n' < n$
- Corresponds to induction step of a proof by induction

$$\begin{aligned} T(n) &\leq 2T(n/2) + c_2n \leq 2C\frac{n}{2}\log\left(\frac{n}{2}\right) + c_2n \\ &= Cn(\log(n) - \log(2)) + c_2n \\ &= Cn \log n - Cn + c_2n \leq Cn \log n, \end{aligned}$$

if we chose  $C \geq c_2$ . ✓



# The Substitution Method (2)

## Step 2. Substitute into the Recurrence

- Assume that our guess  $T(n) \leq Cn \log n$  is correct for every  $n' < n$
- Corresponds to induction step of a proof by induction

$$\begin{aligned} T(n) &\leq 2T(n/2) + c_2n \leq 2C\frac{n}{2}\log\left(\frac{n}{2}\right) + c_2n \\ &= Cn(\log(n) - \log(2)) + c_2n \\ &= Cn \log n - Cn + c_2n \leq Cn \log n, \end{aligned}$$

if we chose  $C \geq c_2$ . ✓

## Verify the Base Case

# The Substitution Method (2)

## Step 2. Substitute into the Recurrence

- Assume that our guess  $T(n) \leq Cn \log n$  is correct for every  $n' < n$
- Corresponds to induction step of a proof by induction

$$\begin{aligned} T(n) &\leq 2T(n/2) + c_2n \leq 2C\frac{n}{2}\log\left(\frac{n}{2}\right) + c_2n \\ &= Cn(\log(n) - \log(2)) + c_2n \\ &= Cn \log n - Cn + c_2n \leq Cn \log n, \end{aligned}$$

if we chose  $C \geq c_2$ . ✓

## Verify the Base Case

$$T(1)$$

# The Substitution Method (2)

## Step 2. Substitute into the Recurrence

- Assume that our guess  $T(n) \leq Cn \log n$  is correct for every  $n' < n$
- Corresponds to induction step of a proof by induction

$$\begin{aligned} T(n) &\leq 2T(n/2) + c_2n \leq 2C\frac{n}{2}\log\left(\frac{n}{2}\right) + c_2n \\ &= Cn(\log(n) - \log(2)) + c_2n \\ &= Cn \log n - Cn + c_2n \leq Cn \log n, \end{aligned}$$

if we chose  $C \geq c_2$ . ✓

## Verify the Base Case

$$T(1) \leq C \cdot 1 \log(1)$$

# The Substitution Method (2)

## Step 2. Substitute into the Recurrence

- Assume that our guess  $T(n) \leq Cn \log n$  is correct for every  $n' < n$
- Corresponds to induction step of a proof by induction

$$\begin{aligned} T(n) &\leq 2T(n/2) + c_2 n \leq 2C \frac{n}{2} \log\left(\frac{n}{2}\right) + c_2 n \\ &= Cn (\log(n) - \log(2)) + c_2 n \\ &= Cn \log n - Cn + c_2 n \leq Cn \log n, \end{aligned}$$

if we chose  $C \geq c_2$ . ✓

## Verify the Base Case

$$T(1) \leq C \cdot 1 \log(1) = 0$$

# The Substitution Method (2)

## Step 2. Substitute into the Recurrence

- Assume that our guess  $T(n) \leq Cn \log n$  is correct for every  $n' < n$
- Corresponds to induction step of a proof by induction

$$\begin{aligned} T(n) &\leq 2T(n/2) + c_2n \leq 2C\frac{n}{2}\log\left(\frac{n}{2}\right) + c_2n \\ &= Cn(\log(n) - \log(2)) + c_2n \\ &= Cn \log n - Cn + c_2n \leq Cn \log n, \end{aligned}$$

if we chose  $C \geq c_2$ . ✓

## Verify the Base Case

$$T(1) \leq C \cdot 1 \log(1) = 0 \not\geq c_1$$

# The Substitution Method (2)

## Step 2. Substitute into the Recurrence

- Assume that our guess  $T(n) \leq Cn \log n$  is correct for every  $n' < n$
- Corresponds to induction step of a proof by induction

$$\begin{aligned} T(n) &\leq 2T(n/2) + c_2 n \leq 2C \frac{n}{2} \log\left(\frac{n}{2}\right) + c_2 n \\ &= Cn (\log(n) - \log(2)) + c_2 n \\ &= Cn \log n - Cn + c_2 n \leq Cn \log n, \end{aligned}$$

if we chose  $C \geq c_2$ . ✓

## Verify the Base Case

$$T(1) \leq C \cdot 1 \log(1) = 0 \not\geq c_1 \quad \times$$

# The Substitution Method (2)

## Step 2. Substitute into the Recurrence

- Assume that our guess  $T(n) \leq Cn \log n$  is correct for every  $n' < n$
- Corresponds to induction step of a proof by induction

$$\begin{aligned} T(n) &\leq 2T(n/2) + c_2 n \leq 2C \frac{n}{2} \log\left(\frac{n}{2}\right) + c_2 n \\ &= Cn (\log(n) - \log(2)) + c_2 n \\ &= Cn \log n - Cn + c_2 n \leq Cn \log n, \end{aligned}$$

if we chose  $C \geq c_2$ . ✓

## Verify the Base Case

$$T(1) \leq C \cdot 1 \log(1) = 0 \not\geq c_1 \quad \times$$

The base case is a problem...

# The Substitution Method (3)

**Recall:**  $T(1) = c_1$  and  $T(n) = 2T(n/2) + c_2n$

Our guess:  $T(n) \leq Cn \log n$  (induction step holds for  $C \geq c_2$ )



# The Substitution Method (3)

**Recall:**  $T(1) = c_1$  and  $T(n) = 2T(n/2) + c_2n$

Our guess:  $T(n) \leq Cn \log n$  (induction step holds for  $C \geq c_2$ )

**Solution:**

# The Substitution Method (3)

**Recall:**  $T(1) = c_1$  and  $T(n) = 2T(n/2) + c_2n$

Our guess:  $T(n) \leq Cn \log n$  (induction step holds for  $C \geq c_2$ )

**Solution:** Choose a different base case!  $n = 2$

# The Substitution Method (3)

**Recall:**  $T(1) = c_1$  and  $T(n) = 2T(n/2) + c_2n$

Our guess:  $T(n) \leq Cn \log n$  (induction step holds for  $C \geq c_2$ )

**Solution:** Choose a different base case!  $n = 2$

$$T(2)$$

# The Substitution Method (3)

**Recall:**  $T(1) = c_1$  and  $T(n) = 2T(n/2) + c_2n$

Our guess:  $T(n) \leq Cn \log n$  (induction step holds for  $C \geq c_2$ )

**Solution:** Choose a different base case!  $n = 2$

$$T(2) \leq 2T(1) + 2c_2$$

# The Substitution Method (3)

**Recall:**  $T(1) = c_1$  and  $T(n) = 2T(n/2) + c_2n$

Our guess:  $T(n) \leq Cn \log n$  (induction step holds for  $C \geq c_2$ )

**Solution:** Choose a different base case!  $n = 2$

$$T(2) \leq 2T(1) + 2c_2 = 2c_1 + 2c_2$$

# The Substitution Method (3)

**Recall:**  $T(1) = c_1$  and  $T(n) = 2T(n/2) + c_2n$

Our guess:  $T(n) \leq Cn \log n$  (induction step holds for  $C \geq c_2$ )

**Solution:** Choose a different base case!  $n = 2$

$$T(2) \leq 2T(1) + 2c_2 = 2c_1 + 2c_2 = 2(c_2 + c_1)$$

# The Substitution Method (3)

**Recall:**  $T(1) = c_1$  and  $T(n) = 2T(n/2) + c_2n$

Our guess:  $T(n) \leq Cn \log n$  (induction step holds for  $C \geq c_2$ )

**Solution:** Choose a different base case!  $n = 2$

$$T(2) \leq 2T(1) + 2c_2 = 2c_1 + 2c_2 = 2(c_2 + c_1)$$

$C2 \log 2$

# The Substitution Method (3)

**Recall:**  $T(1) = c_1$  and  $T(n) = 2T(n/2) + c_2n$

Our guess:  $T(n) \leq Cn \log n$  (induction step holds for  $C \geq c_2$ )

**Solution:** Choose a different base case!  $n = 2$

$$\begin{aligned} T(2) &\leq 2T(1) + 2c_2 = 2c_1 + 2c_2 = 2(c_2 + c_1) \\ C2 \log 2 &= 2C \end{aligned}$$



# The Substitution Method (3)

**Recall:**  $T(1) = c_1$  and  $T(n) = 2T(n/2) + c_2n$

Our guess:  $T(n) \leq Cn \log n$  (induction step holds for  $C \geq c_2$ )

**Solution:** Choose a different base case!  $n = 2$

$$\begin{aligned} T(2) &\leq 2T(1) + 2c_2 = 2c_1 + 2c_2 = 2(c_2 + c_1) \\ C2 \log 2 &= 2C \end{aligned}$$

Hence, for every  $C \geq c_2 + c_1$ , our guess holds for  $n = 2$ :

$$T(2) \leq C2 \log 2 .$$

# The Substitution Method (3)

**Recall:**  $T(1) = c_1$  and  $T(n) = 2T(n/2) + c_2n$

Our guess:  $T(n) \leq Cn \log n$  (induction step holds for  $C \geq c_2$ )

**Solution:** Choose a different base case!  $n = 2$

$$\begin{aligned} T(2) &\leq 2T(1) + 2c_2 = 2c_1 + 2c_2 = 2(c_2 + c_1) \\ C2 \log 2 &= 2C \end{aligned}$$

Hence, for every  $C \geq c_2 + c_1$ , our guess holds for  $n = 2$ :

$$T(2) \leq C2 \log 2 .$$

**Result**

# The Substitution Method (3)

**Recall:**  $T(1) = c_1$  and  $T(n) = 2T(n/2) + c_2n$

Our guess:  $T(n) \leq Cn \log n$  (induction step holds for  $C \geq c_2$ )

**Solution:** Choose a different base case!  $n = 2$

$$\begin{aligned} T(2) &\leq 2T(1) + 2c_2 = 2c_1 + 2c_2 = 2(c_2 + c_1) \\ C2 \log 2 &= 2C \end{aligned}$$

Hence, for every  $C \geq c_2 + c_1$ , our guess holds for  $n = 2$ :

$$T(2) \leq C2 \log 2 .$$

## Result

- We proved  $T(n) \leq Cn \log n$ , for every  $n \geq 2$ , when choosing  $C \geq c_1 + c_2$

# The Substitution Method (3)

**Recall:**  $T(1) = c_1$  and  $T(n) = 2T(n/2) + c_2n$

Our guess:  $T(n) \leq Cn \log n$  (induction step holds for  $C \geq c_2$ )

**Solution:** Choose a different base case!  $n = 2$

$$\begin{aligned} T(2) &\leq 2T(1) + 2c_2 = 2c_1 + 2c_2 = 2(c_2 + c_1) \\ C2 \log 2 &= 2C \end{aligned}$$

Hence, for every  $C \geq c_2 + c_1$ , our guess holds for  $n = 2$ :

$$T(2) \leq C2 \log 2 .$$

## Result

- We proved  $T(n) \leq Cn \log n$ , for every  $n \geq 2$ , when choosing  $C \geq c_1 + c_2$
- **Observe:** This implies  $T(n) \in O(n \log n)$  (important)

# The Substitution Method (3)

**Recall:**  $T(1) = c_1$  and  $T(n) = 2T(n/2) + c_2n$

Our guess:  $T(n) \leq Cn \log n$  (induction step holds for  $C \geq c_2$ )

**Solution:** Choose a different base case!  $n = 2$

$$\begin{aligned} T(2) &\leq 2T(1) + 2c_2 = 2c_1 + 2c_2 = 2(c_2 + c_1) \\ C2 \log 2 &= 2C \end{aligned}$$

Hence, for every  $C \geq c_2 + c_1$ , our guess holds for  $n = 2$ :

$$T(2) \leq C2 \log 2 .$$

## Result

- We proved  $T(n) \leq Cn \log n$ , for every  $n \geq 2$ , when choosing  $C \geq c_1 + c_2$
- **Observe:** This implies  $T(n) \in O(n \log n)$  (important)

**Asymptotic notation allows us to chose arbitrary base case**

# A Strange Problem

# A Strange Problem

**Example:** Give an upper bound on the recurrence

# A Strange Problem

**Example:** Give an upper bound on the recurrence

$$T(1) = 1$$

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



# A Strange Problem

**Example:** Give an upper bound on the recurrence

$$T(1) = 1$$

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

**Step 1: Guess correct solution**

# A Strange Problem

**Example:** Give an upper bound on the recurrence

$$T(1) = 1$$

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

**Step 1: Guess correct solution**  $T(n) \leq f(n) := Cn$

# A Strange Problem

**Example:** Give an upper bound on the recurrence

$$T(1) = 1$$

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

**Step 1: Guess correct solution**  $T(n) \leq f(n) := Cn$

**Step 2: Verify the solution**

# A Strange Problem

**Example:** Give an upper bound on the recurrence

$$T(1) = 1$$

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

**Step 1: Guess correct solution**  $T(n) \leq f(n) := Cn$

**Step 2: Verify the solution**

$$T(n)$$

# A Strange Problem

**Example:** Give an upper bound on the recurrence

$$T(1) = 1$$

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

**Step 1: Guess correct solution**  $T(n) \leq f(n) := Cn$

**Step 2: Verify the solution**

$$T(n) \leq C\lceil n/2 \rceil + C\lfloor n/2 \rfloor + 1$$

# A Strange Problem

**Example:** Give an upper bound on the recurrence

$$T(1) = 1$$

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

**Step 1: Guess correct solution**  $T(n) \leq f(n) := Cn$

**Step 2: Verify the solution**

$$T(n) \leq C\lceil n/2 \rceil + C\lfloor n/2 \rfloor + 1 = Cn + 1$$

# A Strange Problem

**Example:** Give an upper bound on the recurrence

$$T(1) = 1$$

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

**Step 1: Guess correct solution**  $T(n) \leq f(n) := Cn$

**Step 2: Verify the solution**

$$T(n) \leq C\lceil n/2 \rceil + C\lfloor n/2 \rfloor + 1 = Cn + 1 \not\leq f(n) \quad \times$$

# A Strange Problem

**Example:** Give an upper bound on the recurrence

$$T(1) = 1$$

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

**Step 1: Guess correct solution**  $T(n) \leq f(n) := Cn$

**Step 2: Verify the solution**

$$T(n) \leq C\lceil n/2 \rceil + C\lfloor n/2 \rfloor + 1 = Cn + 1 \not\leq f(n) \quad \times$$

- We need a different guess



# A Strange Problem

**Example:** Give an upper bound on the recurrence

$$T(1) = 1$$

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

**Step 1: Guess correct solution**  $T(n) \leq f(n) := Cn$

**Step 2: Verify the solution**

$$T(n) \leq C\lceil n/2 \rceil + C\lfloor n/2 \rfloor + 1 = Cn + 1 \not\leq f(n) \quad \times$$

- We need a different guess
- Let's try:  $f_1(n) := Cn + 1$  and  $f_2(n) := Cn - 1$

# A Strange Problem

**Example:** Give an upper bound on the recurrence

$$T(1) = 1$$

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

**Step 1: Guess correct solution**  $T(n) \leq f(n) := Cn$

**Step 2: Verify the solution**

$$T(n) \leq C\lceil n/2 \rceil + C\lfloor n/2 \rfloor + 1 = Cn + 1 \not\leq f(n) \quad \times$$

- We need a different guess
- Let's try:  $f_1(n) := Cn + 1$  and  $f_2(n) := Cn - 1$

$$f_1 : T(n)$$

# A Strange Problem

**Example:** Give an upper bound on the recurrence

$$T(1) = 1$$

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

**Step 1: Guess correct solution**  $T(n) \leq f(n) := Cn$

**Step 2: Verify the solution**

$$T(n) \leq C\lceil n/2 \rceil + C\lfloor n/2 \rfloor + 1 = Cn + 1 \not\leq f(n) \quad \times$$

- We need a different guess
- Let's try:  $f_1(n) := Cn + 1$  and  $f_2(n) := Cn - 1$

$$f_1 : T(n) \leq C\lceil n/2 \rceil + 1 + C\lfloor n/2 \rfloor + 1 + 1$$

# A Strange Problem

**Example:** Give an upper bound on the recurrence

$$T(1) = 1$$

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

**Step 1: Guess correct solution**  $T(n) \leq f(n) := Cn$

**Step 2: Verify the solution**

$$T(n) \leq C\lceil n/2 \rceil + C\lfloor n/2 \rfloor + 1 = Cn + 1 \not\leq f(n) \quad \times$$

- We need a different guess
- Let's try:  $f_1(n) := Cn + 1$  and  $f_2(n) := Cn - 1$

$$f_1 : T(n) \leq C\lceil n/2 \rceil + 1 + C\lfloor n/2 \rfloor + 1 + 1 = Cn + 3$$

# A Strange Problem

**Example:** Give an upper bound on the recurrence

$$T(1) = 1$$

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

**Step 1: Guess correct solution**  $T(n) \leq f(n) := Cn$

**Step 2: Verify the solution**

$$T(n) \leq C\lceil n/2 \rceil + C\lfloor n/2 \rfloor + 1 = Cn + 1 \not\leq f(n) \quad \times$$

- We need a different guess
- Let's try:  $f_1(n) := Cn + 1$  and  $f_2(n) := Cn - 1$

$$f_1 : T(n) \leq C\lceil n/2 \rceil + 1 + C\lfloor n/2 \rfloor + 1 + 1 = Cn + 3 \not\leq f_1(n)$$

# A Strange Problem

**Example:** Give an upper bound on the recurrence

$$T(1) = 1$$

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

**Step 1: Guess correct solution**  $T(n) \leq f(n) := Cn$

**Step 2: Verify the solution**

$$T(n) \leq C\lceil n/2 \rceil + C\lfloor n/2 \rfloor + 1 = Cn + 1 \not\leq f(n) \quad \times$$

- We need a different guess
- Let's try:  $f_1(n) := Cn + 1$  and  $f_2(n) := Cn - 1$

$$f_1 : T(n) \leq C\lceil n/2 \rceil + 1 + C\lfloor n/2 \rfloor + 1 + 1 = Cn + 3 \not\leq f_1(n) \quad \times$$

# A Strange Problem

**Example:** Give an upper bound on the recurrence

$$T(1) = 1$$

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

**Step 1: Guess correct solution**  $T(n) \leq f(n) := Cn$

**Step 2: Verify the solution**

$$T(n) \leq C\lceil n/2 \rceil + C\lfloor n/2 \rfloor + 1 = Cn + 1 \not\leq f(n) \quad \times$$

- We need a different guess
- Let's try:  $f_1(n) := Cn + 1$  and  $f_2(n) := Cn - 1$

$$f_1 : T(n) \leq C\lceil n/2 \rceil + 1 + C\lfloor n/2 \rfloor + 1 + 1 = Cn + 3 \not\leq f_1(n) \quad \times$$

$$f_2 : T(n)$$

# A Strange Problem

**Example:** Give an upper bound on the recurrence

$$T(1) = 1$$

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

**Step 1: Guess correct solution**  $T(n) \leq f(n) := Cn$

**Step 2: Verify the solution**

$$T(n) \leq C\lceil n/2 \rceil + C\lfloor n/2 \rfloor + 1 = Cn + 1 \not\leq f(n) \quad \times$$

- We need a different guess
- Let's try:  $f_1(n) := Cn + 1$  and  $f_2(n) := Cn - 1$

$$f_1 : T(n) \leq C\lceil n/2 \rceil + 1 + C\lfloor n/2 \rfloor + 1 + 1 = Cn + 3 \not\leq f_1(n) \quad \times$$

$$f_2 : T(n) \leq C\lceil n/2 \rceil - 1 + C\lfloor n/2 \rfloor - 1 + 1$$



# A Strange Problem

**Example:** Give an upper bound on the recurrence

$$T(1) = 1$$

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

**Step 1: Guess correct solution**  $T(n) \leq f(n) := Cn$

**Step 2: Verify the solution**

$$T(n) \leq C\lceil n/2 \rceil + C\lfloor n/2 \rfloor + 1 = Cn + 1 \not\leq f(n) \quad \times$$

- We need a different guess
- Let's try:  $f_1(n) := Cn + 1$  and  $f_2(n) := Cn - 1$

$$f_1 : T(n) \leq C\lceil n/2 \rceil + 1 + C\lfloor n/2 \rfloor + 1 + 1 = Cn + 3 \not\leq f_1(n) \quad \times$$

$$f_2 : T(n) \leq C\lceil n/2 \rceil - 1 + C\lfloor n/2 \rfloor - 1 + 1 = Cn - 1$$

# A Strange Problem

**Example:** Give an upper bound on the recurrence

$$T(1) = 1$$

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

**Step 1: Guess correct solution**  $T(n) \leq f(n) := Cn$

**Step 2: Verify the solution**

$$T(n) \leq C\lceil n/2 \rceil + C\lfloor n/2 \rfloor + 1 = Cn + 1 \not\leq f(n) \quad \times$$

- We need a different guess
- Let's try:  $f_1(n) := Cn + 1$  and  $f_2(n) := Cn - 1$

$$f_1 : T(n) \leq C\lceil n/2 \rceil + 1 + C\lfloor n/2 \rfloor + 1 + 1 = Cn + 3 \not\leq f_1(n) \quad \times$$

$$f_2 : T(n) \leq C\lceil n/2 \rceil - 1 + C\lfloor n/2 \rfloor - 1 + 1 = Cn - 1 = f_2(n)$$

# A Strange Problem

**Example:** Give an upper bound on the recurrence

$$T(1) = 1$$

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

**Step 1: Guess correct solution**  $T(n) \leq f(n) := Cn$

**Step 2: Verify the solution**

$$T(n) \leq C\lceil n/2 \rceil + C\lfloor n/2 \rfloor + 1 = Cn + 1 \not\leq f(n) \quad \times$$

- We need a different guess
- Let's try:  $f_1(n) := Cn + 1$  and  $f_2(n) := Cn - 1$

$$f_1 : T(n) \leq C\lceil n/2 \rceil + 1 + C\lfloor n/2 \rfloor + 1 + 1 = Cn + 3 \not\leq f_1(n) \quad \times$$

$$f_2 : T(n) \leq C\lceil n/2 \rceil - 1 + C\lfloor n/2 \rfloor - 1 + 1 = Cn - 1 = f_2(n) \quad \checkmark$$

(holds for every positive  $C$ )

# A Strange Problem (2)

**Verify Base Case for  $f_2$**

## A Strange Problem (2)

### Verify Base Case for $f_2$

- We have:  $T(1) = 1$  and  $f_2(1) = C - 1 \geq T(1)$  for  $C \geq 2$

# A Strange Problem (2)

## Verify Base Case for $f_2$

- We have:  $T(1) = 1$  and  $f_2(1) = C - 1 \geq T(1)$  for  $C \geq 2$
- We thus set the constant  $C$  in  $f_2$  to  $C = 2$

## A Strange Problem (2)

### Verify Base Case for $f_2$

- We have:  $T(1) = 1$  and  $f_2(1) = C - 1 \geq T(1)$  for  $C \geq 2$
- We thus set the constant  $C$  in  $f_2$  to  $C = 2$
- Then  $f_2(n) = 2n - 1 \geq T(n)$  for every  $n \geq 1$

# A Strange Problem (2)

## Verify Base Case for $f_2$

- We have:  $T(1) = 1$  and  $f_2(1) = C - 1 \geq T(1)$  for  $C \geq 2$
- We thus set the constant  $C$  in  $f_2$  to  $C = 2$
- Then  $f_2(n) = 2n - 1 \geq T(n)$  for every  $n \geq 1$
- This implies that  $T(n) \in O(n)$



# A Strange Problem (2)

## Verify Base Case for $f_2$

- We have:  $T(1) = 1$  and  $f_2(1) = C - 1 \geq T(1)$  for  $C \geq 2$
- We thus set the constant  $C$  in  $f_2$  to  $C = 2$
- Then  $f_2(n) = 2n - 1 \geq T(n)$  for every  $n \geq 1$
- This implies that  $T(n) \in O(n)$

## Comments

# A Strange Problem (2)

## Verify Base Case for $f_2$

- We have:  $T(1) = 1$  and  $f_2(1) = C - 1 \geq T(1)$  for  $C \geq 2$
- We thus set the constant  $C$  in  $f_2$  to  $C = 2$
- Then  $f_2(n) = 2n - 1 \geq T(n)$  for every  $n \geq 1$
- This implies that  $T(n) \in O(n)$

## Comments

- Guessing right can be difficult and requires experience

# A Strange Problem (2)

## Verify Base Case for $f_2$

- We have:  $T(1) = 1$  and  $f_2(1) = C - 1 \geq T(1)$  for  $C \geq 2$
- We thus set the constant  $C$  in  $f_2$  to  $C = 2$
- Then  $f_2(n) = 2n - 1 \geq T(n)$  for every  $n \geq 1$
- This implies that  $T(n) \in O(n)$

## Comments

- Guessing right can be difficult and requires experience
- However, recursion tree method can provide a good guess!