

Algorithms

Dr. Khalil Ahmad Yousef

Lecture 3

Reading Assignment:
Read Section 2.3 of *the book*

Course Learning Outcomes

- **Use algorithm design methods**, such as exhaustive search, **divide-and-conquer** and dynamic programming, to develop efficient algorithms.

Designing Algorithms: Techniques/Strategies

- Brute force
- **Divide and conquer**
- Decrease and conquer
- Transform and conquer
- Space and time tradeoffs
- Greedy approach
- Dynamic programming
- Iterative improvement
- Backtracking
- Branch and bound

Divide-and-Conquer

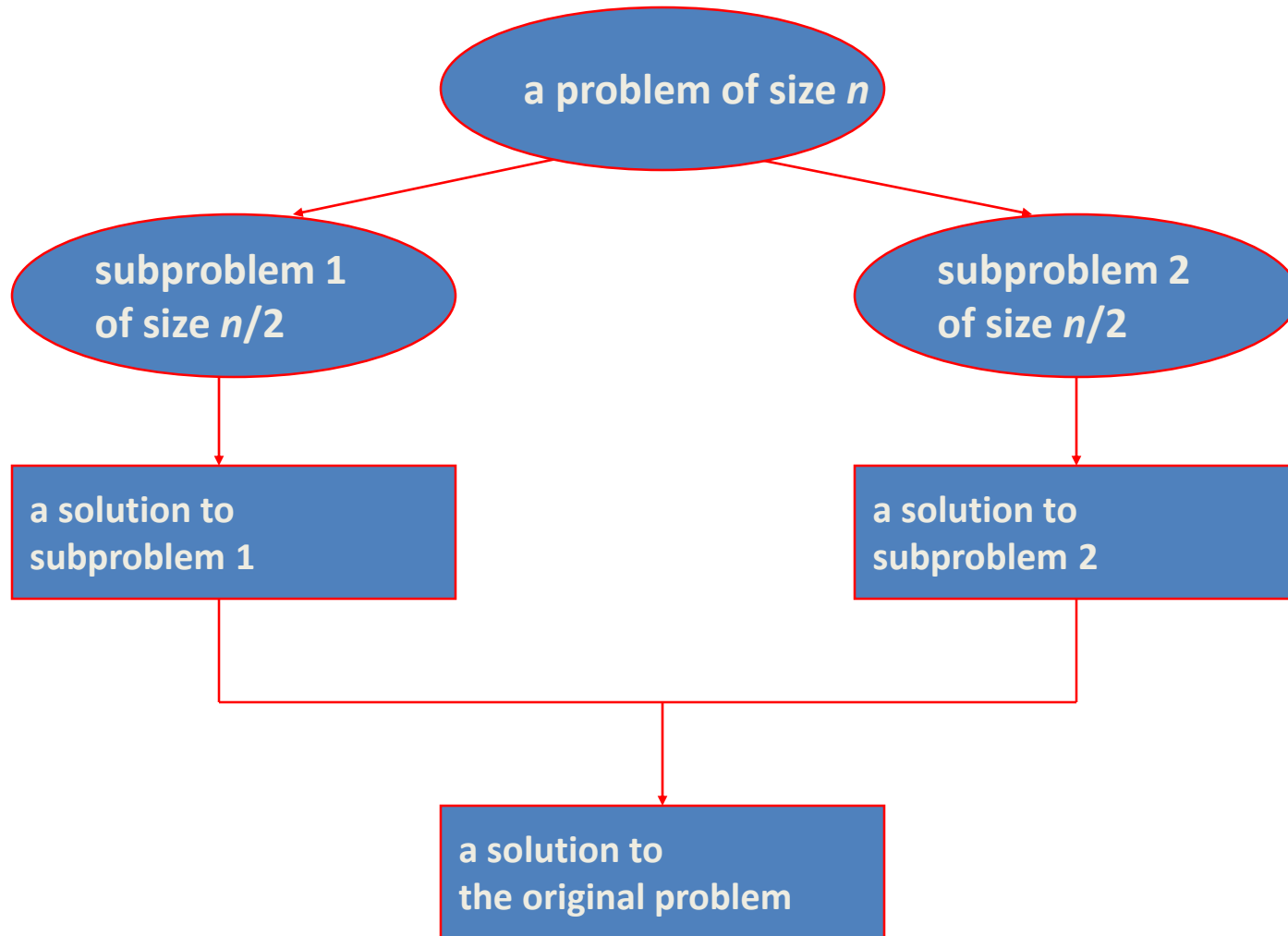
The most-well known algorithm design strategy:

1. Divide instance of problem into two or more smaller instances
2. Solve smaller instances recursively
3. Obtain solution to original (larger) instance by combining these solutions

Divide-and-Conquer Examples

- Sorting: **mergesort** and **quicksort**
- Binary tree traversals
- Multiplication of large integers
- Matrix multiplication: Strassen's algorithm
- Closest-pair and convex-hull algorithms
- Binary search: decrease-by-half (or degenerate divide&conq.)

Divide-and-Conquer Technique (cont.)



MERGE-SORT

- MERGE-SORT is an example of a divide-and-conquer algorithm.

Mergesort

- Split array $A[0..n-1]$ in two about equal halves and make copies of each half in arrays L and R
- Sort arrays L and R recursively
- Merge sorted arrays L and R into array A as follows:
 - Repeat the following until no elements remain in one of the arrays:
 - compare the first elements in the remaining unprocessed portions of the arrays
 - copy the smaller of the two into A, while incrementing the index indicating the unprocessed portion of that array
 - Once all elements in one of the arrays are processed, copy the remaining unprocessed elements from the other array into A.

MERGE-SORT

MERGE-SORT(A, p, r)

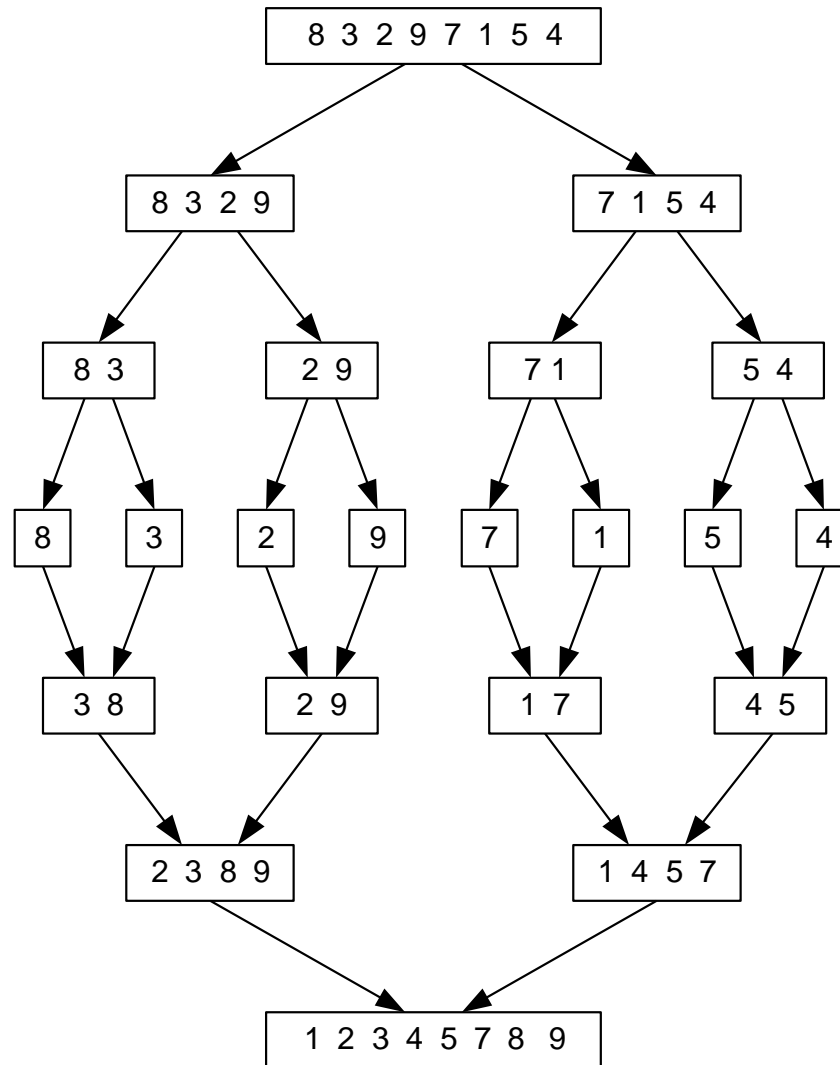
1. **if** $p < r$
2. **then** $q \leftarrow \lfloor \frac{p+r}{2} \rfloor$
3. MERGE-SORT(A, p, q)
4. MERGE-SORT($A, q + 1, r$)
5. MERGE(A, p, q, r)

- **Divide:** Break the problem in half $D(n) = \Theta(1)$.
- **Conquer:** Recursively solve two problems of size $n/2$ in $2T(n/2)$. (Keep dividing until length 1, which is sorted.)
- **Combine:** Merge two sorted arrays into one in $C(n) = \Theta(n)$. (*Keep moving* the smallest element of the two arrays into the result array.)

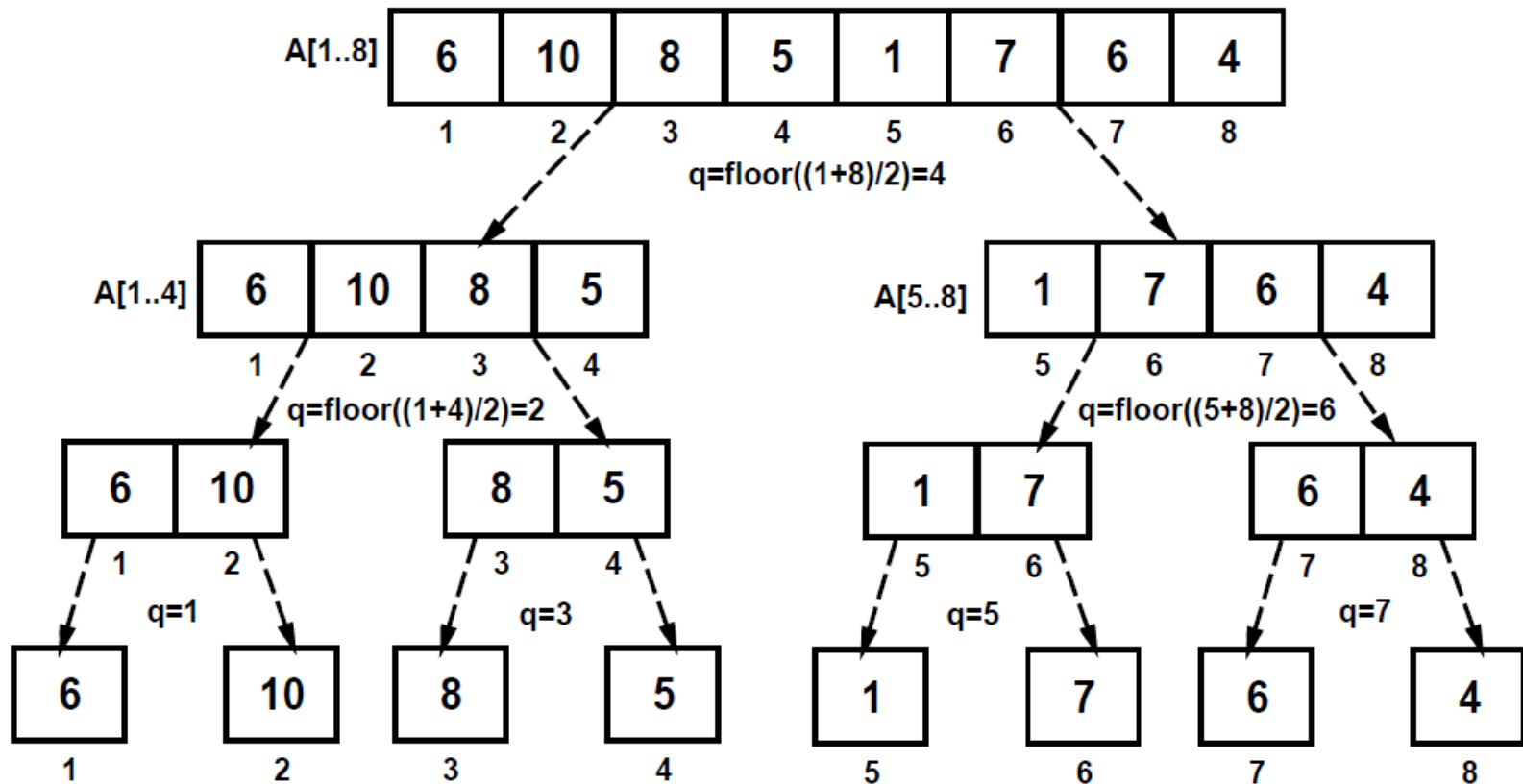
MERGE-SORT

```
MERGE( $A, p, q, r$ )
1   $n_1 \leftarrow q - p + 1$ 
2   $n_2 \leftarrow r - q$ 
3  create arrays  $L[1 \dots n_1 + 1]$  and  $R[1 \dots n_2 + 1]$ 
4  for  $i \leftarrow 1$  to  $n_1$ 
5      do  $L[i] \leftarrow A[p + i - 1]$ 
6  for  $j \leftarrow 1$  to  $n_2$ 
7      do  $R[j] \leftarrow A[q + j]$ 
8   $L[n_1 + 1] \leftarrow \infty$ 
9   $R[n_2 + 1] \leftarrow \infty$ 
10  $i \leftarrow 1$ 
11  $j \leftarrow 1$ 
12 for  $k \leftarrow p$  to  $r$ 
13     do if  $L[i] \leq R[j]$ 
14         then  $A[k] \leftarrow L[i]$ 
15              $i \leftarrow i + 1$ 
16         else  $A[k] \leftarrow R[j]$ 
17              $j \leftarrow j + 1$ 
```

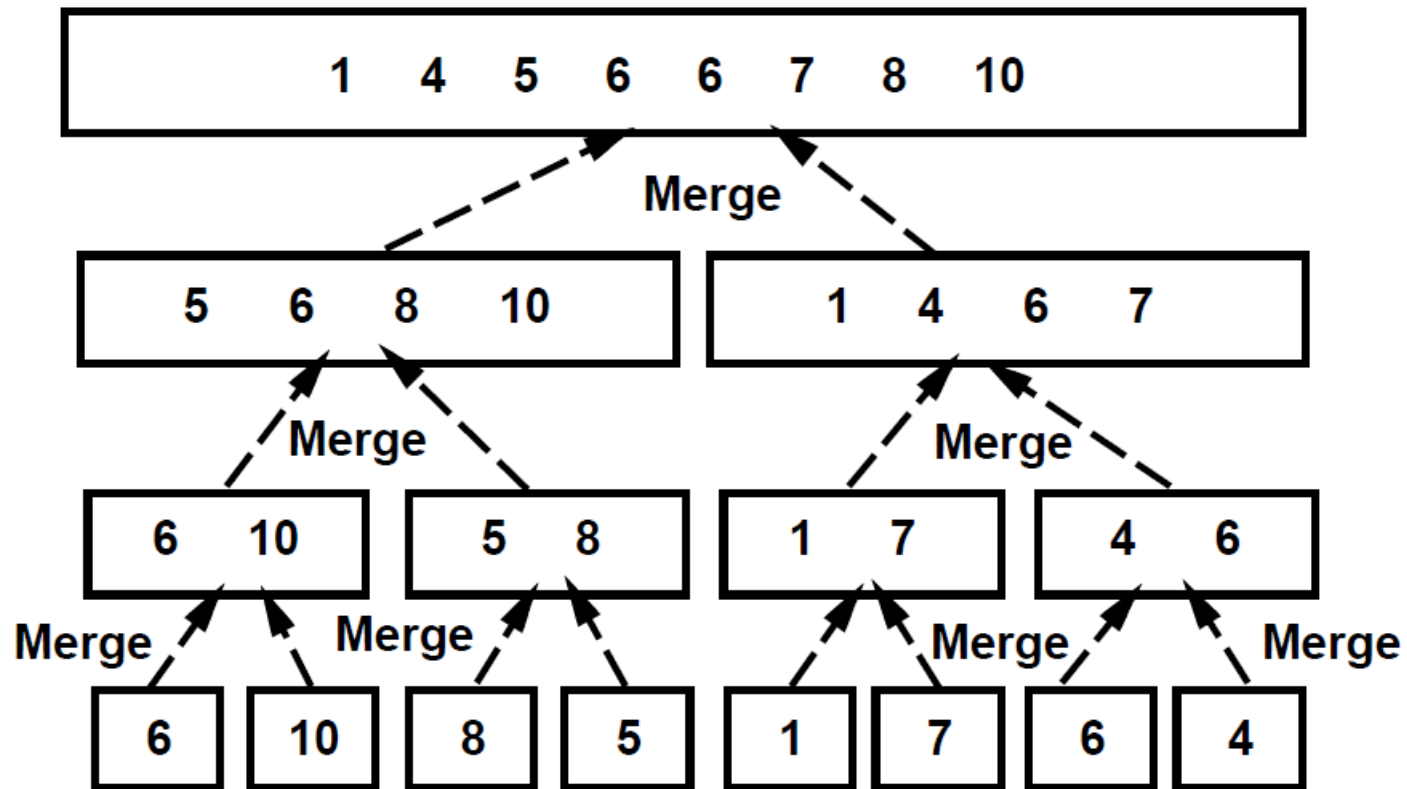
Mergesort Example 1



MERGE-SORT: Example 2: <6, 10, 8, 5, 1, 7, 6, 4>



MERGE-SORT: Example2: <6, 10, 8, 5, 1, 7, 6, 4>



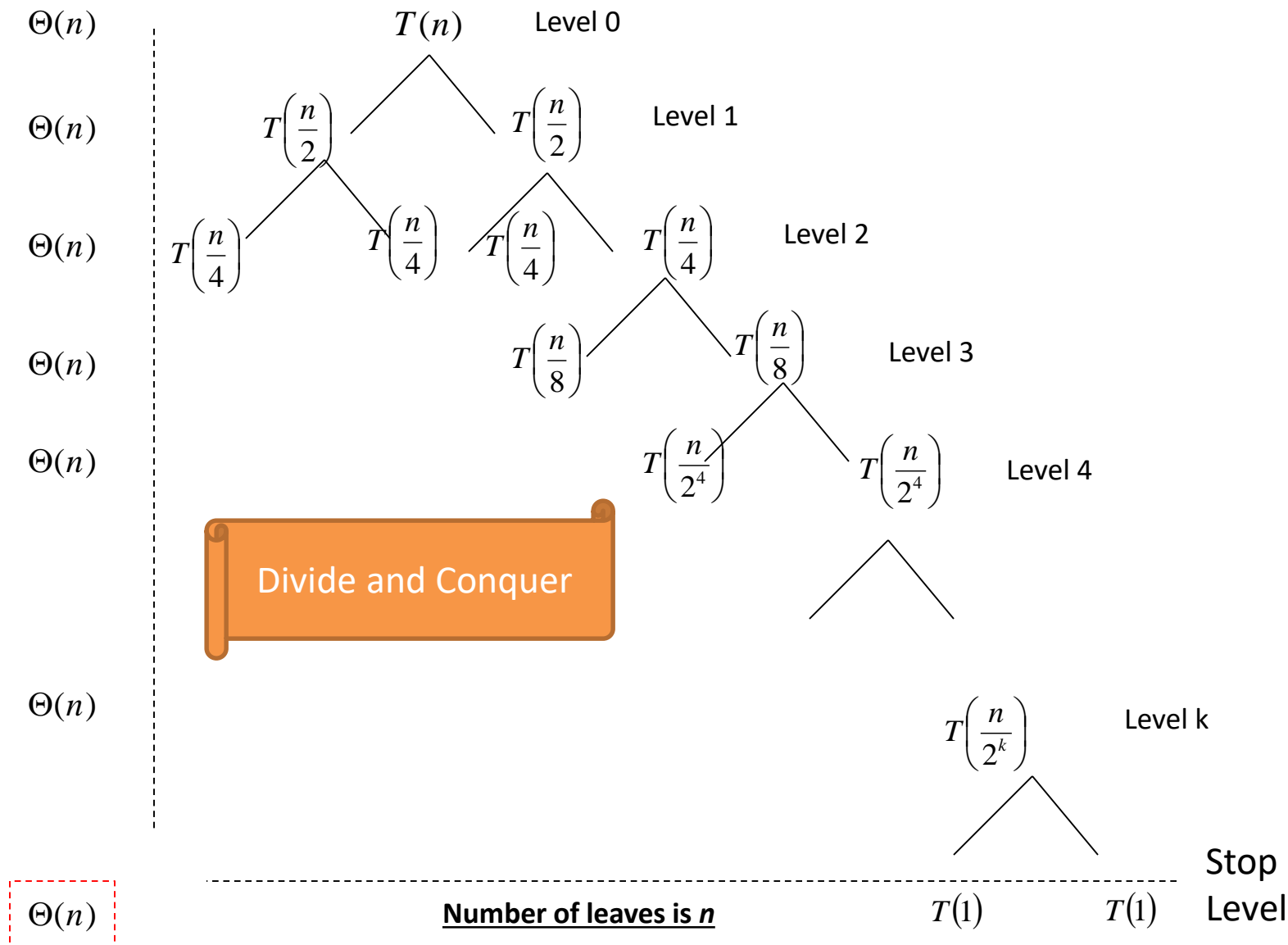
Plan for Analysis of Recursive Algorithms

- Decide on a parameter indicating an input's size.
- Identify the algorithm's basic operation.
- Check whether the number of times the basic op. is executed may vary on different inputs of the same size. (If it may, the worst, average, and best cases must be investigated separately.)
- Set up a recurrence relation with an appropriate initial condition expressing the number of times the basic op. is executed.
- Solve the recurrence (or, at the very least, establish its solution's order of growth) by backward substitutions or another method.

Running Time of Divide and Conquer Algorithms

- We use recurrence equations to analyze the running time of a recursive algorithm.
- Let $T(n)$ be the running time.
 - Divide into a subproblems of size $n/b \rightarrow$ the running time of the subproblems is $aT(n/b)$.
 - Division takes $D(n)$ time.
 - Combining takes $C(n)$ time.

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \leq c, \\ aT(\frac{n}{b}) + D(n) + C(n) & \text{otherwise.} \end{cases}$$



Work Load = work done at each level * num of levels
 $= \Theta(n \lg n)$

$$\frac{n}{2^k} = 1$$

$$\Rightarrow 2^k = n,$$

$\therefore k = \lg n$ Number of levels

MERGE-SORT Analysis

For Merge-Sort, the running time is described by the following:

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1, \\ 2T(\frac{n}{2}) + \Theta(n) & n > 1. \end{cases}$$

We will show that $T(n)$ is $\Theta(n \lg n)$, and so it is asymptotically faster than INSERTION-SORT.

Mathematical Induction

- A general approach to verify a bound **mathematically**
- Verification of the Guess
 - Base case or basis
 - Inductive Hypothesis
 - Maintenance
 - Inductive step
 - Termination

Mathematical Induction

- Claim: $T(n)$ is true for all $n \geq k$
 - Basis:
 - Show formula is true when $n = k$
 - Inductive hypothesis:
 - Assume formula is true for an arbitrary n
 - Step:
 - Show that formula is then true for $n+1$
- Let's now see some mathematical examples...

Induction Example 1

Gaussian Closed Form

- Prove $1 + 2 + 3 + \dots + n = n(n+1) / 2$

– Basis:

- If $n = 0$, then $0 = 0(0+1) / 2$

– Inductive hypothesis:

- Assume $1 + 2 + 3 + \dots + n = n(n+1) / 2$

– Step (show true for $n+1$):

$$\begin{aligned} 1 + 2 + \dots + n + n+1 &= (1 + 2 + \dots + n) + (n+1) \\ &= n(n+1)/2 + n+1 = [n(n+1) + 2(n+1)]/2 \\ &= (n+1)(n+2)/2 = (n+1)(n+1 + 1) / 2 \end{aligned}$$

Induction Example 2

Geometric Closed Form

- Prove $a^0 + a^1 + \dots + a^n = (a^{n+1} - 1)/(a - 1)$ for all $a \neq 1$
 - **Basis:** show that $a^0 = (a^{0+1} - 1)/(a - 1)$
$$a^0 = 1 = (a^1 - 1)/(a - 1)$$
 - **Inductive hypothesis:**
 - Assume $a^0 + a^1 + \dots + a^n = (a^{n+1} - 1)/(a - 1)$
 - **Step (show true for $n+1$):**
$$\begin{aligned} a^0 + a^1 + \dots + a^{n+1} &= a^0 + a^1 + \dots + a^n + a^{n+1} \\ &= (a^{n+1} - 1)/(a - 1) + a^{n+1} = (a^{n+1+1} - 1)/(a - 1) \end{aligned}$$

- Let's now see some mathematical examples related to our course of study ...
 - Merge Sort

Induction Example 3

- Mathematical Induction
 - Merge sort: $T(n) = 2T(n/2) + \Theta(n)$, $T(2)=2$
 - Guess $T(n)$ is $\Theta(n \lg n)$
 - Verify the guess by induction

Merge sort: $T(n) = 2T(n/2) + n$

- Use Mathematical induction to show that when n is exact power of 2, the solution of the recurrence is $\Theta(n \lg n)$
- Thus, we want to show that $T(n) \leq c n \lg n$ (it is sufficient for us to show it is Big-oh. The same analysis can be applied to $\Omega()$).
- Base Step: $T(2) = 2$
 - $T(2) \leq c 2 \lg 2 \quad \Rightarrow \quad 2 \leq c(2) \rightarrow c \geq 1$
 - Thus, let $c = 1$
- Inductive Hypothesis
 - Assume $T(n) \leq c n \lg n$ for all $n = 2^k, k > 0$
- Inductive step
 - We need to show if $n = 2^{k+1}$ $T(2^{k+1}) \leq c(2^{k+1}) \lg(2^{k+1})$
 - $T(2^{k+1}) = 2T(2^{k+1}/2) + (2^{k+1}) \leq ? c(2^{k+1}) \lg(2^{k+1})$
 - $\rightarrow 2T(2^k) + 2^{k+1} \leq ? c(2^{k+1}) \lg(2^{k+1})$
 - $\rightarrow 2(c 2^k \lg 2^k) + (2^{k+1}) \leq ? c(2^{k+1}) \lg(2^{k+1})$ Let $c=1$
 - $\rightarrow 2^{k+1} \lg 2^k + (2^{k+1}) \leq ? (2^{k+1}) \lg(2^{k+1})$
 - $\rightarrow 2^{k+1}(\lg 2^k + 1) \leq ? (2^{k+1}) \lg(2^{k+1})$
 - $\rightarrow 2^{k+1}(\lg 2^k + \lg 2) \leq ? (2^{k+1}) \lg(2^{k+1})$
 - $\rightarrow 2^{k+1}(\lg(2 * 2^k)) \leq ? (2^{k+1}) \lg(2^{k+1})$
 - $\rightarrow 2^{k+1}(\lg(2^{k+1})) \leq (2^{k+1}) \lg(2^{k+1}),$ when $c = 1$ (done)

Induction Example 4

- Insertion Sort: $T(n) = T(n-1) + n$, $T(1) = 1$
- Guess: Worst case complexity is $O(n^2)$ or $T(n) \leq cn^2$
- **Proof**
 - Base case:
 - $T(1) = 1 \leq c(1^2) = c \rightarrow c \geq 1$
 - Thus, let $c = 1$
 - Inductive Hypothesis
 - Assume $T(k) \leq ck^2$ for all $1 \leq k \leq n$
 - Inductive step
 - We need to show $T(n+1) \leq c(n+1)^2$
 - $T((n+1)-1) + (n+1) \leq ? c(n+1)^2$
 - $T(n) + (n+1) \leq ? c(n+1)^2$
 - $\rightarrow (cn^2) + (n+1) \leq ? c(n+1)^2$
 - $\rightarrow cn^2 + n + 1 \leq c(n+1)^2$ when $c = 1$, note that $(n+1)^2 = n^2 + 2n + 1$

Induction Example 5

- Some algorithm: $T(n) = 4T(n/2) + n$, $T(1) = 1$
- Guess $T(n) = O(n^2)$ and solve by induction for all $n \in N$ (Natural Numbers)
 - First we need to put $T(n) \in N$
 - Expand the big Oh notation.
 - Assume $T(n) \leq c n^2$ for all $n \in N$
 - Base case: ...

Induction Example 5 (Cont...)

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

- Base case:
 - $T(1) = 1 = c(1^2) = c \rightarrow c \geq 1$
 - Thus, let $c = 1$
- Inductive Hypothesis
 - Assume $T(k) \leq ck^2$ for all $1 \leq k \leq n$
- Inductive step
 - We need to show $T(n+1) \leq c(n+1)^2$
 - $4T((n+1)/2) + (n+1) \leq c(n+1)^2$
 - $\rightarrow 4(c((n+1)/2)^2) + (n+1) \leq c(n+1)^2$
 - $\rightarrow c(n+1)^2 + \underline{(n+1)} \leq c(n+1)^2$
 - $\underline{(n+1)} \leq 0$ (Contradiction)
 - Therefore, let's modify or fix our inductive hypothesis

Induction Example 5 (Cont...)

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

- Base case:
 - $T(1) = 1 = c(1^2) = c \rightarrow c \geq 1$
 - Thus, let $c = 1$
- Inductive Hypothesis
 - Add a leading term
 - Assume $T(k) \leq ck^2 - c'k$ for all $1 \leq k \leq n$
- Inductive step
 - We need to show $T(n+1) \leq c(n+1)^2 - c'(n+1)$
 - $4T((n+1)/2) + (n+1) \leq c(n+1)^2 - c'(n+1)$
 - $\rightarrow 4(c((n+1)/2)^2 - c'((n+1)/2)) + (n+1) \leq c(n+1)^2 - c'(n+1)$
 - $\rightarrow c(n+1)^2 - 4c'(n+1) + (n+1) \leq c(n+1)^2 - c'(n+1)$
 - $\rightarrow c(n+1)^2 - (4c'-1)(n+1) \leq c(n+1)^2 - c'(n+1)$
 - $\rightarrow (4c'-1)(n+1) \geq c'(n+1)$
 - $\rightarrow (4c'-1) \geq c' \quad \rightarrow 3c'-1 \geq 0 \quad \rightarrow 3c' \geq 1$
 - » $c' \geq 1/3$ (Just pick $c' = 1/3$. Done)

Up Next

- Chapter 4: Solving recurrences
 - Substitution method
 - Master theorem