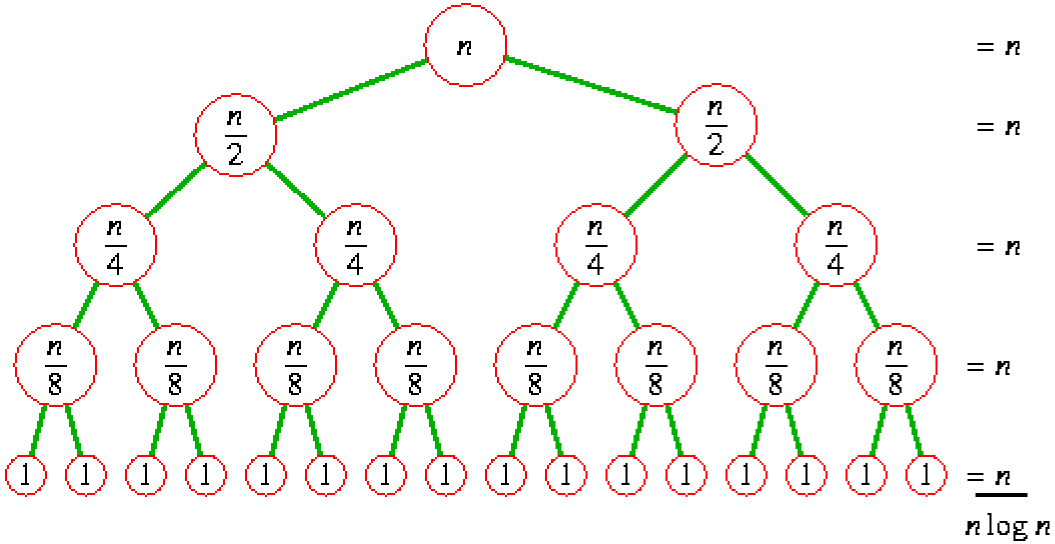
DSE 2256 DESIGN & ANALYSIS OF ALGORITHMS

Lecture 17

Divide-and-Conquer:

Introduction to Divide-and-Conquer,
Master Theorem
Merge sort



Instructors:

Dr. Savitha G,
Assistant Professor, DSCA, MIT, Manipal

Dr. Abhilash K. Pai,
Assistant Professor, DSCA, MIT, Manipal

Recap of L16

Decrease-and-Conquer:

- Directed Acyclic Graph (DAG)
- Topological sorting
  - DSA based topological sorting
  - Source based topological sorting

Divide-and-Conquer

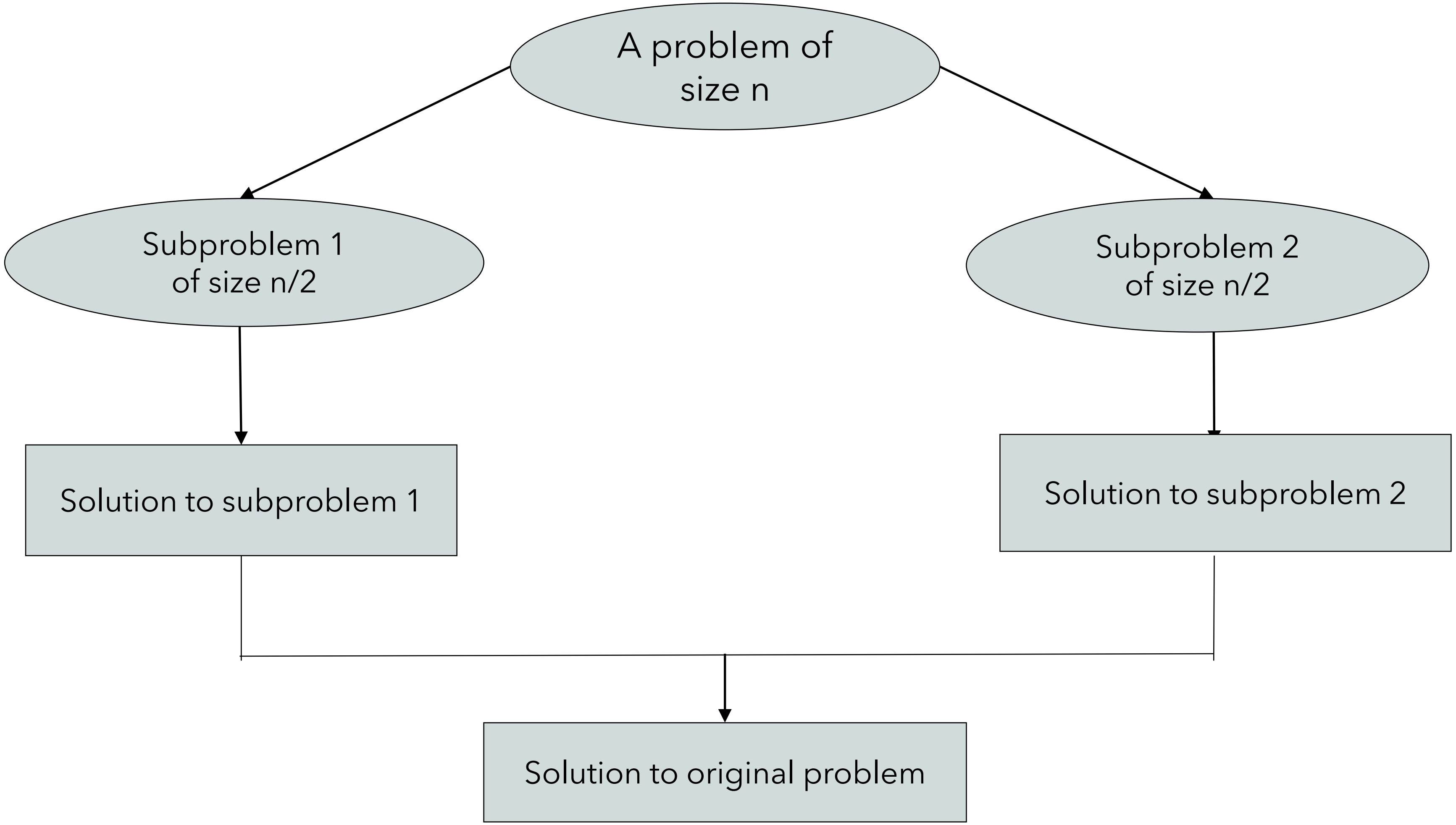
The algorithm design strategy:

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



Divide-and-Conquer: Examples

- Sorting: Merge sort and Quicksort
- Binary tree traversals
- Binary search, Multiplication of large integers
- Matrix multiplication: Strassen's algorithm

General Divide-and-Conquer Recurrence

- General Divide-and-Conquer recurrence:

  $T(n) = a\,T(n/b) + f(n)$, where $f(n) \in \theta(n^d)$, $d \geq 0$, $a>=1$, $b>1$

- Order of growth of solution T(n) depends:

  i. On the values of constants a and b

  ii. Order of growth of function f(n)

Master Theorem: (To solve the divide-and-conquer recurrence relation w.r.t orders of growth)

Case 1: If $a < b^d$, $T(n) \in \Theta(n^d)$

Case 2: If $a = b^d$, $T(n) \in \Theta(n^d \log n)$

Case 3: If $a > b^d$, $T(n) \in \Theta(n^{\log_b a})$

Examples:

1. $T(n) = 4T(n/2) + n$ $\quad T(n) \in ?$
2. $T(n) = 4T(n/2) + n^2$ $\quad T(n) \in ?$
3. $T(n) = 4T(n/2) + n^3$ $\quad T(n) \in ?$

Merge sort

1. Split array A[0..n-1] into about equal halves and make copies of each half in arrays B and C
2. Sort arrays B and C recursively
3. Merge the sorted arrays B and C into the array A as follows:

   Repeat the following until no elements remain in one of the arrays:

   i. Compare the first elements in the remaining unprocessed portions of the arrays.

   ii. Copy the smaller of the two into A, while incrementing the index indicating the unprocessed portion of that array.

   iii. Once all elements in one of the arrays are processed, copy the remaining unprocessed elements from the other array into A.

Merge sort : Algorithm

ALGORITHM *Mergesort*($A[0..n-1]$)

//Sorts array $A[0..n-1]$ by recursive merge sort

//Input: An array $A[0..n-1]$ of orderable elements

//Output: Array $A[0..n-1]$ sorted in nondecreasing order

if $n > 1$

copy $A[0..n/2-1]$ to $B[0..n/2-1]$

copy $A[n/2..n-1]$ to $C[0..n/2-1]$

Mergesort($B[0..n/2-1]$)

Mergesort($C[0..n/2-1]$)

Merge(B, C, A)

Pseudocode of Merge

ALGORITHM *Merge*($B[0..p-1]$, $C[0..q-1]$, $A[0..p+q-1]$)

//Merges two sorted arrays into one sorted array

//Input: Arrays $B[0..p-1]$ and $C[0..q-1]$ both sorted

//Output: Sorted array $A[0..p+q-1]$ of the elements of B and C

$i \leftarrow 0$; $j \leftarrow 0$; $k \leftarrow 0$

while $i < p$ **and** $j < q$ **do**

    if $B[i] \leq C[j]$

        $A[k] \leftarrow B[i]$; $i \leftarrow i + 1$

    else $A[k] \leftarrow C[j]$; $j \leftarrow j + 1$

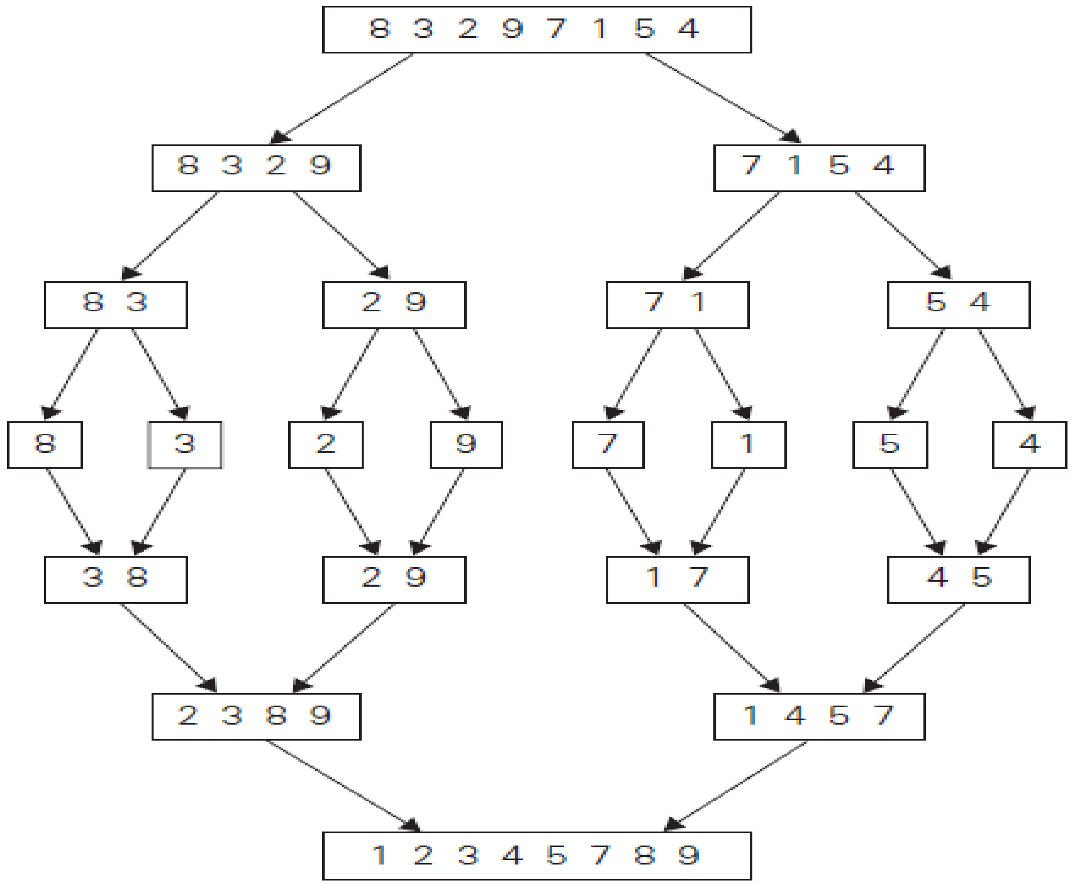
    $k \leftarrow k + 1$

if $i = p$

    copy $C[j..q-1]$ to $A[k..p+q-1]$

else copy $B[i..p-1]$ to $A[k..p+q-1]$

Merge sort: Example



Merge sort: Analysis

- Recurrence relation for Merge Sort: (Assuming that "n" is a power of 2 and n>1)

$C(n) = 2C(n/2) + f(n)$ --- (1)

$C(1) = 0$

Eqn. (1) ➡

$C(n) = 2C(n/2) + C_{merge}(n)$ --- (2)

In worst case:

$C_{merge}(n) = n\text{-}1$

Now, Eqn. (2) ➡

$$C_{worst}(n) = 2C_{worst}(n/2) + n-1 \xrightarrow{\text{Apply Master Theorem}} \approx \boldsymbol{\Theta(n \log n)}$$

Note that:

In the best case (when all of the input elements are sorted):

There will be **n/2 comparisons** made for the "merge" function – this will happen when all elements of the first array is less than the elements of the second array.

Therefore, for best case & worst case:
$C_{merge}(n) \in \Theta(n \log n)$

Thank you!

Any queries?