Introduction to Recursive Formulations for Algorithm Design: I

Partha P Chakrabarti

Indian Institute of Technology Kharagpur

Overview

- ❑ Algorithms and Programs
- ❑ Pseudo-Code
- ❑ Algorithms + Data Structures = Programs
- ❑ Initial Solutions + Analysis + Solution Refinement + Data Structures = Final Algorithm
- ❑ Use of Recursive Definitions as Initial Solutions
- ❑ Recurrence Equations for Proofs and Analysis
- ❑ Solution Refinement through Recursion Transformation and Traversal
- ❑ Data Structures for saving past computation for future use

Time } complexity
Space }

<u>Sample Problems:</u>

1. Finding the Largest
2. Largest and Smallest
3. Largest and Second Largest
4. Fibonacci Numbers
5. Searching for an element in an ordered / unordered List
6. Sorting
7. Pattern Matching
8. Permutations and Combinations
9. Layout and Routing
10. Shortest Paths

First Problem: Largest of a Set of Numbers

Sequential Comparison

$L = \{x_1, x_2, \ldots, x_n\}$ x is an integer

$\max(L)$

$\{$ $L = \{x_1, x_2, \ldots, x_n\}$

if $|L| = 1$ return (x_1)

$L' = L - \{x_1\}$

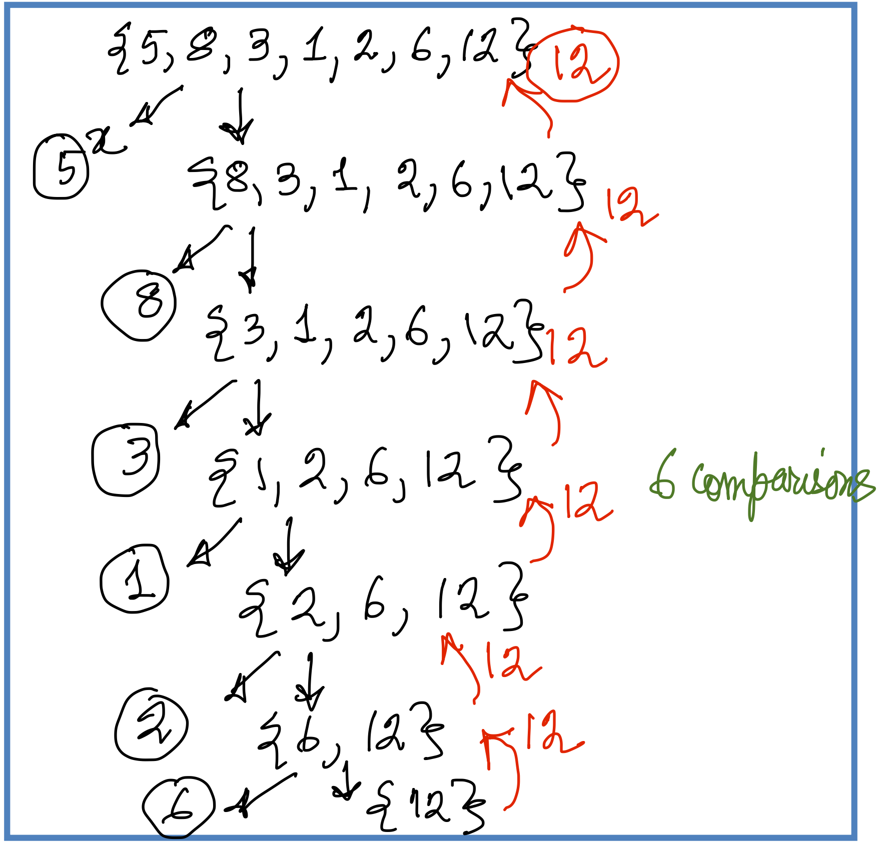
$y = \max(L')$

if $\boxed{(x_1 \geq y)}$ return (x_1)

else (y)

$\}$

$$T(n) = T(n-1) + 1, \; n > 1$$
$$= 0, \; \text{if } n = 1$$
$$= (n-1)$$



Finding Largest: Recursive Formulation

max2(L)

{ $L = \{x_1, x_2, \ldots, x_n\}$

if $|L| = 1$ return (x_1)

split L into 2 non-empty sets L_1, L_2

→ $y_1 = \text{max2}(L_1)$

→ $y_2 = \text{max2}(L_2)$

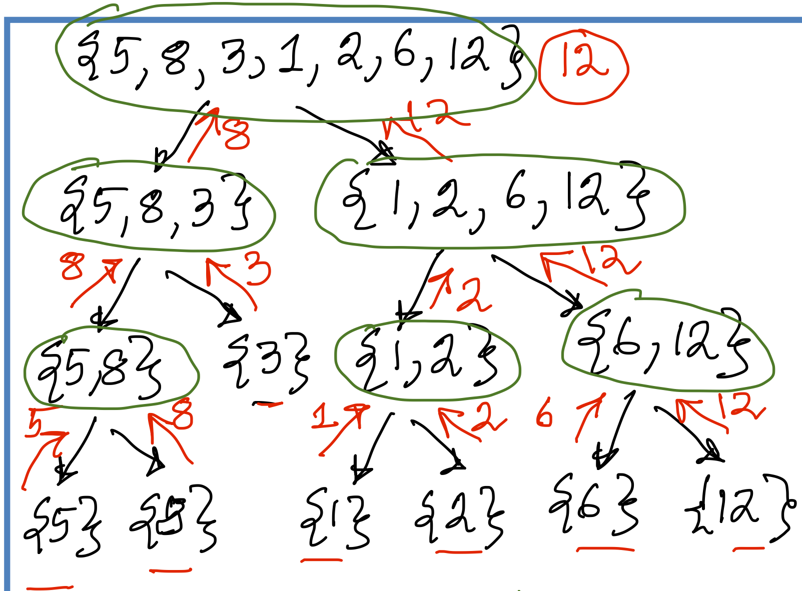
if $\boxed{(y_1 \geq y_2)}$ return (y_1)

else return (y_2)

}

correctness proof by INDUCTION

$$T(n) = T(k) + T(n-k) + 1, \quad n > 1$$
$$= 0, \text{ if } n = 1$$
$$= (n-1)$$



6 comparisons

Largest: Analysis

Proof of Correctness

By induction

Base condition $n = 1$

Inductive condition
correctly for all $n < n_0$
we prove inductively
it is true for $n_0 + 1$

True for all $n \geq 1$

Complexity Analysis

$$T(n) = T(k) + T(n-k) + 1 \quad \text{for } n > 1$$

$$= 0, \quad n = 1$$

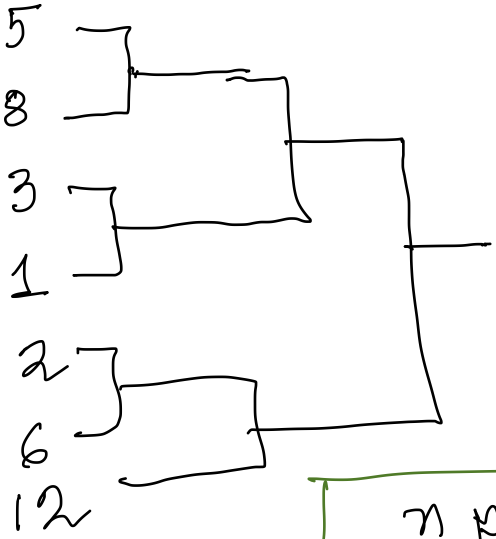
$$T(x) = x - 1$$

$$T(n) = (k-1) + (n-k-1) + 1$$

$$= n - 1$$

Comparison Tournament

5, 8, 3, 1, 2, 6, 12



n players
$(n-1)$ matches

2nd Problem: Largest and Smallest

Sequential Comparison

maxmin (L)

{ let $L = \{x_1, x_2, \ldots, x_n\}$

if $|L| = 1$ return$(\langle x_1, x_1 \rangle)$

$L' = L - \{x_1\}$

$\langle y_1, y_2 \rangle = \text{maxmin}(L')$

if $(x_1 \geq y_1)$ then $m_1 = x_1$

else $m_1 = y_1$

if $(x_1 \leq y_2)$ then $m_2 = x_1$

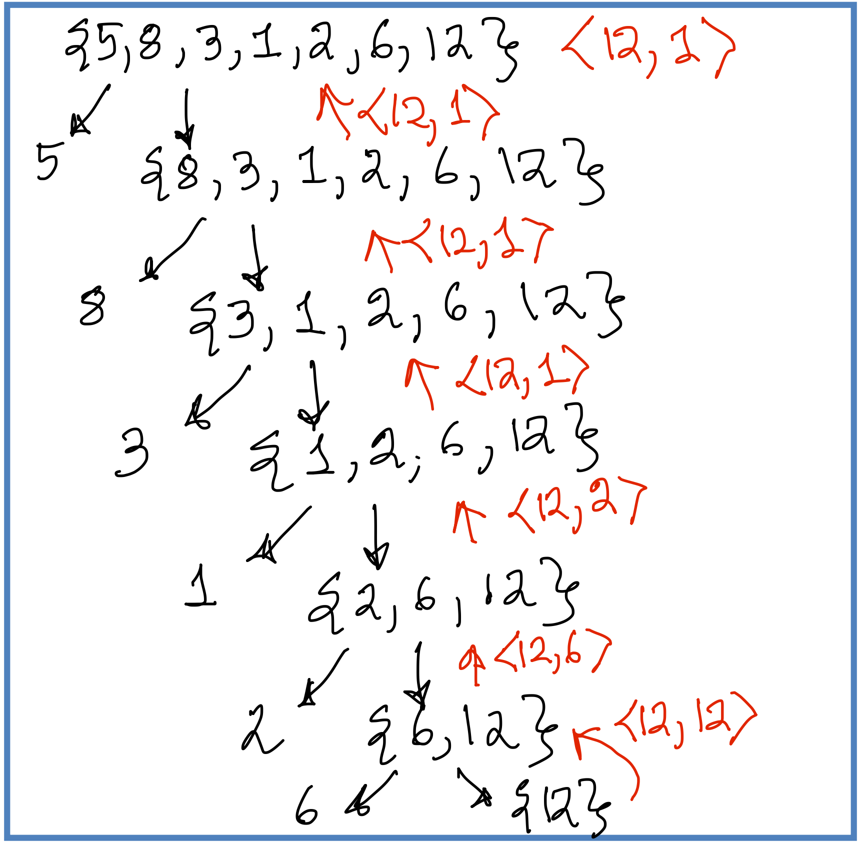
else $m_2 = y_2$

return$(\langle m_1, m_2 \rangle)$

$T(n) = 0$, if $n = 1$

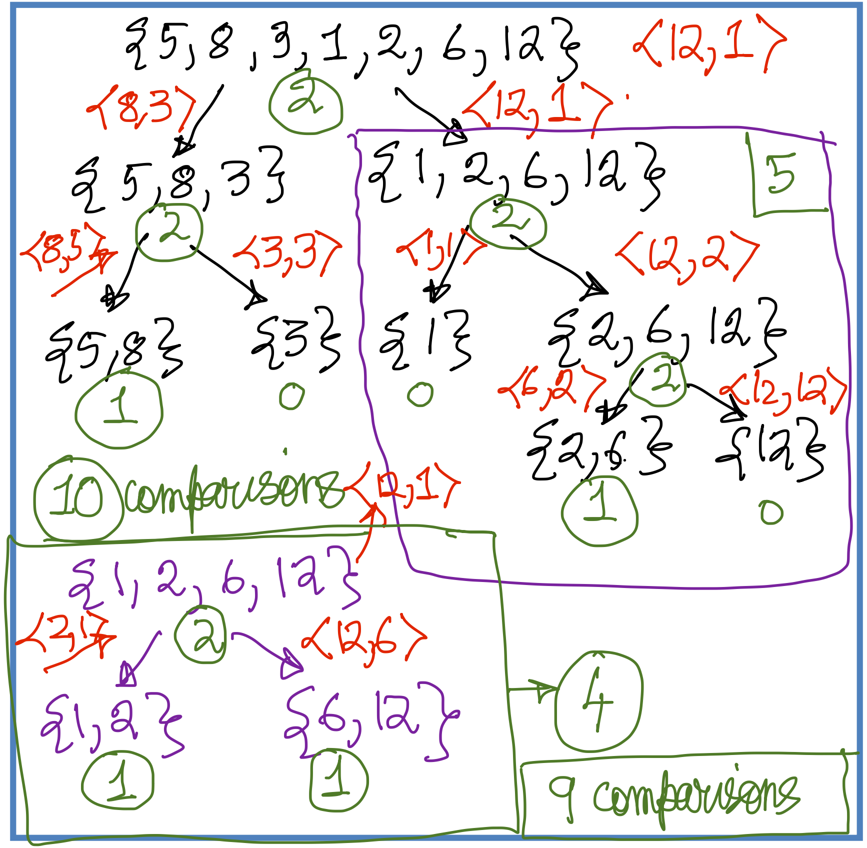
$= T(n-1) + 2, \; n > 1$

$= 2(n-1)$

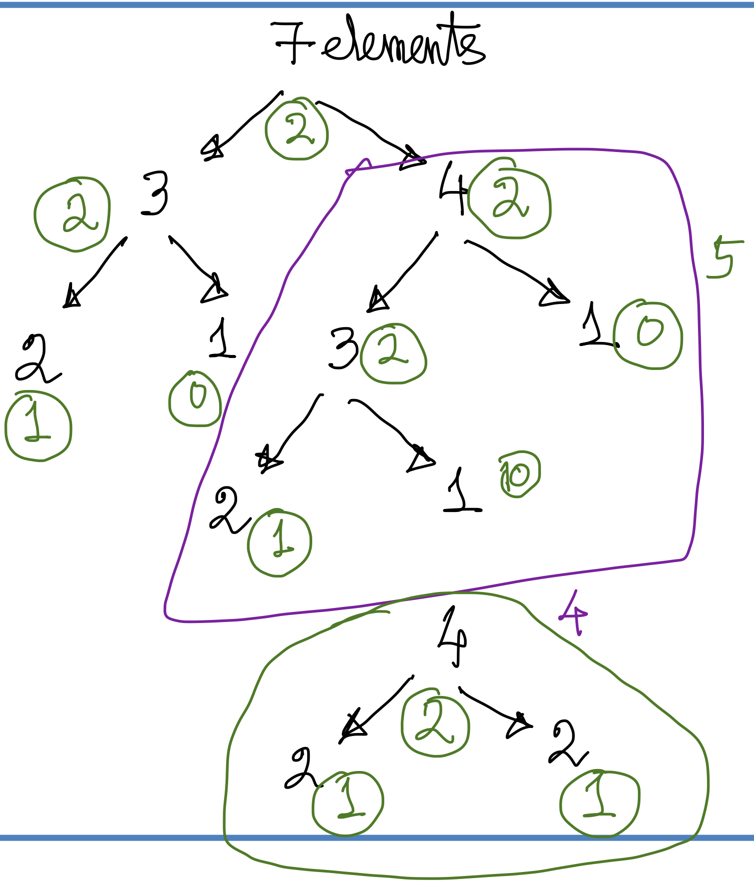


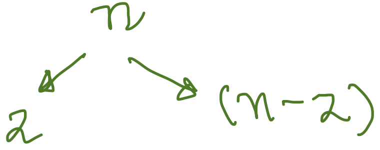
Largest and Smallest: Recursive Definition

```
maxmin2(L)
{ Let L = {x1, x2, ..., xn}
  if |L| = 1, return (<x1, x1>)
  if |L| = 2, if (x1 >= x2)
                 return (<x1, x2>)
            else return (<x2, x1>)
  split L into 2 non-empty sets L1, L2
  <y1, y2> = maxmin2(L1)
  <z1, z2> = maxmin2(L2)
  if (y1 >= z1) m1 = y1 else m1 = z1
  if (y2 <= z2) m2 = y2 else m2 = z2
  return (<m1, m2>)
}
```



Largest & Smallest: Choice of Split





$$T(n) = 0, \text{ if } n = 1$$
$$= 1, \text{ if } n = 2$$
$$= T(k) + T(n-k) + 2, \text{ if } n > 2$$

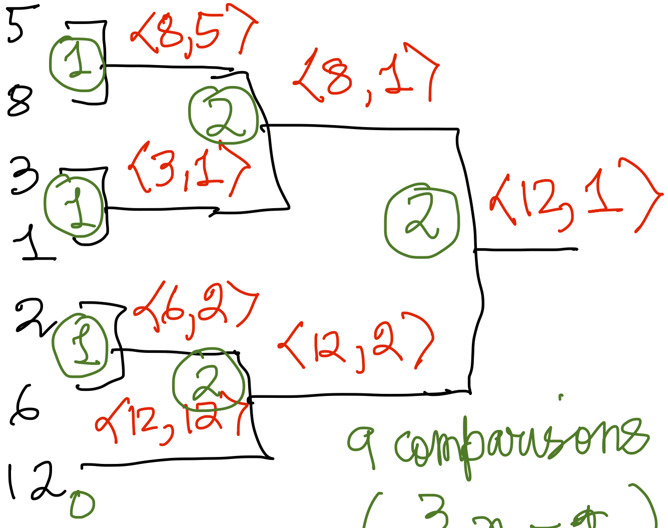
If we choose $k = 1$

$$T(1) + T(n-1) + 2$$
$$= \underline{2n-3} \quad \left(\begin{array}{l}\text{base case of} \\ T(2) = 1\end{array}\right)$$

<u>choose $k = 2$</u>

$$= T(2) + T(n-2) + 2$$
$$= T(2) + T(2) + T(n-4) + 2 + 2$$
$$= \boxed{(3n/2 - 2)}$$

Tournament Based Analysis & Design



9 comparisons

$\left(\frac{3}{2}n - 2\right)$

$\frac{3.7}{2} - 2$

$= 11 - 2$

optimal split ? ✓

$|L| =$ even then split into even, even parts

$4 \rightarrow 3$, $4 \rightarrow 1$ $\qquad$ 2, 2

$|L| =$ odd then split into even, odd

6 elements

$\rightarrow$ 3, 3

$\rightarrow$ 4, 2

$\left\lceil \frac{3n}{2} - 2 \right\rceil$

Summary

- ❑ Algorithms and Programs
- ❑ Pseudo-Code
- ❑ Algorithms + Data Structures = Programs
- ❑ Initial Solutions + Analysis + Solution Refinement + Data Structures = Final Algorithm
- ❑ Use of Recursive Definitions as Initial Solutions
- ❑ Recurrence Equations for Proofs and Analysis
- ❑ Solution Refinement through Recursion Transformation and Traversal
- ❑ Data Structures for saving past computation for future use

<u>Sample Problems:</u>

1. Finding the Largest
2. Largest and Smallest
3. Largest and Second Largest
4. Fibonacci Numbers
5. Searching for an element in an ordered / unordered List
6. Sorting
7. Pattern Matching
8. Permutations and Combinations
9. Layout and Routing
10. Shortest Paths

Thank you

Any Questions?