Design and Analysis of Algorithms

Lecture - 6

Success is always inevitable with Hard Work and Perseverance

N. Ravitha Rajalakshmi

Learning Objective

- Methods to solve recurrence relation

Revisit Recurrence Relation

For finding Factorial of a number , What was recurrence relation?

$$\mathrm{T}(\boldsymbol{n}) = \begin{cases} T(\boldsymbol{n-1}) + 1, & n > 0 \\ 0, & n = 0 \end{cases}$$

What is n ?

Input Size

What does T(0) = 0 indicate?

No computations when input size is zero

What is T(n) ?

Time taken by algorithm for input size n

Solving Recurrences

- Iteration Method
  - Method of forward / backward substitution

- Recursion tree Method

- Master's theorem

Iteration Method

Solve by backward substitution

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

$= (T(n-2) + 1) + 1$

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

$= (T(n-3) + 1) + 2$

$= (T(n-3) + 3)$

$= (T(n-4) + 1) + 3$

$= (T(n-4) + 4)$

Generic Equation = $(T(n-i) + i)$

Use base case to resolve the characteristic equation

$T(0) = 0$

Generic Equation = $(T(n-i) + i)$

$n - i = 0$

$\mathrm{i} = n$

Substitute $\mathrm{i} = n$ in generic equation

$= \mathrm{T}(0) + n$

$= 0 + n$

T(n) $= O(n)$

Iteration Method

Solve by forward substitution

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

$T(1) = T(0) + 1$

= 1

$T(2) = T(1) + 1$

= 2

$T(3) = T(2) + 1$

= 3

2

Recognize the sequence

T(n) = ?

T(n) = n

3

Check whether solution satisfy recurrence

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

T(0) = 0 - 1 + 1 = 0

1

n	T(n)
1	1
2	2
3	3
4	4
5	5

Solve the following recurrence relation using the method of forward substitution

$$T(n) = T(n/2) + \mathrm{n}, n > 0$$

$$T(0) = 0$$

Iteration Method

Solve by forward substitution

$$T(n) = T(n/2) + \mathrm{n}$$

$T(1) = T(0) + 1$

= 1

$T(2) = T(1) + 2$

= 3

$T(3) = T(1) + 3$

= 4

Recognize the sequence

T(n) = ?

T(n) = 2n

Check whether solution satisfy recurrence

T(n) = n + n = 2n

T(0) = 0 + 0 = 0

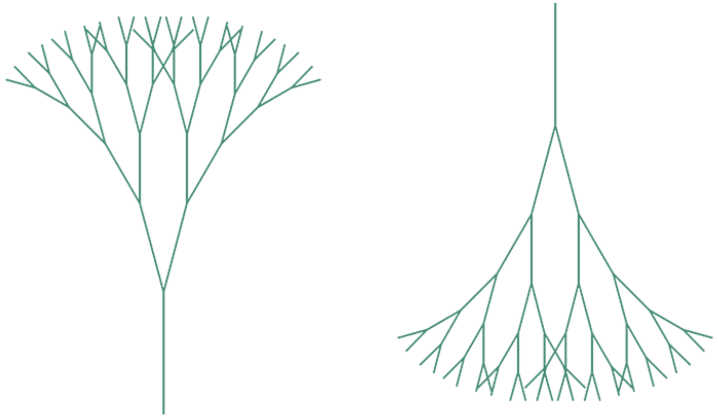
$T(n) = 0(n)$

n	T(n)
1	1
2	3
3	4
4	7
5	8
6	10
7	11
8	15

n	T(n)
16	31
32	63
64	127

Recursion Tree Method

- Visualization tool
- Depicts the number of recursive calls and the amount of work done at each recursive call
- Provides a good guess on time complexity



Recursion Tree Method

$$T(n) = 2T(n/2) + n^2 \quad , \qquad n > 1 \qquad\qquad T(1) = 1$$

Number of recursive calls

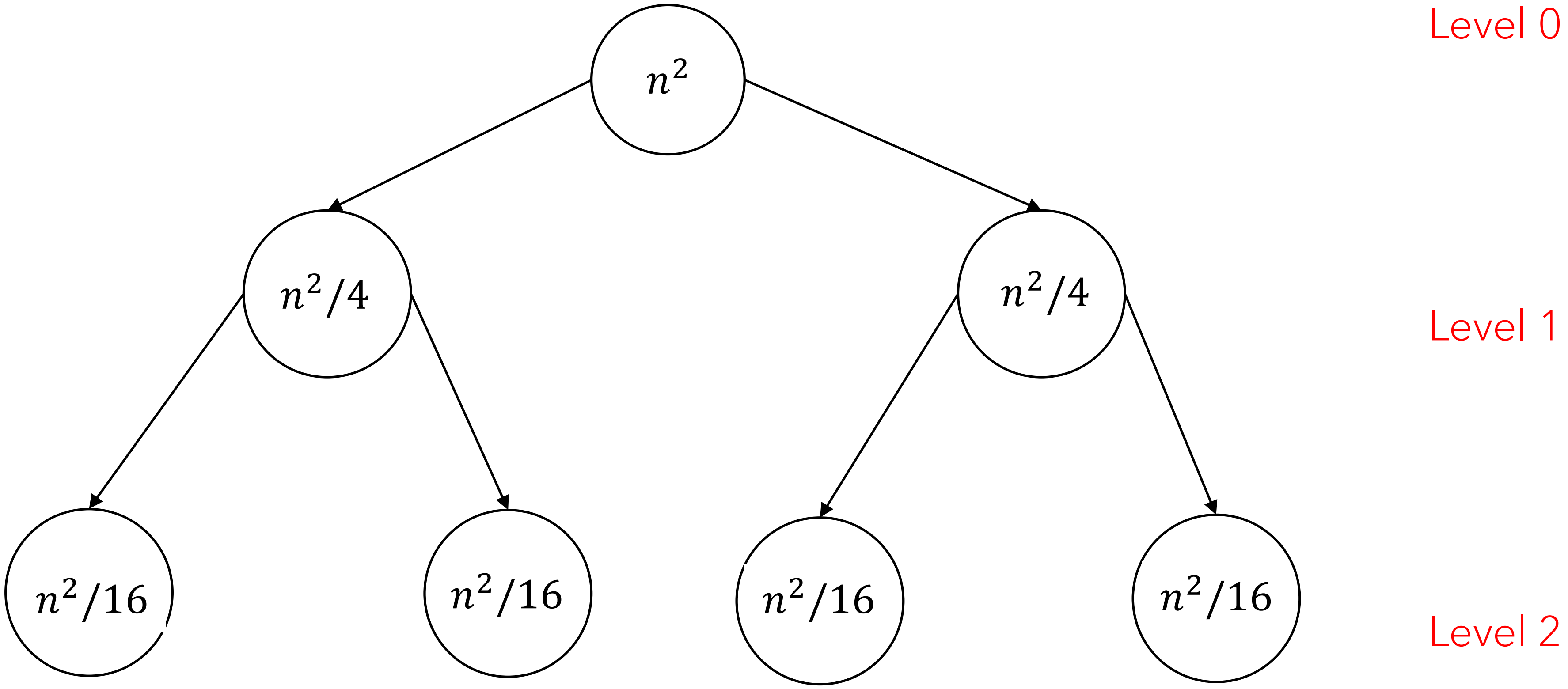
Amount of computation for input size n

Base case input size

Subproblem size

Recursion Tree Method

f(n) = n^2
#recursive calls = 2
Size of sub problem = n/2



Pause & Think

How the recurrence relation components are illustrated in recursion tree

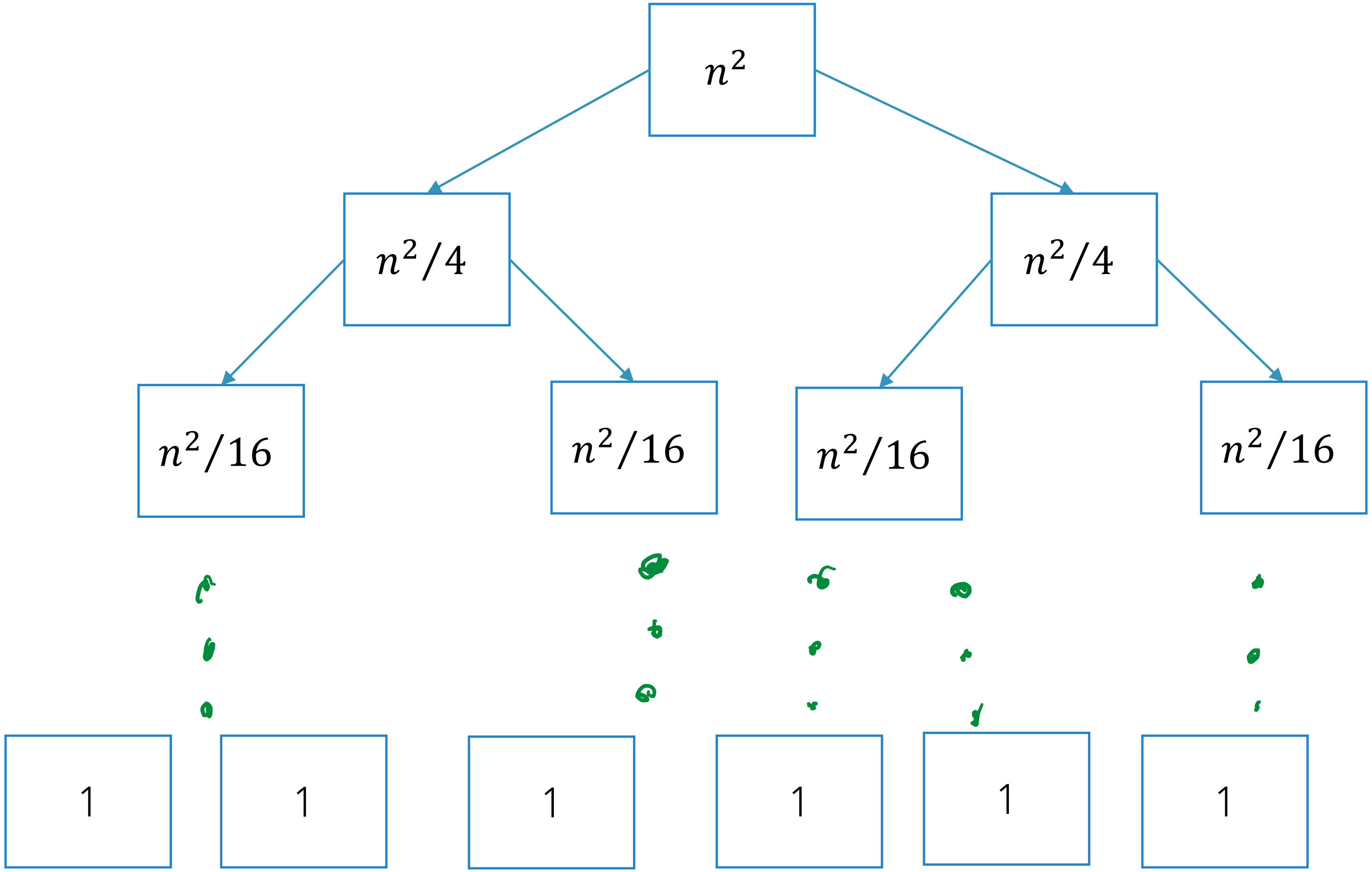
Problem and subproblem	nodes
Recursive calls	Branches
Computation	Value in the node

Pause & Think

How the recurrence relation components are illustrated in recursion tree

Total input size	Value in the root node
Base case	Leaf nodes

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



Problem size	# nodes	Amount of work done
n	1	n^2
$n/2$	2	$2\,n^2/4 = n^2/2$
$n/4$	4	$4\,n^2/16 = n^2/4$
?	?	?

How Problem size related to level ?

$$n/2^{\mathrm{i}}$$

How number of nodes related to level?

$$2^{\mathrm{i}}$$

Leaf nodes

Problem size is reduced by half after every iteration, at ith level problem size is

$$\frac{n}{2^{\mathrm{i}}}$$

At leaf node, what is the problem size ?

$$n/_{2^{\mathrm{i}}} = 1$$

$$2^{\mathrm{i}} = n$$

$$i = \log n$$

How many nodes at leaf level ?

At every layer its doubled , 2^{i} So at base level , there will be n nodes

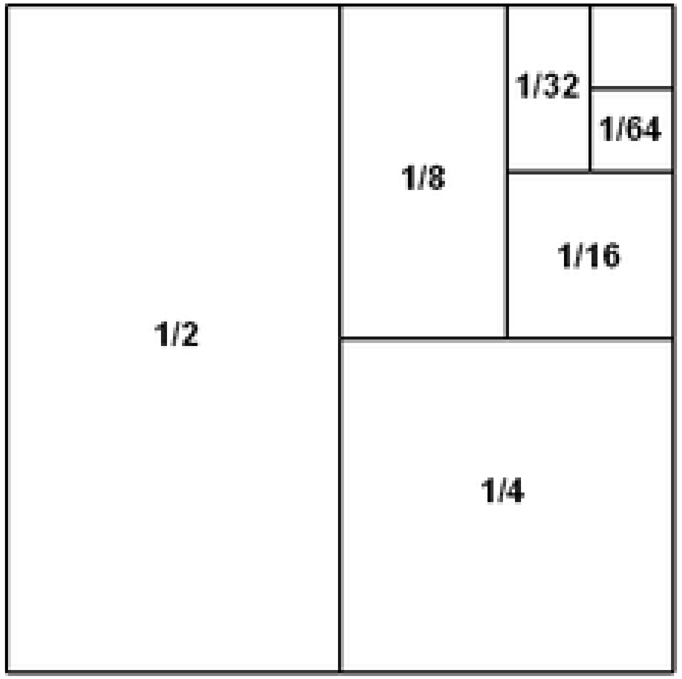
Cost

Cost = Cost of internal nodes + Cost of leaf nodes

$= [\ n^2 + n^2/_2 + n^2/_4 \cdots 1\ ]\ +\ n * 1$

$= n^2\ +\ n^2\ [^1/_2 + ^1/_4 \ldots]\ +\ n$

$n^2 + n^2 + n\ =\ O(n^2)$



Pause & Think

Write a recursive function for finding the power of a number and access its time complexity.

Given x and n,   find x^n

$$x^n = \begin{cases} x * x^{n-1}, & n > 1 \\ x, & n = 1 \end{cases}$$

Function FindPower (x, n)

```
# x is base  # n – exponent
    if (n==1){
            return x;
    }
    else {
            res = FindPower(x, n-1);
            res = x * res;
            return res;
    }
```

Input Size
Exponent n

Basic Operation
Multiplication

$$T(n) = \begin{cases} T(n-1) + 1, & n > 1 \\ 0, & n = 1 \end{cases}$$

$$T(n) = O(n - 1)$$

Can I get a better algorithm than this??

Summary

- Discussed the method of iteration and recursion tree for solving recurrences

Thank You
Happy Learning

Success is always inevitable with Hard Work and Perseverance