$M_1 \times M_2 \times M_3 \times \cdots \times M_n$

$m(i,j)$ = optimal cost of multiplying Matrices $M_i \times M_{i+1} \times \ldots \times M_j$

$m(i,j)$ = BASE: $\boxed{0 \text{ if } i=j}$

$M_i : r_i, c_i$

$M_1 \times M_2 \times M_3 \times M_4$

$M_1 \times (M_2 \times M_3 \times M_4)$ $(M_1 \times M_2) \times (M_3 \times M_4)$

ALGORITHM DESIGN USING DYNAMIC PROGRAMMING METHOD: I

$(M_1 \times M_2 \times M_3) \times M_4$

$\boxed{\text{if } j=i+1, \ \boxed{r_i * c_i * c_j}}$

Recursive:- $[r_i, c_k]$ $[r_{k+1}, c_j]$

$m_{ij} = \{ m_{ik} + m_{k+1,j} + r_i * c_k * c_j \}$ MINIMUM $k=i$ to j

m_{ij}

$M_{ii} \times M_{i+1,j} \ldots M_{i,k} * M_{k+1,j} \ldots$

$M_{i,j-1}, M_{jj}$

Feb 7, 2022

1. Identical Subproblem Recognition
2. Remember (Memoization)
3. Reuse

Partha P Chakrabarti
Indian Institute of Technology Kharagpur

Overview of Algorithm Design

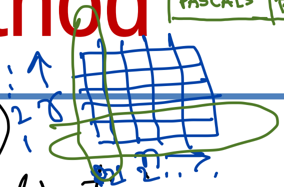
1. Initial Solution
   a. Recursive Definition – A set of Solutions
   b. Inductive Proof of Correctness
   c. Analysis Using Recurrence Relations
2. Exploration of Possibilities
   a. Decomposition or Unfolding of the Recursion Tree
   b. Examination of Structures formed
   c. Re-composition Properties
3. Choice of Solution & Complexity Analysis
   a. Balancing the Split, Choosing Paths
   b. Identical Sub-problems memoization
4. Data Structures & Complexity Analysis
   a. Remembering Past Computation for Future
   b. Space Complexity
5. Final Algorithm & Complexity Analysis
   a. Traversal of the Recursion Tree
   b. Pruning
6. Implementation
   a. Available Memory, Time, Quality of Solution, etc

1. Core Methods
   a. Divide and Conquer
   b. Greedy Algorithms
   c. Dynamic Programming
   d. Branch-and-Bound
   e. Analysis using Recurrences
   f. Advanced Data Structuring
2. Important Problems to be addressed
   a. Sorting and Searching
   b. Strings and Patterns
   c. Trees and Graphs
   d. Combinatorial Optimization
3. Complexity & Advanced Topics
   a. Time and Space Complexity
   b. Lower Bounds
   c. Polynomial Time, NP-Hard
   d. Parallelizability, Randomization

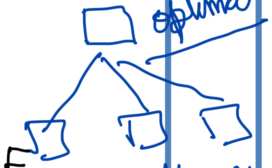
Basics of Dynamic Programming Method

$^{n}C_{r} = {}^{n-1}C_{r-1} + {}^{n-1}C_{r}$

PASCAL's Triangle



1. Recursive Decomposition
   ↳ optimal sub-structure — optimization in nature
2. HANDLING IDENTICAL SUB-PROBLEMS ✓
3. MEMOIZATION ✓ & REUSE ✓
4. Evaluation
   - A) Top-down
   - B) Iterative Bottom up
5. Data Structures



optimal

subproblems are optimally solved

Done[0] → 0, 1, 2

VAL[0] ← when Done = 2

gcd(x, y)

a) Fibonacci (Pingala)

b) Matrix Chain Multiplication

c) String Related
- Longest Common Subsequence
- Sequence Alignment
- NLP related problems

d) Matrix operations

e) Graph Algorithms

f) Coins / Knapsack

g) optimal BST

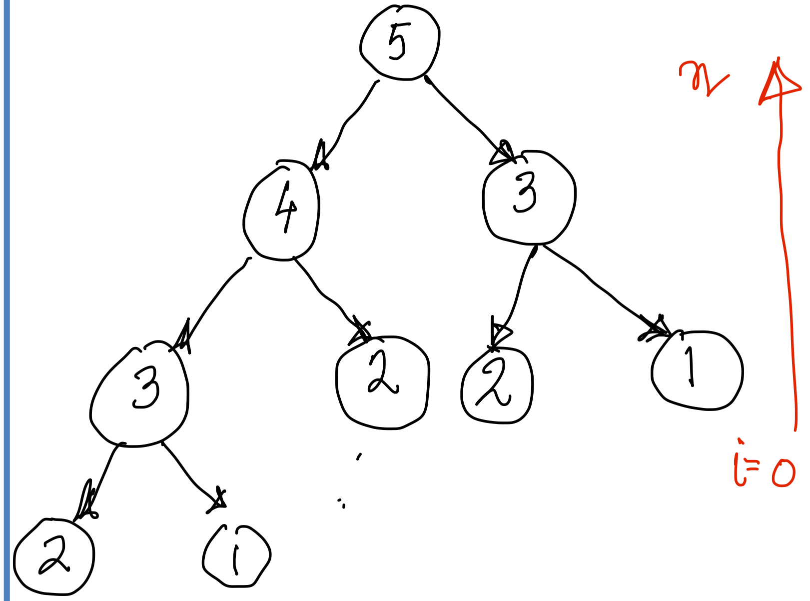
n!, coin, knapsack

Revising Fibonacci-like Structures

$$f(n) = f(n-1) + \underbrace{f(n-2)}_{n-k},\quad n \geq 2$$

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

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



F[ ], Done[ ]

Done[0] = Done[1] = 1   Top-down

All other Done[i] = 0

F[0] = 0   F[1] = 1

```
eval-f(n)
{ if (Done[n] = 1) return (F[n])
    m = eval-f(n-1) + eval-f(n-2)
    Done[n] = 1
    F[n] = m
    return(F[n])
}
```

Bottom up

```
F[0]=0, F[1]=1
for i = 2 to n do
    F[i] = F[i-1] + F[i-2]
```

only 2 add'l variables

Fibonacci-like Structures (cntd.)

$f(n) = r(n)$ if $c(n)$ is true

$= f(g(n)) + f(h(n))$

if $c(n)$ is false

where $c(n)$, $r(n)$, $g(n)$, $h(n)$ are not recursive and can be computed deterministically

F[ ]

Done[i] —
- 0 if it has not been evaluated
- 1 if evaluation has begun but not completed
- 2 if final value is computed

```
eval-f(n)
{ if (Done[n] = 2) return (F[n])
  if (c(n) = true)
  { Done[n] = 2,  F[n] = r(n)
    return (F[n]) }
  if (Done[n] = 1) return ("CYCLE")
  Done[n] = 1
  x = g(n) , y = h(n)
  z = eval-f (x) + eval-f(y)
  F[n] = z
  Done[n] = 2
  return (F[n])
}
```

$M_1 \times M_2 \times \cdots \times M_n$

a+b+c = (a+b)+c, a+(b+c)

$y_1 = x_2$, $y_2 = x_3$

$\boxed{M1 \times M2 \times M3}$

$[x_1, y_1]$ $[x_2, y_2]$ $[x_3, y_3]$

MATRIX CHAIN MULTIPLICATION Problem

MatMul → ASSOCIATIVE but NOT COMMUTATIVE

(M1 X (M2 X (M3 X M4))) = ((M1 x M2) X (M3 X M4)) = (((M1 x M2) X M3) X M4) = (M1 X (M2 X M3)) X M4)

BUT THE NUMBER OF MULTIPLICATIONS TO GET THE ANSWER DIFFER !!

Let A be a [p by q] Matrix and B be a [q by r] Matrix. The number of multiplications needed to compute A X B = p*q*r

M1 X M2 X M3 ⇒ ((M1 X M2) X M3

→ M1 X (M2 X M3)

A = [60, 6]

B = [6, 3]

Thus if M1 be a [10 by 30] Matrix, M2 be a [30 by 5] Matrix and M3 be a [5 by 60] Matrix

AB = A X B [10 X 3]

AB[i, j] = 6

10 X 30 X 6

Then the number of computations for

P X M3

[10 x 5] [5 x 60]

1500 3000

(M1 X M2) X M3 = 10*30*5 for P = (M1 X M2) and 10*5*60 for P X M3. Total = $\boxed{4500}$

18000

M1 X (M2 X M3) = 30*5*60 for Q = (M2 X M3) and 10*30*60 for M1 X Q. Total = $\boxed{27000}$

Q 9000 [10 X 30] [30 X 60]

Matrix Chain Multiplication: Recursive Definition

$M_1 \times M_2 \times M_3 \times \cdots \times M_n$ = $[p_0, p_n]$

$[p_0, p_1]\ [p_1, p_2]\ [p_2, p_3] \quad [p_{n-1}, p_n]$

m_{ij} = optimal multiplications for multiplying $M_i \times M_{i+1} \cdots \times M_j$

① ②

$M_{ik} \times (M_{k+1,j})$

$m_{ij} = 0 \quad \text{if} \quad i = j$

$= \min_{k=i}^{j-1} \{ m_{ik} + m_{k+1,j} + p_{i-1} * p_k * p_j \}$ if $i < j$

$M_1 = 10 \times 30$

$M_2 = 30 \times 5$

$M_3 = 5 \times 60$

$M_4 = 60 \times 4$

$p_0 = 10$

$p_1 = 30$

$p_2 = 5$

$p_3 = 60$

$p_4 = 4$

$M_1 \times M_2 \times M_3 \times M_4$

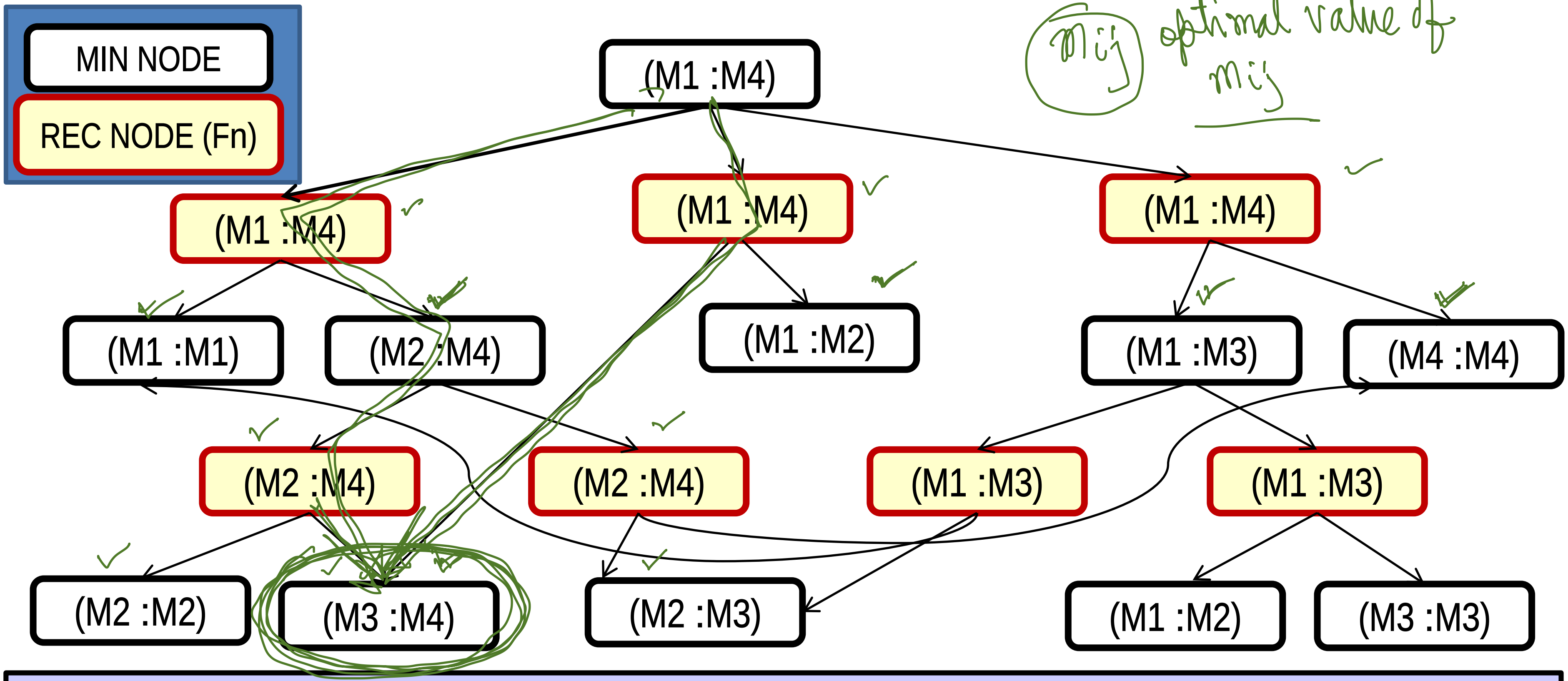
$M_1 \times (M_2 \times M_3 \times M_4)$

$[M_{11} \times M_{24}]$

$M_{12} \times M_{34}$

$M_{13} \times M_{44}$

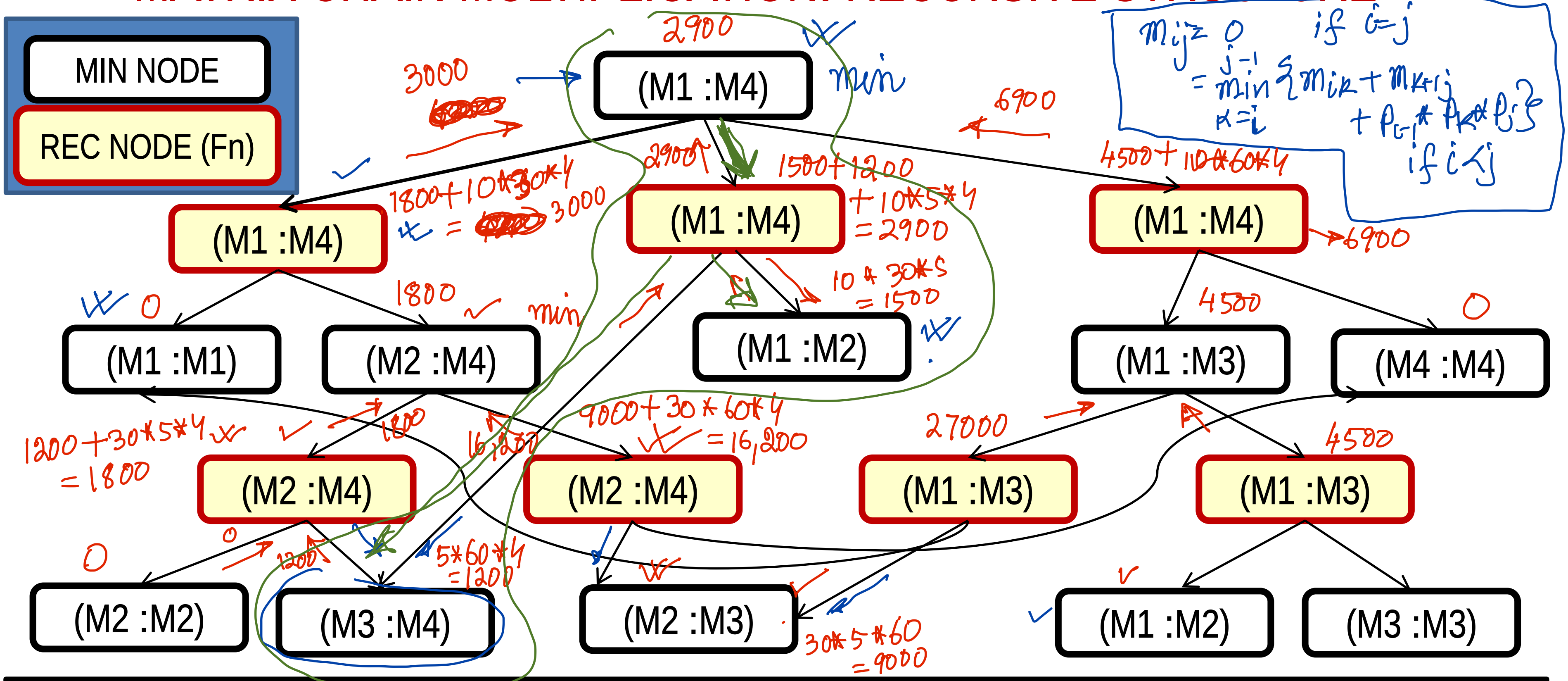
MATRIX CHAIN MULTIPLICATION: RECURSIVE STRUCTURE



(M1 X (M2 X (M3 X M4))) = ((M1 x M2) X (M3 X M4)) = (((M1 x M2) X M3) X M4) = (M1 X (M2 X M3)) X M4)

M1 [10 by 30], M2 [30 by 5], M3 [5 by 60], M4 [60 by 4]

MATRIX CHAIN MULTIPLICATION: RECURSIVE STRUCTURE



$m_{ij} = 0$ if $i=j$

$= \min_{k=i}^{j-1} \{ m_{ik} + m_{k+1,j} + P_{i-1} * P_k * P_j \}$ if $i<j$

(M1 X (M2 X (M3 X M4))) = ((M1 x M2) X (M3 X M4)) = (((M1 x M2) X M3) X M4) = (M1 X (M2 X M3)) X M4)

M1 [10 by 30], M2 [30 by 5], M3 [5 by 60], M4 [60 by 4]

P[0] = 10 P[1] = 30 P2[5] P3=[60] P4[4]

Matrix Chain Multiplication: Top-Down Evaluation

$M[i,j]$, $Done[i,j] = 0$

```
eval-m(i,j)
{ if (Done[i,j] = 1) return (M[i,j])
  if (i=j) { Done[i,j]=1;
             M[i,j]=0;
             return (M[i,j])}
  val = ∞
  for (k = i to j-1)
  { v_k = eval-m(i,k) + eval-m(k+1,j) + p[i-1]*p[k]*p[j]
    if (v_k < val) val = v_k
  }
  Done[i,j]=1
  M[i,j]=val
  return (M[i,j])
}
```

REUSE

BASE

$O(n)$

MINIMUM

Recursive

Decomposition

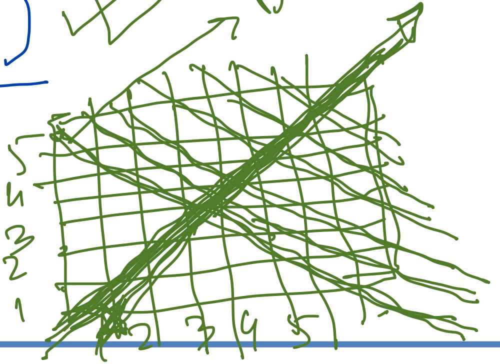
$n^2 \rightarrow M[i,j]$

n

$O(n^3)$

BASE

$M[i,j]$



Matrix Chain Multiplication: Iterative Evaluation

	1	2	3	4
1	0	*	*	*
2		0		
3			0	
4				0

0	1	2	3	4
10	30	5	60	4

	1	2	3	4
1	0	1500	4500	2900
2		0	9000	1800
3			0	1200
4				0

$O(n^2 * n)$

$O(n^3)$

iterative eval()

```
{for (i = 1 to n) M[i,i] = 0
   for (diff = 1 to n-1)
     for (i = 1 to n-diff)
       { j = i + diff
         M[i,j] = ∞
         for (k = i to j-1)
  O(n)   { q = M[i,k] + M[k+1,j]
                + p[i-1]*p[k]*p[j]
         } if (q < M[i,j]) M[i,j] = q
       }
}
```

Summary

1. Recursive Sub-structure
2. Memoization & Reuse
3. Top-down (Recursive) & Iterative

Algorithms

Reuse

Base

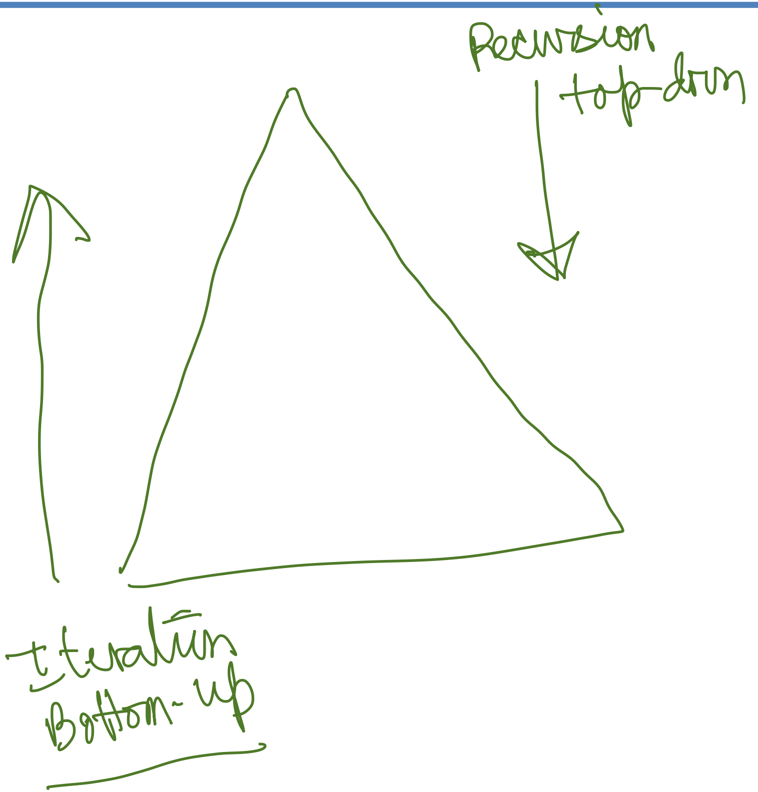
Decomp

Recursive

Recomp

Remember

Top-down



Thank you

Any Questions?