ALGORITHM DESIGN USING DYNAMIC PROGRAMMING METHOD: I

Feb 8, 2021

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
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

Divide & Conquer

Greedy

memoization

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

1. Recursive Decomposition optimization
   ↳ optimal sub-structure
2. HANDLING IDENTICAL SUB-PROBLEMS } ⟸ exponential time complexity
3. MEMOIZATION & REUSE
   Remember → Data Storage
4. Evaluation
   A) Top-down done[ ]
   B) Iterative Bottom up ⟸
5. Data Structures
   ↳ preprocessing
   Acyclic structure

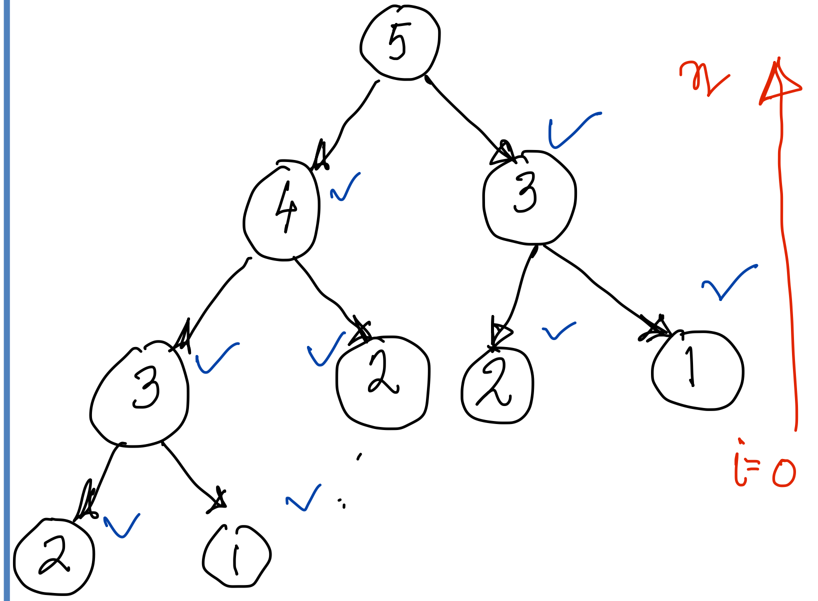
a) Fibonacci (Pingala)
b) Matrix Chain Multiplication
c) String Related
   - Longest Common Subsequence
   - Sequence Alignment
   - NLP related problems
d) Matrix operations
e) Graph Algorithms
f) Coins / Knapsack
g) optimal BST

Revising Fibonacci-like Structures

$$f(n) = f(n-1) + f(n-2), \quad n \geq 2$$
$$= 0, \quad n = 0$$
$$= 1, \quad n = 1$$

$n-k$

constant k



F[ ], Done[ ]

Done[0] = Done[1] = 1
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])
}
```

```
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.)

Ackermann's Function

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

$= \boxed{f(g(n)) + f(h(n))}$ 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

check for cycles

Top-down

1

```
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])
}
```

2 Base

3 cycle detection

4 Recursive

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

MATRIX CHAIN MULTIPLICATION Problem

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

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

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

associative

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

$C(1) = \cancel{0}\ 1$

$C(2) = 1$

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

$[P_0, P_1]\ [P_1, P_2]\ [P_2\ P_3]\ [P_3\ P_4]$

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

$C(n) = \sum_{i=1}^{n-1} C(i) * C(n-i)$ 5 ways $= C_1 * C_1$

$\cancel{\frac{n(n-1)}{2}} - 1$

Then the number of computations for

P

(M1 X M2) X M3 = 10*30*5 for P = (M1 X M2) and 10*5*60 for P X M3. Total = 4500

M1 X (M2 X M3) = 30*5*60 for Q = (M2 X M3) and 10*30*60 for M1 X Q. Total = 27000

$C(n) = \sum C(i) * \cancel{C(n-i)}$

Matrix Chain Multiplication: Recursive Definition

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

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

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

① ②

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

$$m_{ij} = \begin{cases} 0, & \text{if } i=j \\ \min\limits_{k=i}^{j-1} \{ \boxed{m_{ik}} + \boxed{m_{k+1,j}} + p_{i-1} * p_k * p_j \} & \text{if } i<j \end{cases}$$

$\boxed{M_i \text{ to } M_j}$

$m_{ij} \rightarrow$ No. of computation to multiply $M_i \ldots M_j$

$[p_0\ p_1]\ [p_1\ p_2]\ [p_2\ p_3]$

$M_1 \times M_2 \times M_3$

$[p_1\ p_2]\ [p_2\ p_3]\ [p_3\ p_4]$

$M_2 \to 0$

$M_1 = 10 \times 30$

$M_2 = 30 \times 5$

$M_3 = 5 \times 60$

$M_4 = 60 \times 4$

$p_0 = 10$

$p_1 = 30$

$p_2 = 5$

$p_3 = 60$

$p_4 = 4$

$M_3 \times M_4$

$[p_2, p_3]\ [p_3, p_4]$

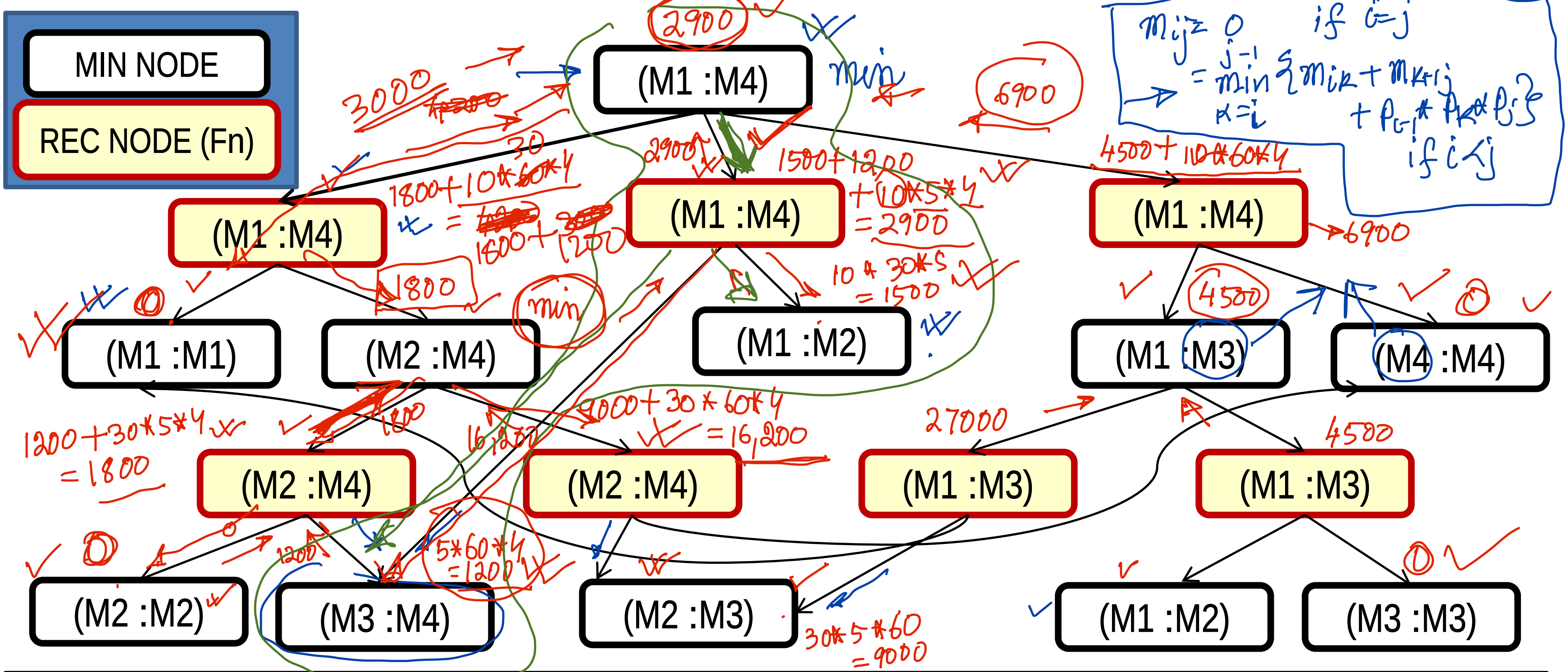
$p_2 * p_3 * p_4$

$\underbrace{(M_i \ldots M_k)}_{m_{ik}} \times \underbrace{(M_{k+1} \ldots M_j)}_{m_{k+1,j}}$

$[p_{i-1}\ p_k]\ [p_k\ p_j]$

$p_{i-1} * p_k * p_j$

MATRIX CHAIN MULTIPLICATION: RECURSIVE STRUCTURE



$$m_{ij} = 0 \quad \text{if } i=j$$
$$= \min_{k=i}^{j-1}\{m_{ik} + m_{k+1,j} + p_{i-1} * p_k * p_j\} \quad \text{if } i<j$$

(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)

REUSE

{ if ($Done[i,j] = 1$) return ($M[i,j]$)

BASE

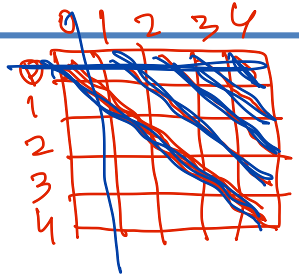
if ($i = j$) { $Done[i,j] = 1$;
$M[i,j] = 0$;
return ($M[i,j]$) }

RECURSIVE

val $= \infty$
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 $= v_k$
}

$(1, n)$

$Done[i,j] = 1$
$M[i,j] = val$
return ($M[i,j]$)
}



$n^2 \rightarrow M[i,j]$
n
$O(n^3)$

I can decipher my acyclic dependency structure

$O(n^2 * n)$
$= O(n^3)$

Bottom-up iterative algorithm

(i, i)
$(i, i+1)$
$(i, i+2)$ $(i, i+3)$ .... (i, n)

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

$i, i+1$

$i, i+2$

$i, i+diff$

$O(n^2 * n)$

$O(n^3)$

$(1,2), (2,3) \ldots$

$(1,3)\ (2,4) \ldots$

```
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)
      { q = M[i,k] + M[k+1,j]
            + p[i-1] * p[k] * p[j]
        if (q < M[i,j]) M[i,j] = q
      }
    }
}
```

$O(n \cdot n \cdot n) = O(n^3)$

Summary

$M_{7,9}$ $(7, 9)$

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

M

~~$M_{x_k} M_{y_k}$~~

$M_{x_k y_k}$

$L = \{(i, i)\}$

~~From L~~

Let $L = \{(x_1, y_1) (x_2, y_2) \ldots\ldots (x_k, y_k) \ldots\ldots\}$

~~select some (p, q) from L~~

select some $\boxed{(p_1, q_1)\ (p_2, q_2)}$ from L which can be merged (p_3, q_3)

$\boxed{\begin{array}{l} p_1 < q_1 \\ p_2 < q_2 \end{array}}$ $\boxed{q_1 = p_2 - 1}$ $\boxed{q_3 = q_2}$

$\boxed{p_3 = p_1}$

Thank you

Any Questions?