



Unit III

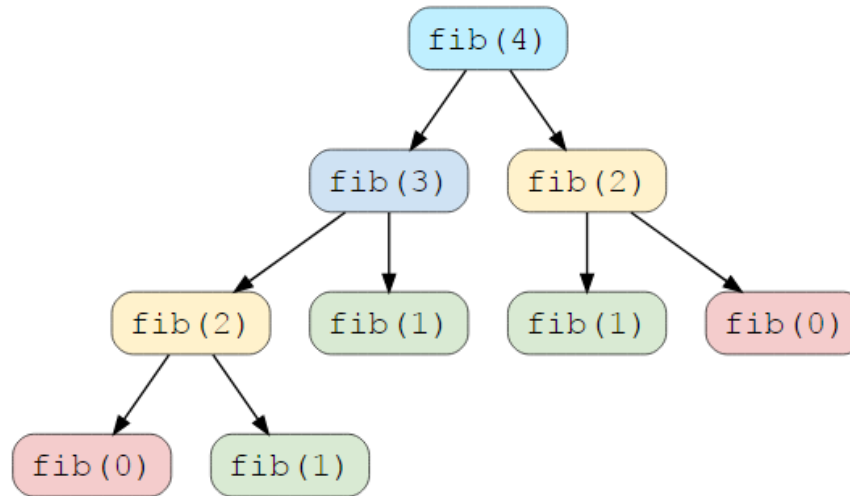


Dynamic Programming and Greedy Technique

- Dynamic Programming
 - ***Computing a Binomial Coefficient***
 - Warshall's algorithm
 - Floyd's algorithm
 - Optimal Binary Search Trees
 - Knapsack Problem and Memory functions

Dynamic Programming

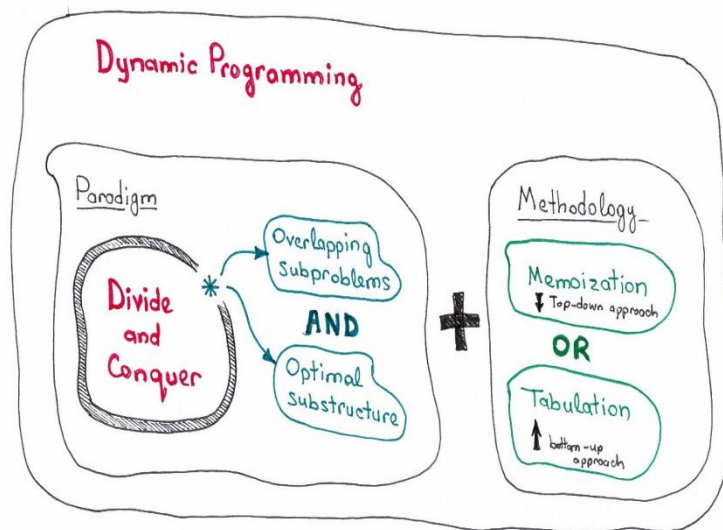
- Dynamic programming – pblm $\square$ similar sub problems $\square$ reuse the solution
- **Characteristics**
 - Overlapping sub problems – solving same sub problems
 - Optimal substructure property – optimal solution can be built from sub problem
 - Example : Fibonacci series



Dynamic Programming

- **Methodology**

- Top-down with memoization
 - Storing the result of already solved sub-problem is called memoization
- Bottom-up with Tabulation
 - Sub-problems (bottom – up)



Difference between Divide and conquer and Dynamic Programming

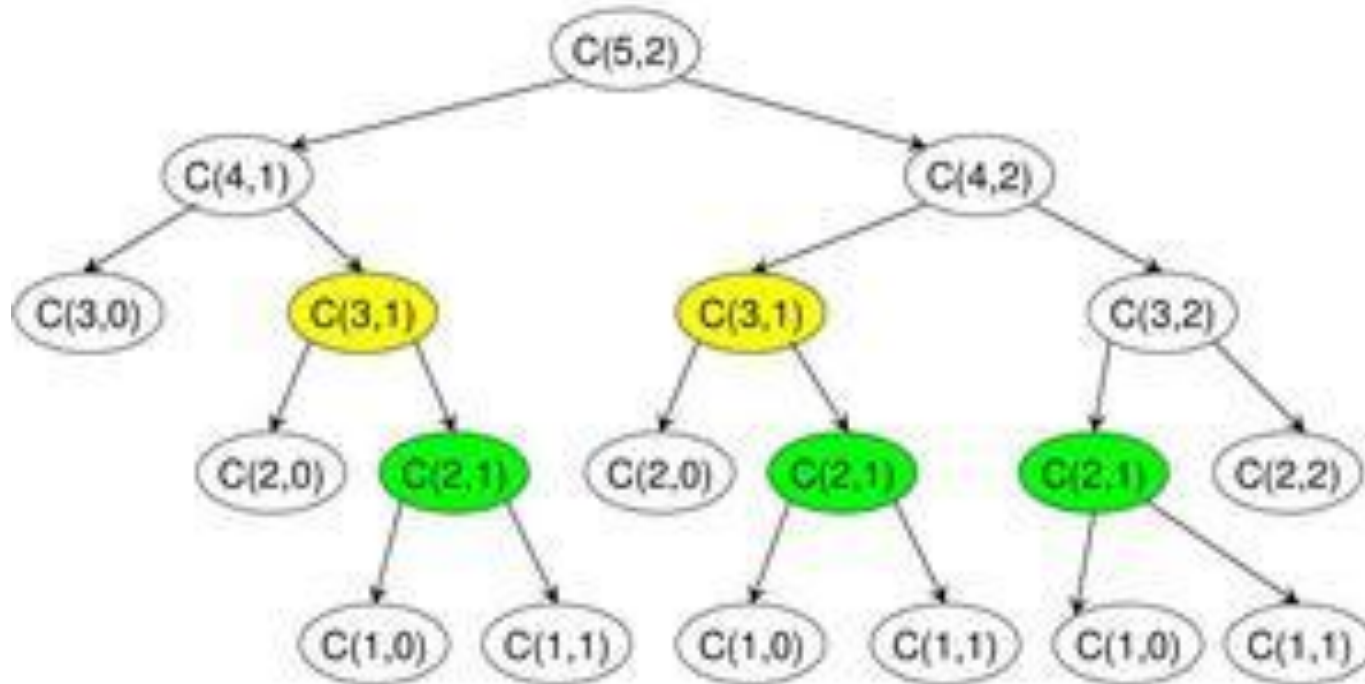
Divide and conquer	Dynamic Programming
Sub problems are not dependent on each other	Sub problems are dependent on each other
Doesn't store the solution of sub-problem	Stores the solution of sub problem

Computing a Binomial Coefficient

- Binomial coefficient – computation of no of ways r items that can be chosen from n elements $C(n, r)$
- $C(n, k) = n! / (n-k)! * k!$
- $C(n, k) = C(n-1, k-1) + C(n-1, k)$, $n > k$, $k > 0$
- $C(n, 0) = 1$, $C(n, n) = 1$
- Example:
- 1st formula : $C(4, 2) = 4! / (2! * 2!) = 24 / 4 = 6$
- 2nd formula : $C(4, 2) = C(3, 1) + C(3, 2) = \dots = 6$
- $C(4, 2)$ = how many two combinations of elements can be picked from set of 4 elements
- Example: possibilities of 1,2,3,4 = (1,2) (1,3) (1,4) (2,3) (2,4) (3,4)

Computing a Binomial Coefficient

- Example : $C(5,2)$
- $C(n, k) = C(n-1, k-1) + C(n-1, k)$, $n > k$, $k > 0$
- $C(n, 0) = 1$, $C(n, n) = 1$



Computing a Binomial Coefficient - Tabulation

	0	1	2	3	4	5	...	$(k-1)$	k
0	1								
1	1	1							
2	1	2	1						
3	1	3	3	1					
4	1	4	6	4	1				
5									
$\vdots$									
k	1								1
$\vdots$									
$(n-1)$	1							$C(n-1, k-1)$	$C(n-1, k)$
n	1								$C(n, k)$

Computing a Binomial Coefficient - Algorithm

Algorithm *Binomial*(n, k)

for $i \leftarrow 0$ **to** n **do** // fill out the table row wise

for $i = 0$ **to** $\min(i, k)$ **do**

if $j == 0$ or $j == i$ **then** $C[i, j] \leftarrow 1$ // IC

else $C[i, j] \leftarrow C[i-1, j-1] + C[i-1, j]$ // recursive
relation

return $C[n, k]$

Computing a Binomial Coefficient - Analysis

- Cost of the algorithm – table
- Sum – 2 parts (upper and lower triangle)
- $A(n, k) = \text{sum for upper triangle} + \text{sum for the lower rectangle}$

$$\begin{aligned}
 A(n, k) &= \sum_{i=1}^k \sum_{j=1}^{i-1} 1 + \sum_{i=k+1}^n \sum_{j=1}^k 1 \\
 &\Rightarrow \sum_{i=1}^k ((i-1) - \cancel{x} + \cancel{x}) + \sum_{i=k+1}^n (k - \cancel{x} + \cancel{x}) \\
 &\Rightarrow \sum_{i=1}^k (i-1) + \sum_{i=k+1}^n k \\
 &\Rightarrow \left[\sum_{i=1}^k i - \sum_{i=1}^k 1 \right] + k \sum_{i=k+1}^n 1 \\
 &\Rightarrow \frac{k(k+1)}{2} - (k - \cancel{x} + \cancel{x}) + k [n - (k+1) + 1]
 \end{aligned}$$

Computing a Binomial Coefficient - Analysis

$$\begin{aligned} &\Rightarrow \frac{k^2 + k}{2} - k + k[n - k - 1 + 1] \\ &\Rightarrow \frac{k^2 + k - 2k + 2(nk - k^2)}{2} \\ &\Rightarrow \frac{k^2 - k + 2nk - 2k^2}{2} \\ &\Rightarrow \frac{-k^2 - k + 2nk}{2} \\ &\approx nk \\ &\boxed{O(nk)} \end{aligned}$$

20210223_143517.jpg (3120 x 4160)