DYNAMIC PROGRAMMING

Dynamic programming is a problem solving technique like Divide and conquer, solves problems by dividing them into sub problems. Dynamic programming is used when the sub-problems are not independent.

Dynamic programming solves each sub problem once and stores the result in a table so that it can rapidly retrieved if needed again again. It is often used in optimization problems: A problem with many possible solutions for which we want to find an optimal (the best) solution. (There may be more than 1 optimal solution). Dynamic programming is equally applicable for decision problems where sequential property is induced solely for computational convenience.

Dynamic programming can be thought of as being the reverse of recursion. Recursion is a top-down mechanism – we take a problem split it up and solve the smaller problems that are created. Dynamic programming is a bottom-up mechanism – we solve all possible small problems and then combine them to obtain solutions for bigger problems

Applications:

⇒ Knapsack problem

⇒ Mathematical optimization problems

⇒ Shortest paths problems

⇒ Matrix chain multiplication

⇒ longest common sequence (LCS)

⇒ Control (Cruise Control, Robotics, Thermostates)

⇒ Flight Control (Balance factors that oppose one another eg maximize accuracy, minimize time)

⇒ Time Sharing: Schedule user and Jobs to maximize CPU usage

⇒ other types of Scheduling..

DIFFERENCE BETWEEN DIVIDE-AND CONQUER $ DYNAMIC PROGRAMMING

DIVIDE AND CONQUER ALGORITHM	DYNAMIC PROGRAMMING
1. Divide-and Conquer algorithms splits a problem into separate sub problems, Solve the sub problems and Combine the results for a solution to the original problem. eg Quick sort, merge sort, Binary search. e.t.c	Dynamic programming splits a problem into sub problems, Some of which are common, Solves the sub problems, and Combines the results for a solution to the original problem. eg. Matrix chain Multiplica—, longest common sub sequence e.t

2. Divide and Conquer algorithm can be thought of as top-down algorithm	Dynemic programming can be thought of as bottom-up
3. In divide and Conquer, Sub-problems are independent.	In Dynemic programming, Sub-problems are not independent
4. Divide & Conquer solutions are simple as compared to Dynemic programming	Dynemic programming solutions are quite complex and tricky.
5. Only one ~~Sequence~~ Decision sequence is ever generated	Many decision sequence may be generated
6. Divide and Conquer can be used for any kind of problems.	Dynemic programming is generally used for optimization problems.

BACKTRACKING.

Have you ever seen blind People walking in the roads? If they find any obstacles in their way, they would just move backward then they will proceed in other direction ~~with~~ with intelligence! Similarly, if an algorithm backtracks with intelligence, it is called

Backtracking algorithm. Backtracking is a methodological way of trying out various sequences of decisions until we find one that "works". eg. N-Queens problem.

N-QUEENS PROBLEM

N-queen problem is to place n-queens in such a manner on an nxn chessboard that no two queens attack each other by being in the same row, column or diagonal.

It can be seen that for n=1, the problem has a trivial solution, and no solution exist for n=2 and n=3, so first we will consider the 4-queens problems and then generalise it to n-queens problem.

Given, a 4x4 chessboard and number the rows and column of the chessbard 1 through 4. Since we have to place 4 queens such as q1, q2, q3, and q4 on a chessboard, such that no two queens attack each other. In such condition, each queen must be placed on a diffrent row and diffrent column.

	1	2	3	4
1			q1	
2	q2			
3				q3
4		q4		

<3, 1, 4, 2>

	1	2	3	4
1		q1		
2				q2
3	q3			
4			q4	

<2, 4, 1, 3>

LONGEST COMMON SUBSEQUENCE

A subsequence of a given sequence is just the given sequence with some elements left out. Given two sequence X and Y, We say that a sequence Z is a common sequence of X and Y if Z is a subsequence of both X and Y. LCS problem can be solved using Dynamic Programming. eg.

String 1: a b c d e f g h i j

String 2: c d g i

LCS = cdgi

X = a b c d e F g h i j

Y = e c d g 1

~~[illegible]~~ = eg1 ane cdgi

Lcs = cdgi

LCS USING DYNAMIC PROGRAMMING

Algorithm:

```
IF (A[i] = B[j]
    Lcs[i, j] = 1 + Lcs[i-1, j-1]
else
    Lcs[i, j] = [Max(Lcs(i-1, j)], Lcs[i, j-1]
```

EXAMPLES

A = b, d

B = a b c d

		a	b	c	d
	0	1	2	3	4
0	0	0	0	0	0
b 1	0	0	↖ 1	← 1	1
d 2	0	0	1	1	↖ 2

b d

Lcs = bd.

Ex 2: Find the longest common subsequence of the following strings:

String 1: Stone

String 2: longest

			L	o	n	g	e	s	t
		0	1	2	3	4	5	6	7
	0	0	0	0	0	0	0	0	0
S	1	0	0	0	0	0	0	1	1
t	2	0	0	0	0	0	0	1	2
o	3	0	0	↖1	1	1	1	1	2
n	4	0	0	1	↖2	←2	2	2	2
e	5	0	0	1	2	2	↖3	←3	←3
				o	n		e		

LCS = one

Ex 3; Determine the LCS of (1, 0, 0, 1, 0, 1, 0, 1) and (0, 1, 0, 1, 1, 0, 1, 1, 0)