Page No.: Date :

Shri Krishna Mahavidyalaya, Gunjoti

Department of Mathematics.

M.Sc Math :- Sem II nd (2021-22)

Project papar :- operation Reasearch -II (MAT 536)

project subect :- study on dynamic programming

Student Name :- sontakke suvarna Ashok

PRN NO :- 2014015200199211

Seat No :- CTD401580

16/20

Head
Department of Mathematics,
Shrikrishna Mahavidyalaya,
Tq.Omerga Dist.Osmanabad
(M.S.)

INDEX

(5)

Dynamic Programming

★ INTRODUCTION :-

Dynamic programming is a useful. mathematical techingue for making a sequence of interrelated decisions. It provided a systematic procedure for determinig the optimal combination of decisions.

In Constrast to Linear programming there does not exist a standard mathematical formulation of the dynamic programming problem. Rather, dynamic programming is a genral type of approch to problem solving, and the particular equation and used must be developed to fit each situation Therefore, a certain degree of ingenuity and insight into the general structure of dynamic programming procedure. These abilities can best be developed by an expousure to a wide variety of dynammic programming applications and a study of the charactereistics that are variety to all these situations. A large number of illustrative examples are presented for this purpose.

Dynammic programming is used in various process includding sequence comparison gene recognition and many other problems.

Theory of Dyanamic Programming

Dynamic programming method for solving a complex problem by breaking down the given problem into a number of sub problems and solving these sub problems once & storing the solⁿ to theses sub problems in a table.

Subsequence :-

If we define a sequence p as $p = \langle p_1, p_2, \ldots, p_m \rangle$, then we can define a subsequence R as $R = \langle r_1, r_2, \ldots r_K \rangle$ for the given sequence p This is true if and only if R is derived from p. That is elements of the subsequence R is chosen from the sequence p in a strictly increasing order.

Common Subsequence :-

Let us consider two sequences p & Q where p is defined as $p = p_1, p_2, \ldots, p_m \rangle$ & Q as $Q = \langle q_1, q_2, \ldots, q_n \rangle$. Then the common subsequence of p and textit Q is the defined as $R = \langle r_1, r_2, \ldots, r_K \rangle$, where R is a subsequce of p and textit Q is defined as $R = \langle r_1, r_2, \ldots, r_K \rangle$, where R is a subsequece both p and Q.

for example, if p = < AGCGTAG > & Q = < GTCAGA >, Then by inspection we get the length of the common subsequence as 4 & one of the common subsequence of GTAC.

Four step involved in devloping a dynamic programming algorithm :-

- (i) Identify the structure of the optimal solution in a given problem.
- (ii) use recursion to define the value of an optimal solution
- (iii) Find the optimal value using bottom up approach
- (iv) construct an optimal solution for the given problem using the above information

Elements of dynamic Programming :-

In order to apply dynamic programming algorithm in a problem, the problem must satisfy two Properties. They are optimal substructure and overlapping subproblems. Subsequent sections discuss about these two properties.

optimal substructure :-

A problem is said to have an optimal substructure if the problem

has an optimal solution which is contained in the optimal solution of the subproblemes. i.e. an optimal solution so that problem is obtained by combining the optimal solution of the subproblems. while solving the problem using dynamic programming we must ensure that the range of subproblems we consider includes those that can be used in an optimal solution.
This property can be understood by the given example from graph theory The shortest path p from a vertex a to a vertex c in a given graph exhibits substructure. Take any intermediate vertex b on this shortest path p. If the initial choice that p is the shortest path between the between the vertex a and c is true, then there excists intermediate shortest paths betn ab (shortest path betn a and b) and bc (shortest path betn b and c).

Overlapping Subproblems :-

An optimization problem have overlapping subproblem if the recursive algorithm repeatedly solves the same subproblem only once and then store the soln of the same in a table which can be used later. This can be understood by the computing fibonacci sequence where one compute fibonacci number recursively The problem of computing nth fibonacci number Fn includes computing Fn-1 &

Fn-2 and adding the two. In computing Fn-2 for Fn-1 and by storing the same we avoid the repetition of computing it.

storing the (basis) values of the optimal solution :-

In general there are two ways by which we can store the sol^n to the subproblems of a given problem :

- (1) Bottom up approach
- (2) memoization.

Bottom up apporch :-

In dynamic programmig, we have a table to store the computed values of the optimal sol^n to the subproblems, once we have formulated the solution to a given problem recursively using the $soln$ to the subproblem, using this we solve the subproblems first and combine the solution to the subproblem to arrive at the solution to a given problem. This is usually done in a tubular from by using the computed sol^n of the subproblems from the teble.

Longest Common Subsequence [LCS] :-

Logest common subsequence problem is the problem of finding the longest common subsequence in given two sequences in same relative order. Bioformatics is one of the important area where comparison of two sequences is used. comparing DNA of two organism to understand the similarity betn two organism is often required in the field of bioinformatics.

As mentioned before the brute force way of finding the longest common sequence will take exponential time. If we have a sequence p with m element then it will have 2^m subsequence. comparing each subsequence of p with another subsequence Q will result in exponential time This is practically impossible for longer sequences. In the section well shall see that LCS problem satisfies optimal substructure property and overlapping subproblems.

Step I : Identifiying the structure of the LCS problem :-

Theorem I : optimal substructure of an LCS :-

Let $P_m = \langle p_1, p_2, \cdots, p_m \rangle$ $Q_n = \langle q_1, q_2, \cdots, q_n \rangle$

two sequences and the LCS of P & Q is given by R where $R = \langle r_1, r_2, \dots, r_k \rangle$.

- 1) If $p_m = q_n$ Then $r_k = p_m = q_n$ & R_{k-1} is an LCS of P_{m-1} and Q_{n-1}
- 2) If $p_m \neq q_n$ Then $r_k \neq p_m$ implies that R is an LCS of P_{m-1} and Q
- 3) If $p_m \neq q_n$ Then $r_k \neq q_n$ implies that R is an LCS of p_m and Q_{n-1}

Proof :-

(1) Assume to the contrary, that is if $r_k \neq p_m$ then by adding $p_m = q_n$ to the sequence R we obtain common subsequence of P and Q. This common subsequence will be of length k+1, which is a contradiction since maximum length of the common subsequence is k. Thus $r_k = p_m = q_n$ consider the common subsequence (R_k) of p_{m-1} and Q_{n-1} with length k-1 Now we have to show that this is an LCS. In order to show that this this is an LCS, assume to the contrary that there exists a common subsequence S of p_{m-1} and Q_{n-1} with length greater than k-1. But we know that if we add $p_m = q_n$ to S, it will give a common subseqence of length greater than k. This is a contradiction.

∴ R_{k-1} is an LCS of p_{m-1} and Q_{n-1}

Step II : Defining a recursion for the optimal Solution :-

Given two sequences P_i and Q_j where $P_i = \langle p_1, q_2, \ldots, p_i \rangle$ and $Q_j = \langle q_1, q_2, \ldots, q_j \rangle$. Now, let us define LCS [i,j] to be the length of an LCS of the sequence P and Q. The optimal structure of the LCS problem is given by the following recursion.

$$LCS[i,j] = \begin{cases} 0 & \text{if } i=0 \text{ \& } j=0 \\ LCS[i-1, j-1]+1 & \text{if } i,j>0 \text{ \& } p_i = p_j \\ \max\{LCS[i,j-1], LCS[i-1,j] & \text{if } i,j>0 \text{ \& } p_i \neq p_j \end{cases} \quad -(1)$$

Theorem 1 and the above recursion (1) can we well understood by the below example.

Lets look at the LCS three of two sequences P and Q where P = < AGCGTAG > AND Q = < GTCAGA >.

LCS [AGCGTA, GTCAGA] = max (LCS [AGCGTA, GTCAGA], LCS [AGCGTAG, GTCAG]]

(a) LCS [AGCGTA, GTCAGA) = max (LCS[AGCGT, GTCAG] + A (theorm I

(b) LCS[AGCGTAG, GTCAG] = max (LCS [AGCGTA, GTCA] + G (theorm i)

Then again applying theorem 1 on (a) & (b) we get

LCS [AGCGT, GTCAG] = max (LCS [AGCG, GTCAG], LCS [AGCGT, GTCA])

LCS [AGCGTA, GTCA] = LCS [AGCGT, GTC] + A.

and so on.

Thus by applying theorem(1) to be the above problem further we can see that the LCS problem satisfies optimal substructure property. Since the LCS problem satisfies both optimal substructure and overlapping subproblems property we can apply dynamic programming to LCS.

Step III :-

Finding an optimal value of LCS.

The pictorial representation of the above defined recursion (1) for sequence P' & Q' with P' along the top row and Q along the leftmost column is shown below

Table I : LCS Table

	0	A	G	C	G	T
0	0	0	0	0	0	0
T	0	0	0	0	0	1
G	0	0	0	1	1	1
C	0	0	1	2	2	2
A	0	1	1	2	2	2
T	0	1	1	2	2	3

We start by filling zeros in the first row and first column of the table 1. Next we will see if the second element on the left side of the table is same as the second element of the sequence given above the table. If the optimal (element) are different then we table the value entered in the corrosponding cell will be the maximum of the values in the cells right above and left to the respective cell If the element matches then the value entered in the corresponding cell will be the value of the diagonal value plus one. We continue the same process till we reach the right bottom corner of the table. the optimal value of LCS will be present at the bottom right corner of the table.

optimal value of the LCS for the subsequences p' and q' is 3.

LCS Table showing up bottom-up-approach

	O	A	G	C	G	T
O	O	O	O	O	O	O
T	O	O	O	O	O	1
G	O	O	1	1	1	1
C	O	O	1	2	2	2
A	O	1	1	2	2	2
T	O	1	1	2	2	3

one of the LCS for given sequnces would be GCT (table) 2. To simpler terms, the element corresponding to the cell where the tail of the arrow begins is a member of LCS in inverse order.

Additional Exploration.

Memoization / Top down approach.

This is an alternate method for bottom up apporach for solving a problem using dynamic programming Memoization can be used if a problem can be solved recursively using the values in a table but in a top down the subproblems are over lapping. This method also requires storing the values in a table but in a top down approach Unlike tabulation, memoization does not fill up all cells unless it it definitely required. Each call is filled based on demand ie, Memoization does not ask for filling up of all cells of a table to reach the final answer.

Result :→

Dynamic programming is very useful techinque for making a sequence of interelated decision. It requires formulating an appropriate recursive relationship for each individeal problem.

References :-

1) Thomas H Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein. Introduction to algorithms, third edition. pp 360-395

2) Longest common subsequence - Introduction LCS length, Techie delight, Coding made easy

3) Dynamic programming and Longest common subsequence, Greeks for greeks, computer science portal for greek.