Dynamic Programming

Raveesh Gupta

March 10, 2019

Chapter 1

Longest Increasing Sub-sequence

The Longest Increasing Sub-sequence (LIS) problem is to find the length of the longest sub-sequence of a given sequence such that all elements of the sub-sequence are sorted in increasing order.

1.1 Numerical Analysis

This is a classical problem in *dynamic programming* to find the longest increasing sub-sequence in a sequence of integers $arr_{1...n}$. First step is to define dp_i.

dp_i := length of longest possible increasing sub-sequence in $arr_{1...n}$

Second step is transition . Definition of a *recurrence relation* is defined using dynamic technology.

$$dp_i = MAX_{arr_j > arr_i}(1 + dp_j), \forall j \epsilon [i+1, n]$$

Final step is to define answer hence Ans $Definition.$

$$MAX_{\forall i \epsilon [1,n]}(dp_i)$$

1.2 Non-Numerical Analysis

Define $forw_i$ that will store j ie index of the the next element in the sub-sequence $arr_{1...n}$, then assume k is the index s.t dp_k is the solution for the given problem. Longest Increasing Sub-sequence is,

$$arr[k], forw[arr[k]], forw[forw[arr[k]]]]...tothelengthofdp[k]$$

Chapter 2

Nails on the board

Consider a marked wooden board, at different points n nails are hammered such that ith nail is on position x_i. Given limitless string, you are required to calculate the minimum total length of string such that.

$$\forall i \epsilon [1, n]$$

is connected to some j.

2.1 Observations

1. It is optimal to connect neighbouring nails to avoid overlapping. $i.e$ j will always equal to $i + 1$.

2. When $n > 4$ sometimes it is optimal to use 2 strings instead of 3 strings to satisfy the condition.

2.2 Analysis

First step is to define a DP definition.

dp_i := Minimum Possible Length of string to connect first i nails.

Second step is to define a transitive function. DP Recursive Definition case 1.

$$dp_i = MIN_{i \epsilon [4,n]}(dp_{i-1} + x_i - x_{i-1}, dp_{i-2} + x_i - x_{i-1}),$$

case 2.

$$dp_3 = x_3 - x_1, i = 3$$

case 3.

$$dp_2 = x_2 - x_1, i = 2$$

Final step is answer definition: dp_n.

Chapter 3

Avoid-two-neighbouring-ones problem

Given positive integer n; how many sequences of n zeroes and ones such that no any two ones occur in neighbouring positions?

3.1 Observations

Let $S_{1..n}$ be a good sequence then if S_n is 0 then S_{n-1} can be 1 or 0 thus reducing our problem to solving $S_{1..n-1}$, otherwise if S_n is 1 then S_{n-1} can only be 0 hence reducing our problem to solving $S_{1...n-2}$.

Trivial cases $n = 1$ or $n = 2$ can be seen easily $i.e$ $dp_1 = 2$ and $dp_2 = 3$.

3.2 Analysis

First step is to define a DP definition.

Let dp_i := Number of good sequences with length i.

Second step is to define a transitive function. DP Recursive Definition.

$$dp_i = dp_{i-1} + dp_{i-2}, \forall i >= 3$$

Final step is answer definition: dp_n.

Chapter 4

Largest-Common-Sub-sequence LCS

Its a classical problem in which given two sequences

$$A = (A_1, A_2, ..., A_n)$$

and

$$B = (B_1, B_2, ..., B_m)$$

, we are to find such two sequences

$$1 <= i_1 < i_2 < ... < i_k <= n$$

,

$$1 <= j_i < j_2 < ... < j_k <= m$$

for maximal possible k, such that

$$A_{i_1} = B_{j_1}, A_{i_2} = B_{j_2}, ..., A_{i_k} = B_{j_k}.$$

4.1 Observations

1. Its obvious that $dp_{i,j} = 0$ if $i = 0$ or $j = 0$.

2.If $A_i = B_j$ then there is only 1 solution $dp_{i-1,j-1} + 1$ and A_i becomes part of our ans, otherwise if $A_i != B_j$ then there are 2 cases $dp_{i-1,j}$ or $dp_{i,j-1}$ (these two statements can be true at the same time, but at least one must be true definitely).

4.2 Analysis

First step is to define a DP definition.

Let $dp_{i,j}$:= Length of the largest common sub-sequence in $A_{1\ldots i}$ and $B_{1\ldots j}$.

Second step is to define a transitive function. DP Recursive Definition.
case 1. if $A_i = B_j$, then

$$dp_{i,j} = dp_{i-1,j-1} + 1, \forall i, j > 0$$

case 2. if $A_i! = B_j$, then

$$dp_{i,j} = MAX(dp_{i-1,j}, dp_{i,j-1}), \forall i, j > 0$$

case 3. if $i = 0$ or $j = 0$, then

$$dp_{i,j} = 0$$

Final step is answer definition: $dp_{n,m}$.