ADA 2022 Tutorial 3

February 10/11, 2022

This tutorial is more warmup on DPs. What we would like you to do for each of the problems is:-

1. Define the subproblems clearly
2. Write a recursion using the above definition and argue properly about why the recursion is correct (this is the optimal substructure property)
3. Implement using tables and argue runtime.

1 Maximum Sum Subarray

You are given an array of size n containing integers. Design a linear time algorithm to find the contiguous sub-array such that the sum of elements is as large as possible.

Solution.

Structure of the Optimal Solution. Think about an optimal solution. The first observation is, it has to *end* somewhere (we do not know where, but there are only n options). Now think about solving the easier problem - Max sum sub-array that ends at a given position i - call this $MSE[i]$. Then $MSE[i]$ has to look like one of the following cases.

1. $MSE[i]$ is just the element $A[i]$
2. $MSE[i]$ is $MSE[i-1]$ plus $A[i]$

Now the recurrence is easy. Just take max of the two. One can implement this using a one-dimensional table in linear time. Also, one can reconstruct the actual solution by backtracking. Finally, the real solution can be found by traversing the table once and finding the maximum $MSE[i]$.

(Note: This can be implemented more cleverly without using a table explicitly. But we do it more elaborately to practice DP)

2 Maximum Monotonically non-decreasing Subsequence

You are given an array of size n containing integers. Design a linear time algorithm to find the sub-array (not necessarily contiguous) of maximum length such the values are non-decreasing.

Solution. Again, let the trick lies in defining the correct subproblem. Let $M[i]$ denote the required subsequence that includes i as the last element. Now, the optimal solution can look like one of the following two cases :

1. $M[i]$ is just the single number $A[i]$ or
2. $M[i]$ is the longest monotonically non-decreasing subsequence that ends at an index $k < i$ plus the element $A[i]$, provided that $A[i] \geq A[k]$

Again, rest is easy. What is the runtime ?

3 Maximum Switching Subsequence

Suppose you are given an array of distinct real numbers, $A[1:n]$. Define a switching subsequence of A as a sequence $A[k_1], A[k_2], \cdots A[k_\ell]$ such that ,

$$A[k_i] < A[k_{i+1}], \text{for odd i}$$
$$A[k_i] > A[k_{i+1}], \text{for even i}$$

In plain English, the sequence switches between increasing and decreasing, starting with increasing. The goal is to find the longest switching sub-sequence of A. Deisgn a linear time algorithm for this.

Solution. The main trick here is to realize that you actually need to define two different subproblems for each $i = 1, 2, \cdots n$. The rest is very similar to the above problem with a minor twist.

Suppose $longestInc(i)$ denote the longest switching subsequence ending at i where the last entry is bigger than the preceding one while $longestDec(i)$ denote the longest switching subsequence ending at i where the last entry is smaller than the preceding one. Then the following is true :

$$longestInc(i) = max_{j<i, A[j]<A[i]} longestDec(i) + 1$$
$$longestDec(i) = max_{j<i, A[j]>A[i]} longestInc(i) + 1$$

Now the rest is again converting this to an iterative solution using table.

(Note: There is a simpler $\mathcal{O}(n)$-time solution to solve this problem. Can you figure that out?)

4 Recurrence using Substitution Method

$$T(n) = 2T(n/3) + T(n/2) + 5n$$