MCS-375: Algorithms: Analysis and Design Handout #DP7
San Skulrattanakulchai
Gustavus Adolphus College Oct 17, 2014

Longest Monotonically Increasing Subsequence

CLRS: Exercise 15.4-5

Let $A : a_1, a_2, \ldots, a_n$ and $B : b_1, b_2, \ldots, b_k$ be sequences of positive integers. We say that B is a *subsequence* of A if $k \leq n$ and there exists an increasing function $i : \{1, 2, \ldots, k\} \to \{1, 2, \ldots, n\}$ such that $b_j = a_{i(j)}$ for all $1 \leq j \leq k$. If B is also monotonically increasing, i.e., $b_j \leq b_{j'}$ whenever $j < j'$, then B is said to be a *monotonically increasing subsequence* of A. A *logest monotonically increasing subsequence* (LMIS) of A is an increasing subsequence of A of maximum length.

Examples. Let A be the sequence $20, 50, 30, 10, 40$. Then $50, 10, 40$ is a subsequence of A even though it is not monotonically increasing. The sequence $10, 40$ is a monotonically increasing subsequence of A even though it is not the longest. Finally, the sequence $20, 30, 40$ is an LMIS of A.

Problem. Given a sequence $A : a_1, a_2, \ldots, a_n$ of positive integers, find an LMIS of A.

Solution by Dynamic Programming.
For each $1 \leq i \leq n$, define $m(i)$ to be the length of any longest monotonically increasing subsequence of $a_1, a_2, \ldots, a_i$ that has a_i as the last element of the subsequence.
We seek $\max\{\, m(i) : 1 \leq i \leq n \,\}$.

Optimal Substructure Property.
If $i = 1$, then a_i is obviously the LMIS of itself. Thus, $m(1) = 1$.
Now suppose $1 < i \leq n$. Fix such an i. Let $B : b_1, b_2, \ldots, b_k$ be an LMIS of $a_1, a_2, \ldots, a_i$ subject to $b_k = a_i$. Then $b_1, b_2, \ldots, b_{k-1}$ must be an LMIS of $a_1, a_2, \ldots, a_{i-1}$. This means that there exists some j, where $1 \leq j \leq i - 1$, such that $b_1, b_2, \ldots, b_{k-1}$ is an LMIS of $a_1, a_2, \ldots, a_j$ subject to $b_{k-1} = a_j$.

Recurrence.
The above reasoning shows $m(i)$ satisfies the recurrence

$$m(i) = \begin{cases} 1 & \text{if } i = 1 \\ \max\{\, 1, m(j) + 1 : 1 \leq j < i \text{ and } a_j \leq a_i \,\} & \text{if } 1 < i \leq n. \end{cases}$$

Algorithm.
The algorithm consists of 3 steps.

Step 1 Fill in the table of $m(\cdot)$ values and a companion table of maximizers.

Step 2 Scan through the $m(\cdot)$ table to find a maximum value. Say $m(i)$ is the maximum.

Step 3 Starting from the ith entry of the table of maximizers stored in Step 1, where i is the maximizer from Step 2, obtain the LMIS by using these maximizers as pointers.

Running Time.
Step 1 takes time $O(n^2)$ since we fill in 2 tables each of size $O(n)$, and each table entry takes time $O(n)$ to compute.
Step 2 takes time $O(n)$ since the size of the table is $O(n)$.
Step 3 takes time $O(n)$ since there are $O(n)$ maximizers and we spend $O(1)$ time per maximizer.
Therefore, total running time is $O(n^2)$.