Lecture Notes

Doina Bein

The longest non-increasing subsequence problem

The *longest non-increasing subsequence* problem is to find a subsequence of a given sequence in which the subsequence's elements are in sorted order, lowest to highest, and in which the subsequence is as long as possible. This subsequence is not necessarily contiguous, or unique.

The *longest non-increasing subsequence* problem can be formulated as follows:

input: n, a positive integer, and the array A of n comparable elements

output: array R that contains the longest non-increasing subsequence

The longest non-increasing subsequence problem is solvable in $O(n\log n)$ time, where n denotes the length of the input sequence.[1]

Let us consider an example (an instance to this problem) by mixing the first 14 Fibonacci numbers[2] with 253 and writing them in reverse order:

143, 233, 55, 89, 21, 34, 8, 13, 3, 5, 1, 2, 253, 0, 1

(I took the first 14 Fibonacci numbers and I swapped the first with the second, the third with the fourth, and so on, then added 253 on the third position, and then reverse everything.)

This input sequence has no eight-member non-increasing subsequences.

A longest non-increasing subsequence is

233, 89, 34, 13, 5, 2, 0

This subsequence has length seven.

The longest non-increasing subsequence in this example is not unique. For instance,

143, 55, 21, 8, 3, 1, 1 or 143, 55, 21, 8, 4, 2, 1

are other non-increasing subsequences of the same length, seven.

The way one can identify a longest non-increasing subsequence for this particular instance of the problem is by selecting the largest value in the array as the first element of the subsequence and then continue to select elements in non-increasing order:

Algorithm Max_then_Non_Increasing

Step 1. Select the maximum value in A and add it to R. Let *pos* be the position of that value in A, i.e. *A[pos]* is the maximum element in A. Let *last* be the last element in R.

Step 2. For each element in A located after *A[pos]:* if the element is less than or equal to *last*, then add it to R, adjust *pos* to be the position of that value in A, and make *last* be the last element in R.

Step 3. Repeat Step 2 until you reach the end of A.

---

[1] Schensted, C. (1961), "Longest increasing and decreasing subsequences", Canadian Journal of Mathematics 13: 179–191, doi:10.4153/CJM-1961-015-3

[2] https://www.mathsisfun.com/numbers/fibonacci-sequence.html

Let us see how the algorithm Max_then_Non_Increasing works on another instance of this problem.

For example, let us consider the first 16 terms of the binary Van der Corput[3] sequence, written in reverse order:

15, 7, 11, 3, 13, 5, 9, 1, 14, 6, 10, 2, 12, 4, 8, 0

For this instance of the problem, we start with the maximum element 15, which happens to be the first element in A, and we add it to R:

R: 0

Then the next element 7 is smaller than 15, so we include it:

R: 15, 7

The third element 11 is larger than 7, the last element in R, so it cannot be included. The fourth element 3 is smaller than 7, the last element in R so we include it:

R: 15, 7, 3

The next three elements, 13, 5, 9, are all larger than 3, the last element in R, so they cannot be included. The eighth element 1 is smaller than 3, the last element in R so we include it:

R: 15, 7, 3, 1

The rest of the elements, except the last one 0, are all larger than 1, the last element in R, so they cannot be included. The last element 0 is smaller than 1, the last element in R we include it:

R: 15, 7, 3, 1, 0

We have obtained a subsequence of five elements that is non-increasing, but it is not the optimal solution for the problem. This input sequence has no seven-member non-increasing subsequences.

A longest non-increasing subsequence is

15, 11, 9, 6, 2, 0.

This subsequence has length six.

The longest non-increasing subsequence in this example is not unique. For instance,

15, 11, 9, 6, 4, 0

or

15, 13, 9, 6, 4, 0

are other non-increasing subsequences of equal length six.

There are many instances, including the example given above, for which this strategy will not work.

So this algorithm does not always produce the correct output.

[3] https://en.wikipedia.org/wiki/Van_der_Corput_sequence

A simple algorithm for the longest non-increasing subsequence problem

There is a relatively simple algorithm that solves the problem and has $O(n^2)$ running time, where n denotes the length of the input sequence. The algorithm uses an additional array H of length *n*, of non-negative integers. The value $H[i]$ will indicate how many elements smaller than or equal to $A[i]$ are further in A could be in a longest non-increasing subsequence. Initially, the array H is set to 0 (all the elements are 0). The algorithm proceeds by trying to modify the values of H starting with the previous to last element and going down to the first element. Then a longest non-increasing subsequence can be identified by selecting the elements of A in non-increasing order of their H values, starting with the largest value in H.

Algorithm End_to_Beginning

Step 1. Set all the values in the array H to 0.

Step 2. Starting with $H[n-1]$ and going down to $H[0]$ try to increase the value of each $H[i]$, $i = n-1..0$, as follows:

Step 3. Starting with index $i+1$ and going up to $n-1$ (the last index in A) repeat Steps 4 and 5:

Step 4. See if any element is smaller than **or equal to** $A[i]$ and has its H value also bigger or equal to $H[i]$.

Step 5. If yes, then $A[i]$ can be followed by that element in a non-increasing subsequence, thus set $H[i]$ to be 1 plus that H value.

Step 6. Calculate the largest (maximum) value in H. By adding 1 to that value we have the length of a longest non-increasing subsequence.

Step 7. Identify a longest non-increasing subsequence by identifying elements in A that have decreasing H values, starting with the largest (maximum) value in H.

Let us see how the algorithm End_to_Beginning works for the instance:

143, 233, 55, 89, 21, 34, 8, 13, 3, 5, 1, 2, 0, 1

The array H has initially the value 0 for all its elements:

index	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]
H	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Leaving blank the cells that are not involved in a computation at some particular step, the array H is modified, starting with index $n-2$ and going down to 0.

Since there are no elements smaller or equal than or equal to $A[n-2]$ that follow $A[n-2]$ in A, then $H[n-2]$ remains 0.

index	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]
A	143	233	55	89	21	34	8	13	3	5	1	2	0	1
H													0	0

We compare now $A[n-3]$ with the elements that follow it in A, namely $A[n-2]$ and $A[n-1]$:

$A[n-3] \geq A[n-2]$ and $H[n-3] \leq H[n-2]$ thus $H[n-3] = H[n-2] + 1 = 1$

$A[n-3] \geq A[n-1]$ but $H[n-3] \not\leq H[n-1]$ so we do not change $H[n-3]$

index	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]
A	143	233	55	89	21	34	8	13	3	5	1	2	0	1
H												1	0	0

We compare now $A[n-4]$ with the elements that follow it in A, namely $A[n-3]$, $A[n-2]$ and $A[n-1]$:

$A[n-4] \not\geq A[n-3]$ so we do not change $H[n-4]$

$A[n-4] \geq A[n-2]$ and $H[n-4] \leq H[n-2]$ thus $H[n-4] = H[n-2] + 1 = 1$

$A[n-4] \geq A[n-1]$ but $H[n-4] \not\leq H[n-1]$ so we do not change $H[n-4]$

We continue until we finish calculating $H[0]$. The array H has now the values:

index	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]
A	143	233	55	89	21	34	8	13	3	5	1	2	0	1
H	6	6	5	5	4	4	3	3	2	2	1	1	0	0

The maximum value in H is 6, so a longest non-increasing subsequence has length 7.

We will look in H for the values 6, 5, 4, 3, 2, 1, and 0.

We look in H for the value 6: the element with index 0 has H[0]=6, thus A[0] is added to R:

R: 143

We look in H for the value 5 and check that is smaller than the last element in R: the element with index 2 has H[2]=5, thus A[2] is added to R:

R: 143, 55

We look in H for the value 4 and check that is smaller than the last element in R: the element with index 4 has H[4]=4, thus A[4] is added to R:

R: 143, 55, 21

We look in H for the value 3 and check that is smaller than the last element in R: the element with index 6 has H[6]=3, thus A[6] is added to R:

R: 143, 55, 21, 8

We look in H for the value 2: the element with index 8 has H[8]=2, thus A[8] is added to R:

R: 143, 55, 21, 8, 3

We look in H for the value 1: the element with index 10 has H[10]=1, thus A[10] is added to R:

R: 143, 55, 21, 8, 3, 1

We look in H for the value 0: the element with index 12 has H[12]=0, thus A[12] is added to R:

R: 143, 55, 21, 8, 3, 1, 0

We are done. We have found a longest non-increasing subsequence for this instance.

Let us see how the algorithm End_to_Beginning works for the second instance:

15, 7, 11, 3, 13, 5, 9, 1, 14, 6, 10, 2, 12, 4, 8, 0

The array H has initially the value 0 for all its elements:

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]	[15]
H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Leaving blank the cells that are not involved in a computation at some particular step, the array H is modified, starting with index $n-2$ and going down to 0.

We compare now $A[n-2]$ with the elements that follow $A[n-2]$ in A, namely $A[n-1]$:

$A[n-2] \geq A[n-1]$ and $H[n-1] \leq H[n-2]$ thus $H[n-2] = H[n-3] + 1 = 1$

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]	[15]
A	15	7	11	3	13	5	9	1	14	6	10	2	12	4	8	0
H															1	0

We compare now $A[n-3]$ with the elements that follow it in A, namely $A[n-2]$ and $A[n-1]$:

$A[n-3] \not\geq A[n-2]$ thus so we do not change $H[n-3]$

$A[n-3] \geq A[n-1]$ and $H[n-3] \leq H[n-1]$ thus $H[n-3] = H[n-1] + 1 = 1$

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]	[15]
A	15	7	11	3	13	5	9	1	14	6	10	2	12	4	8	0
H														1	1	0

We compare now $A[n-4]$ with the elements that follow it in A, namely $A[n-3]$, $A[n-2]$ and $A[n-1]$:

$A[n-4] \geq A[n-3]$ and $H[n-4] \leq H[n-3]$ thus $H[n-4] = H[n-3] + 1 = 2$

$A[n-4] \geq A[n-2]$ and $H[n-4] \not\leq H[n-2]$ so we do not change $H[n-4]$

$A[n-4] \geq A[n-1]$ but $H[n-4] \not\leq H[n-1]$ so we do not change $H[n-4]$

We continue until we finish calculating $H[0]$. The array H has now the values:

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]	[15]
A	15	7	11	3	13	5	9	1	14	6	10	2	12	4	8	0
H	5	3	4	2	4	2	3	1	3	2	2	1	2	1	1	0

The maximum value in H is 5, so a longest increasing subsequence has length 6.

We will look in H for the values 5, 4, 3, 2, 1, and 0.

We look in H for the value 5: the element with index 0 has H[0]=5, thus A[0] is added to R:

R: 15

We look in H for the value 4 and check that is smaller than the last element in R: the element with index 2 has H[2]=4, thus A[2] is added to R:

R: 15, 11

We look in H for the value 3 and check that is smaller than the last element in R: the element with index 6 has H[6]=3, thus A[6] is added to R:

R: 15, 11, 9

We look in H for the value 2 and check that is smaller than the last element in R: the element with index 9 has H[9]=2, thus A[9] is added to R:

R: 15, 11, 9, 6

We look in H for the value 1 and check that is smaller than the last element in R: the element with index 11 has H[11]=1, thus A[11] is added to R:

R: 15, 11, 9, 6, 2

We look in H for the value 0 and check that is smaller than the last element in R: the element with index 15 has H[15]=0, thus A[15] is added to R:

R: 15, 11, 9, 6, 2, 0

We are done. We have found a longest non-increasing subsequence for this instance.