Longest Increasing Subsequence

By Adri Wessels

IOI Training Camp 3 (9-10 March 2019)

The Problem:

Given a sequence (e.g. 10, 2, 6, 13, 4, 5) find a subsequence (e.g. 2, 13, 4) such that the subsequence is the longest (strictly) increasing subsequence.

The Solution:

The solution uses DP.
Let seq be the array containing the sequences.
We let mem be an array where mem[j] stores the index k of the smallest seq[k] such that there is an increasing subsequence of length j ending with seq[k].
We let prev[j] store the second last number in the longest increasing subsequence ending at seq[j].

Now we build up mem by noticing that if seq[i] is less than seq[mem[j]] and seq[i] is greater than seq[mem[j-1]] then mem[j] should be i because then you end the subsequence with a lower number.

The Solution:

Also prev[i] is then set to mem[j-1] because at this point in time the sequence ending with seq[mem[j-1]] is the smallest sequence that is j-1 long and thus the optimal choice to go before I in a subsequence.

We then iterate over the entire list and fill in mem and prev while keeping track of the length of our longest increasing subsequence.

At the end we start at mem[length] and work backwards along the subsequence by using prev to eventually get the full sequence.

The Example:

We will use the commonly used example from Wikipedia which is the sequence:

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

Reminder:

mem[j] = k s.t. s[k] is the smallest last number in an increasing subsequence of length j.

prev[j] = the second last number in the longest increasing subsequence ending at seq[j].

The Example:

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

Current index i = -1

Mem: 0

Prev:

Reminder:

mem[j] = k s.t. s[k] is the smallest last number in an increasing subsequence of length j.

prev[j] = the second last number in the longest increasing subsequence ending at seq[j].

The Example:

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

Current index i = 0

Mem: 0, 0

Prev: 0

Reminder:

mem[j] = k s.t. s[k] is the smallest last number in an increasing subsequence of length j.

prev[j] = the second last number in the longest increasing subsequence ending at seq[j].

The Example:

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

Current index i = 1;

Mem: 0, 0, 1

Prev: 0, 0

Reminder:

mem[j] = k s.t. s[k] is the smallest last number in an increasing subsequence of length j.

prev[j] = the second last number in the longest increasing subsequence ending at seq[j].

The Example:

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

Current index i = 2;

Mem: 0, 0, 2

Prev: 0, 0, 0

Reminder:

mem[j] = k s.t. s[k] is the smallest last number in an increasing subsequence of length j.

prev[j] = the second last number in the longest increasing subsequence ending at seq[j].

The Example:

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

Current index i = 3;

Mem: 0, 0, 2, 3

Prev: 0, 0, 0, 2

Reminder:

mem[j] = k s.t. s[k] is the smallest last number in an increasing subsequence of length j.

prev[j] = the second last number in the longest increasing subsequence ending at seq[j].

The Example:

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

Current index i = 4;

Mem: 0, 0, 4, 3

Prev: 0, 0, 0, 2, 0

Reminder:

mem[j] = k s.t. s[k] is the smallest last number in an increasing subsequence of length j.

prev[j] = the second last number in the longest increasing subsequence ending at seq[j].

The Example:

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

Current index i = 5;

Mem: 0, 0, 4, 5

Prev: 0, 0, 0, 2, 0, 4

Reminder:

mem[j] = k s.t. s[k] is the smallest last number in an increasing subsequence of length j.

prev[j] = the second last number in the longest increasing subsequence ending at seq[j].

The Example:

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

Current index i = 6;

Mem: 0, 0, 4, 6

Prev: 0, 0, 0, 2, 0, 4, 4

Reminder:

mem[j] = k s.t. s[k] is the smallest last number in an increasing subsequence of length j.

prev[j] = the second last number in the longest increasing subsequence ending at seq[j].

The Example:

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

Current index i = 7;

Mem: 0, 0, 4, 6, 7

Prev: 0, 0, 0, 2, 0, 4, 4, 6

Reminder:

mem[j] = k s.t. s[k] is the smallest last number in an increasing subsequence of length j.

prev[j] = the second last number in the longest increasing subsequence ending at seq[j].

The Example:

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

Current index i = 8;

Mem: 0, 0, 8, 6, 7

Prev: 0, 0, 0, 2, 0, 4, 4, 6, 0

Reminder:

mem[j] = k s.t. s[k] is the smallest last number in an increasing subsequence of length j.

prev[j] = the second last number in the longest increasing subsequence ending at seq[j].

The Example:

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

Current index i = 9;

Mem: 0, 0, 8, 6, 9

Prev: 0, 0, 0, 2, 0, 4, 4, 6, 0, 6

Reminder:

mem[j] = k s.t. s[k] is the smallest last number in an increasing subsequence of length j.

prev[j] = the second last number in the longest increasing subsequence ending at seq[j].

The Example:

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

Current index i = 10;

Mem: 0, 0, 8, 10, 9

Prev: 0, 0, 0, 2, 0, 4, 4, 6, 0, 6, 8

Reminder:

mem[j] = k s.t. s[k] is the smallest last number in an increasing subsequence of length j.

prev[j] = the second last number in the longest increasing subsequence ending at seq[j].

The Example:

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

Current index i = 11;

Mem: 0, 0, 8, 10, 9, 11

Prev: 0, 0, 0, 2, 0, 4, 4, 6, 0, 6, 8, 9

Reminder:

mem[j] = k s.t. s[k] is the smallest last number in an increasing subsequence of length j.

prev[j] = the second last number in the longest increasing subsequence ending at seq[j].

The Example:

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

Current index i = 12;

Mem: 0, 0, 8, 12, 9, 11

Prev: 0, 0, 0, 2, 0, 4, 4, 6, 0, 6, 8, 9, 8

Reminder:

mem[j] = k s.t. s[k] is the smallest last number in an increasing subsequence of length j.

prev[j] = the second last number in the longest increasing subsequence ending at seq[j].

The Example:

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

Current index i = 13;

Mem: 0, 0, 8, 12, 9, 13

Prev: 0, 0, 0, 2, 0, 4, 4, 6, 0, 6, 8, 9, 8, 9

Reminder:

mem[j] = k s.t. s[k] is the smallest last number in an increasing subsequence of length j.

prev[j] = the second last number in the longest increasing subsequence ending at seq[j].

The Example:

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

Current index i = 14;

Mem: 0, 0, 8, 12, 14, 13

Prev: 0, 0, 0, 2, 0, 4, 4, 6, 0, 6, 8, 9, 8, 9, 12

Reminder:

mem[j] = k s.t. s[k] is the smallest last number in an increasing subsequence of length j.

prev[j] = the second last number in the longest increasing subsequence ending at seq[j].

The Example:

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

Current index i = 15;

Mem: 0, 0, 8, 12, 14, 13, 15

Prev: 0, 0, 0, 2, 0, 4, 4, 6, 0, 6, 8, 9, 8, 9, 12, 13

Reminder:

mem[j] = k s.t. s[k] is the smallest last number in an increasing subsequence of length j.

prev[j] = the second last number in the longest increasing subsequence ending at seq[j].

The Example:

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

Current index i = 15;

Mem: 0, 0, 8, 12, 14, 13, 15

Prev: 0, 0, 0, 2, 0, 4, 4, 6, 0, 6, 8, 9, 8, 9, 12, 13

Longest Increasing Subsequence:

In reverse: 15

Reminder:

mem[j] = k s.t. s[k] is the smallest last number in an increasing subsequence of length j.

prev[j] = the second last number in the longest increasing subsequence ending at seq[j].

The Example:

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

Current index i = 15;

Mem: 0, 0, 8, 12, 14, 13, 15

Prev: 0, 0, 0, 2, 0, 4, 4, 6, 0, 6, 8, 9, 8, 9, 12, 13

Longest Increasing Subsequence:

In reverse: 15, 11

Reminder:

mem[j] = k s.t. s[k] is the smallest last number in an increasing subsequence of length j.

prev[j] = the second last number in the longest increasing subsequence ending at seq[j].

The Example:

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

Current index i = 15;

Mem: 0, 0, 8, 12, 14, 13, 15

Prev: 0, 0, 0, 2, 0, 4, 4, 6, 0, 6, 8, 9, 8, 9, 12, 13

Longest Increasing Subsequence:

In reverse: 15, 11, 9

Reminder:

mem[j] = k s.t. s[k] is the smallest last number in an increasing subsequence of length j.

prev[j] = the second last number in the longest increasing subsequence ending at seq[j].

The Example:

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

Current index i = 15;

Mem: 0, 0, 8, 12, 14, 13, 15

Prev: 0, 0, 0, 2, 0, 4, 4, 6, 0, 6, 8, 9, 8, 9, 12, 13

Longest Increasing Subsequence:

In reverse: 15, 11, 9, 6

Reminder:

mem[j] = k s.t. s[k] is the smallest last number in an increasing subsequence of length j.

prev[j] = the second last number in the longest increasing subsequence ending at seq[j].

The Example:

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

Current index i = 15;

Mem: 0, 0, 8, 12, 14, 13, 15

Prev: 0, 0, 0, 2, 0, 4, 4, 6, 0, 6, 8, 9, 8, 9, 12, 13

Longest Increasing Subsequence:

In reverse: 15, 11, 9, 6, 2

Reminder:

mem[j] = k s.t. s[k] is the smallest last number in an increasing subsequence of length j.

prev[j] = the second last number in the longest increasing subsequence ending at seq[j].

The Example:

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

Current index i = 15;

Mem: 0, 0, 8, 12, 14, 13, 15

Prev: 0, 0, 0, 2, 0, 4, 4, 6, 0, 6, 8, 9, 8, 9, 12, 13

Longest Increasing Subsequence:

In reverse: 15, 11, 9, 6, 2, 0

Reminder:

mem[j] = k s.t. s[k] is the smallest last number in an increasing subsequence of length j.

prev[j] = the second last number in the longest increasing subsequence ending at seq[j].

The Example:

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

Current index i = 15;

Mem: 0, 0, 8, 12, 14, 13, 15

Prev: 0, 0, 0, 2, 0, 4, 4, 6, 0, 6, 8, 9, 8, 9, 12, 13

Longest Increasing Subsequence:

In reverse: 15, 11, 9, 6, 2, 0

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

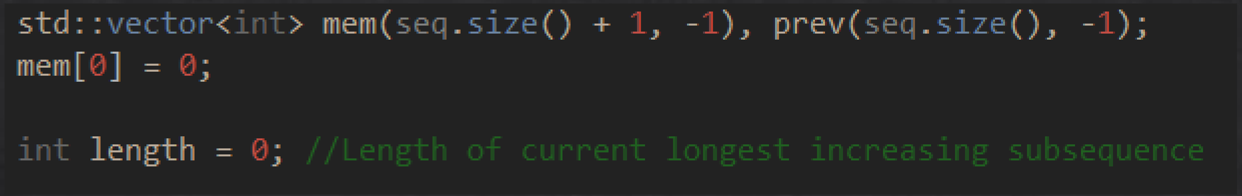
Reminder:

mem[j] = k s.t. s[k] is the smallest last number in an increasing subsequence of length j.

prev[j] = the second last number in the longest increasing subsequence ending at seq[j].

The Code:

The Setup:

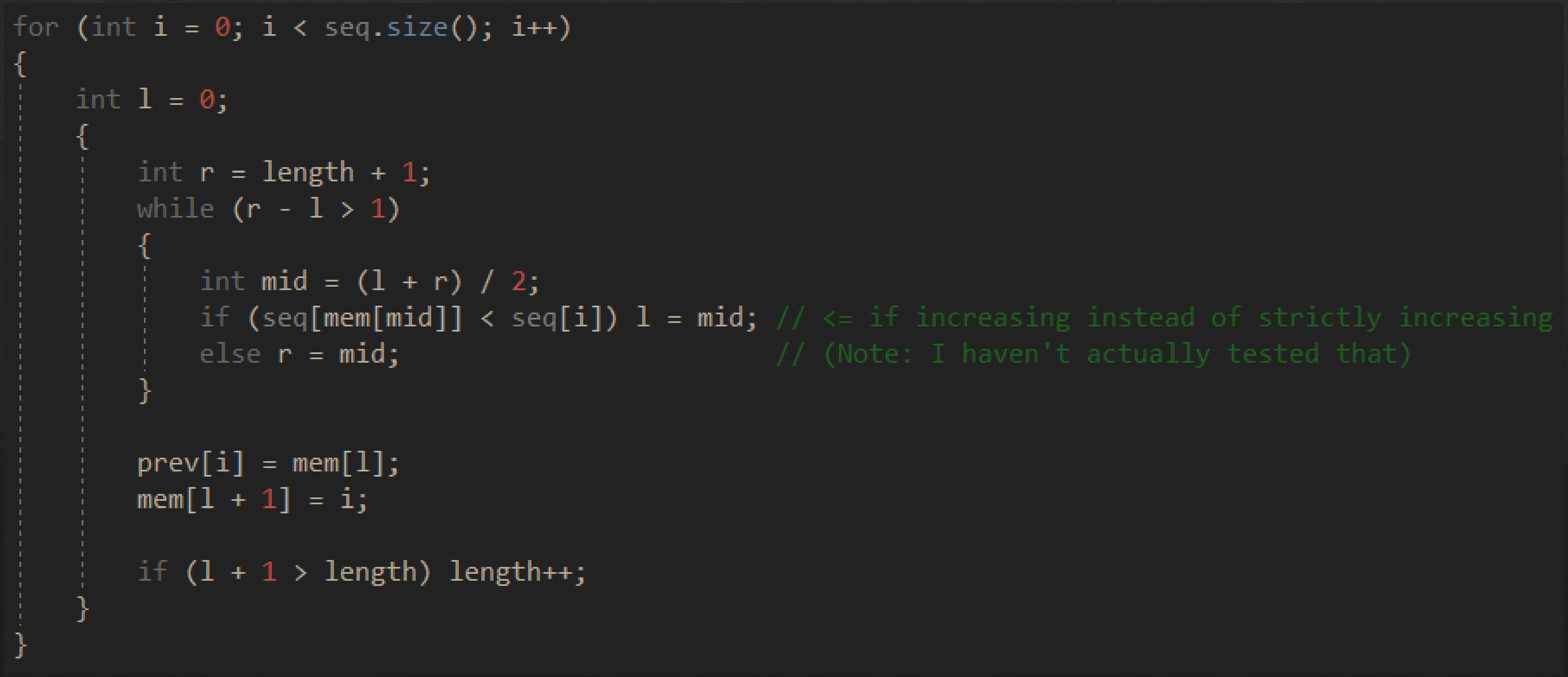


```
std::vector<int> mem(seq.size() + 1, -1), prev(seq.size(), -1);
mem[0] = 0;

int length = 0; //Length of current longest increasing subsequence
```

The Code:

The Loop:



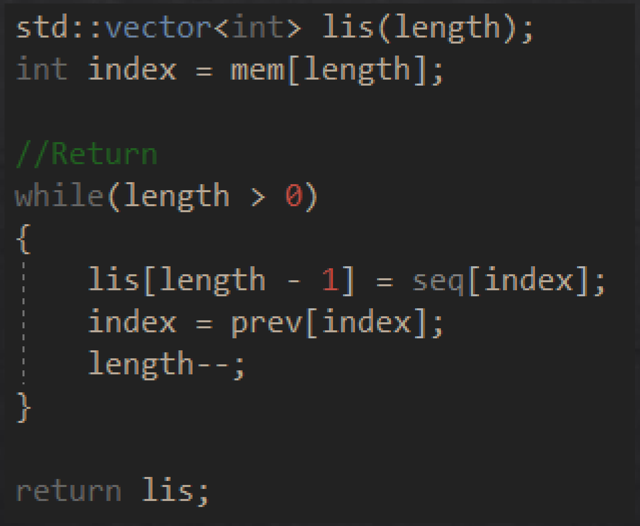
```
for (int i = 0; i < seq.size(); i++)
{
    int l = 0;
    {
        int r = length + 1;
        while (r - l > 1)
        {
            int mid = (l + r) / 2;
            if (seq[mem[mid]] < seq[i]) l = mid; // <= if increasing instead of strictly increasing
            else r = mid;                        // (Note: I haven't actually tested that)
        }

        prev[i] = mem[l];
        mem[l + 1] = i;

        if (l + 1 > length) length++;
    }
}
```

The Code:

The Result:



```
std::vector<int> lis(length);
int index = mem[length];

//Return
while(length > 0)
{
    lis[length - 1] = seq[index];
    index = prev[index];
    length--;
}

return lis;
```