Longest increasing subsequence

By Taariq Mowzer

What is an increasing subsequence?

Given a sequence a subsequence is a ordered subset of the subsequence…
Probably.

E.G given [5, 12, 6, 2 ,1 ,9, 1]

The following are subsequences: [12], [6, 1], [5, 12, 1, 1] and [5, 12, 6, 2]

The following are not [1, 6] and [12, 5, 1]

A increasing subsequence is a subsequence that is increasing*.

I’ll be working with strictly increasing

How to find an increasing subsequence?

Brute Force of course!

[5, 1, 3, 2]

Is just:

- [5], [1], [3], [2]
- [5, 1], [5, 3], [5, 2], [1, 3], [1, 2], [3, 2]
- [5, 1, 3], [5, 1, 2], [5, 3, 2], [1, 3, 2]
- [5, 1, 3 , 2]

Brute Force of course!

- [5, 1, 3, 2]
- Removing all subsequences that aren’t increasing results in:
- [5], [1], [3], [2]
- [1, 3], [1, 2]
- [1, 3, 2]

And it is clear to see that [1, 3, 2] is the largest increasing subsequence

And it is only $O(2^n)$

End

But wait

What if we have more than 25 elements?

Then consider the following algorithm which operates in $O(n^2)$

Let our sequence of numbers be stored in an array called ‘X’.

We will proceed with dynamic programming.

Let mem[j] store the index k of the smallest X[k] such that there is an increasing subsequence ending at k.

Let prev[j] store the predecessor of X[k] in the longest increasing subsequence of size j ending at X[k]

mem[j]: is the index k with smallest X[k] such that is an increasing subsequence of length j ending at k

prev[k]: is the predecessor of X[k]

Notice **if** we find some **X[i] < X[mem[j]]**, then **mem[j] = i** ,as X[i] is smaller

Also prev[i] = mem[j - 1]

We will now iterate over the entire list X and update mem and prev

mem[j]: is the index k with smallest X[k] such that is an increasing subsequence of length j ending at k
prev[k]: is the predecessor of X[k]

X = [0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15]
mem = [0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]
prev = [-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]

mem[j]: is the index k with smallest X[k] such that is an increasing subsequence of length j ending at k
prev[k]: is the predecessor of X[k]

X = [0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15]
mem = [0, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]
prev = [0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]

0

mem[j]: is the index k with smallest X[k] such that is an increasing subsequence of length j ending at k
prev[k]: is the predecessor of X[k]

X = [0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15]
mem = [0, 0, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]
prev = [0, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]

0

0, 8

mem[j]: is the index k with smallest X[k] such that is an increasing subsequence of length j ending at k
prev[k]: is the predecessor of X[k]

X = [0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15]
mem = [0, 0, 2, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]
prev = [0, 0, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]

0

~~0, 8~~ 0,4

mem[j]: is the index k with smallest X[k] such that is an increasing subsequence of length j ending at k
prev[k]: is the predecessor of X[k]

X = [0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15]
mem = [0, 0, 2, 3, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]
prev = [0, 0, 0, 2, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]

0

0,4

0, 4, 12

mem[j]: is the index k with smallest X[k] such that is an increasing subsequence of length j ending at k
prev[k]: is the predecessor of X[k]

X = [0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15]
mem = [0, 0, 4, 3, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]
prev = [0, 0, 0, 2, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]

0

~~0,4~~ 0, 2

0, 4, 12

mem[j]: is the index k with smallest X[k] such that is an increasing subsequence of length j ending at k
prev[k]: is the predecessor of X[k]

X = [0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15]
mem = [0, 0, 4, 3, 5, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]
prev = [0, 0, 0, 2, 0, 4, -1, -1, -1, -1, -1, -1, -1, -1, -1]

0

0, 2

~~0, 4, 12~~ 0, 4, 10

mem[j]: is the index k with smallest X[k] such that is an increasing subsequence of length j ending at k
prev[k]: is the predecessor of X[k]

X = [0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15]
mem = [0, 0, 4, 3, 6, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]
prev = [0, 0, 0, 2, 0, 4, 4, -1, -1, -1, -1, -1, -1, -1, -1]

0

0, 2

~~0, 4, 10~~ **0, 2,** 6

mem[j]: is the index k with smallest X[k] such that is an increasing subsequence of length j ending at k
prev[k]: is the predecessor of X[k]

X = [0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15]
mem = [0, 0, 4, 3, 6, 7, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]
prev = [0, 0, 0, 2, 0, 4, 4, 6, -1, -1, -1, -1, -1, -1, -1]

0

0, 2

0, 2, 6

0, 2, 6, 14

mem[j]: is the index k with smallest X[k] such that is an increasing subsequence of length j ending at k
prev[k]: is the predecessor of X[k]

X = [0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15]
mem = [0, 0, 8, 3, 6, 7, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]
prev = [0, 0, 0, 2, 0, 4, 4, 6, 0, -1, -1, -1, -1, -1, -1]

0

~~0, 2~~ 0, 1

0, 2, 6

0, 2, 6, 14

mem[j]: is the index k with smallest X[k] such that is an increasing subsequence of length j ending at k
prev[k]: is the predecessor of X[k]

X = [0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15]
mem = [0, 0, 8, 3, 6, 9, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]
prev = [0, 0, 0, 2, 0, 4, 4, 6, 0, 6, -1, -1, -1, -1, -1]

0

0, 1

0, 2, 6

~~0, 2, 6, 14~~ 0, 2, 6, 9

mem[j]: is the index k with smallest X[k] such that is an increasing subsequence of length j ending at k
prev[k]: is the predecessor of X[k]

X = [0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15]
mem = [0, 0, 8, 10, 6, 9, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]
prev = [0, 0, 0, 2, 0, 4, 4, 6, 0, 6, 8, -1, -1, -1, -1]

0

0, 1

~~0, 2, 6~~ 0, 1, 5

0, 2, 6, 9

mem[j]: is the index k with smallest X[k] such that is an increasing subsequence of length j ending at k
prev[k]: is the predecessor of X[k]

X = [0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15]
mem = [0, 0, 8, 10, 6, 9, 11, -1, -1, -1, -1, -1, -1, -1, -1, -1]
prev = [0, 0, 0, 2, 0, 4, 4, 6, 0, 6, 8, 9, -1, -1, -1]

0

0, 1

0, 1, 5

0, 2, 6, 9

0, 2, 6, 9, 13

mem[j]: is the index k with smallest X[k] such that is an increasing subsequence of length j ending at k
prev[k]: is the predecessor of X[k]

X = [0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15]
mem = [0, 0, 8, 12, 6, 9, 11, -1, -1, -1, -1, -1, -1, -1, -1, -1]
prev = [0, 0, 0, 2, 0, 4, 4, 6, 0, 6, 8, 9, 8, -1, -1]

0

0, 1

~~0, 1, 5~~ 0, 1, 3

0, 2, 6, 9

0, 2, 6, 9, 13

mem[j]: is the index k with smallest X[k] such that is an increasing subsequence of length j ending at k
prev[k]: is the predecessor of X[k]

X = [0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15]
mem = [0, 0, 8, 12, 6, 9, 13, -1, -1, -1, -1, -1, -1, -1, -1, -1]
prev = [0, 0, 0, 2, 0, 4, 4, 6, 0, 6, 8, 9, 8, 9, -1]

0

0, 1

0, 1, 3

0, 2, 6, 9

~~0, 2, 6, 9, 13~~ 0, 2, 6, 9, 11

mem[j]: is the index k with smallest X[k] such that is an increasing subsequence of length j ending at k
prev[k]: is the predecessor of X[k]

X = [0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15]
mem = [0, 0, 8, 12, 6, 14, 13, -1, -1, -1, -1, -1, -1, -1, -1, -1]
prev = [0, 0, 0, 2, 0, 4, 4, 6, 0, 6, 8, 9, 8, 9, 12]

0

0, 1

0, 1, 3

~~0, 2, 6, 9~~ 0, 2, 6, 7

0, 2, 6, 9, 11

mem[j]: is the index k with smallest X[k] such that is an increasing subsequence of length j ending at k
prev[k]: is the predecessor of X[k]

X = [0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15]
mem = [0, 0, 8, 12, 6, 14, 13, 15, -1, -1, -1, -1, -1, -1, -1, -1]
prev = [0, 0, 0, 2, 0, 4, 4, 6, 0, 6, 8, 9, 8, 9, 12, 13]

0

0, 1

0, 1, 3

0, 2, 6, 7

0, 2, 6, 9, 11

0, 2, 6, 9, 11, 15

mem[j]: is the index k with smallest X[k] such that is an increasing subsequence of length j ending at k
prev[k]: is the predecessor of X[k]

X = [0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15]
mem = [0, 0, 8, 12, 6, 14, 13, 15, -1, -1, -1, -1, -1, -1, -1, -1]
prev = [0, 0, 0, 2, 0, 4, 4, 6, 0, 6, 8, 9, 8, 9, 12, 13]

0

0, 1

0, 1, 3

0, 2, 6, 7

0, 2, 6, 9, 11

0, 2, 6, 9, 11, 15 <-- The answer !

Notice when we look for where our X[i] will go instead of running a for loop over mem, we can binary search

This reduces the time complexity from $O(n^2)$ to $O(n\log n)$

```python
n = len(X)
mem = [0]
prev = []
for i in range (n):
    mem.append(-1)
    prev.append(-1)

li = 0
# li keeps track of the length of our LIS
for i in range (n):
    hi = li + 1
    lo = 0
    while hi - 1 > lo:
        mid = (lo + hi)//2
        if X[mem[mid]] < X[i]:
            # if you want to find the LIS with
            #increasing instead of strictly increasing
            # change the above to X[mem[mid]] - 1 < X[i]
            lo = mid
        else:
            hi = mid
    #When the binary search ends 'lo' points 1
    #to the left of the element we want


    prev[i] = mem[lo]
    mem[lo + 1] = i
    # Checks and increases li if our LIS increased
    if lo + 1 > li:
        li += 1

s = []
# gives the LIS in the form of 's'
for i in range (li):
    s.append(0)
k = mem[li]
for i in range (li - 1, -1, -1):
    s[i] = X[k]
    k = prev[k]
print (s)
```