17 Sept 2021

* check discussion board!! regularly

- Exam → everything through Monday of this week!
  → one piece of paper, both sided, handwritten notes.
  → see D2L POST for page 1.
- Homeworks → see Gradescope for indiv. feedback
  → see D2L for global feedback
  → feel free to email / post improvements to sol'n for feedback.
- misc assignments
  → point Based
  → follow pushes to website + posts on D2L
  → will post soon: Oct 15-16 Full Workshop on Computational Geometry. (morning & early afternoon)
- next week: we begin Dynamic Programming (Ch. 3)
  → others may join
  → I'll post a zoom in case you want to join remotely.

Longest Increasing Subsequence

A B A B C X Y G A T

- sequence: a (multi) set with an order to the elements. (here, think finite so we can rep. it in an array) that order.
- subsequence: a subset of a seq, preserving
  e.g., BAT is a subsequence
  but TAG is not.
- substring: a contiguous subsequence.
  eg., BAT is a subseq, not a substring
  XYG is a substring & a subsequence.

Given: A (finite) sequence of ordered values

Want: Find the longest subsequence s.t. the values are increasing

e.g., 1 5 2 3 7

increasing subsequences:

1
5
2
3
7
15
157
57
23
237
37
1237 longest!!

note: subsequence is obtained by choosing a subset of things to delete from the given sequence.

My first pass: try <u>everything</u>

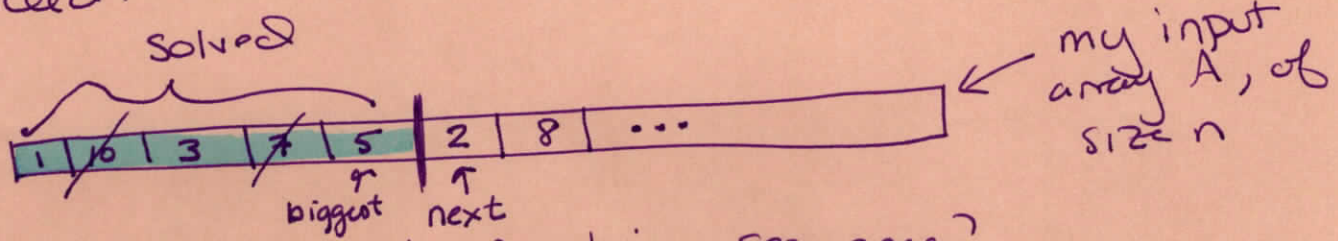
#choices:

$$\underbrace{2}_{\text{try keep + not keep}} \quad \underbrace{2}_{+,-} \quad \underbrace{2}_{+,-} \quad \underbrace{2}_{} = 2^n$$

possible subseq. to try.

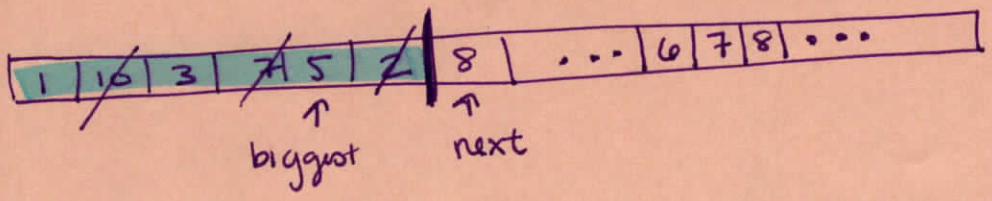
(Hopefully, we can eventually do better).

Now, think of this recursively. I've "solved" it for a part + need to figure out how best to proceed:



Q: How do I best extend this sequence?
(Note: don't think smart, think brute force)

→ look at next. If $A[next] \leq A[biggest]$, move on



→ otherwise, need to try both including the next one + skipping over it

In algo, j = next, the index of the next thing we are processing
i = biggest, the index of the largest value of the already solved part of it

def. as a recursive fcn

$$LISbigger(i,j) = \begin{cases} 0 & \text{if } j > n \leftarrow \text{base case} \\ LISbigger(i, j+1) & \text{if } A[i] \geq A[j] \quad \text{skip the next b/c would violate inc. sequence} \\ \max\left\{ \begin{array}{l} LISbigger(i, j+1) \\ 1 + LISbigger(j, j+1) \end{array} \right\} & \text{otherwise} \end{cases}$$

Alternatively, if you prefer pseudocode:

← my helper fcn aux. fcn. the recursive formulation of a generic call "in the middle"

```
LISBIGGER(i, j):
1:  if j > n
2:      return 0                                  } base case
3:  else if A[i] ≥ A[j]
4:      return LISBIGGER(i, j + 1)
5:  else
6:      skip ← LISBIGGER(i, j + 1)                ← try not including this val
7:      take ← LISBIGGER(j, j + 1) + 1            ← try including it
8:      return max{skip, take}
```

In groups: ① try on a small example.

② Augment the fcn to return the actual sequence. Currently, it just returns the length of the subsequence!

③ Give example of non-unique sol'n.

What does the first call to this look like?

→ Assume A is an array in shared memory of n comparable elts in it A[1....n]

→ Assume $-\infty$ is a value such that $-\infty < k$ for all values in A.

```
A[0] ← -∞
return LisBigger(0, 1)
```

(called a sentinal value)

What is the recurrence relation for LisBigger?

$$T(n) = \Theta(1) + 2T(n-1) \Rightarrow T(n) \in \Theta(2^n)$$

We've seen this in Hanoi!