FINAL PRACTICE!

Problem Solving with Computers-II

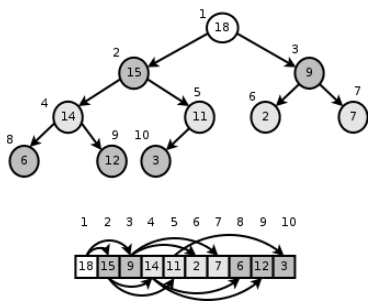
Link to class handout:

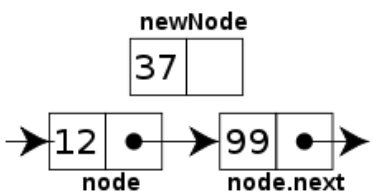
https://bit.ly/one-problem-to-rule-them-all

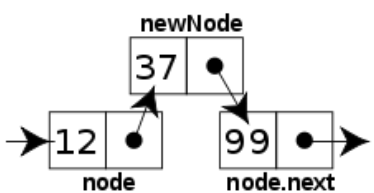
I can deal with pressure, and deadlines.

Resources for the Final Exam

- Leet code problems categorized to help you focus on the most relevant ones.
  - Those marked as “Optional challenge” won’t be on the final
- Review session led by LAs and TAs, this Sunday (over Zoom) at 10a
- Code from lectures: https://github.com/ucsb-cs24-s25/cs24-s25-lectures
- Past Exams: Available on Canvas
- Tool to visualize data structures: https://visualgo.net/







INSERTION-SORT(A)	*cost*	*times*
1 **for** $j = 2$ **to** $A.length$	c_1	n
2 $key = A[j]$	c_2	$n-1$
3 // Insert $A[j]$ into the sorted sequence $A[1..j-1]$.	0	$n-1$
4 $i = j-1$	c_4	$n-1$
5 **while** $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
6 $A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n}(t_j-1)$
7 $i = i-1$	c_7	$\sum_{j=2}^{n}(t_j-1)$
8 $A[i+1] = key$	c_8	$n-1$

Data Structures and C++

Complexity Analysis

One problem to rule them all!

Problem (LP04): Longest Consecutive Sequence (LCS)

Description: Given an unsorted array of integers `nums`, return the length of the longest consecutive elements sequence.

Try solving it in as many different ways as you can, then pick the best!

Ex: `nums = [100, 4, 200, 1, 3, 2]`

Output: 4

because the LCS is [1,2,3,4]

`nums = [1, 0, 1, 2]`

Output: 3

because the LCS is [0,1,2]

Approach 1 : Sort + scan

```
nums = [100, 4, 200, 1, 3, 2]
```

sorted nums: [1, 2, 3, 4, 100, 200]

i	nums[i]+1 == nums[i+1] ?
0	1 + 1 == 2 ✓
1	2 + 1 == 3 ✓
2	3 + 1 == 4 ✓
3	4 + 1 == 100 X
4	100 + 1 == 200 X

update maxlen to be seqlen
reset seqlen to 1

```
nums = [1, 0, 1, 2]
```

[0, 1, 1, 2]

seqlen = 1 maxlen = 0

2 [seqlen++]

3

4

1

1

maxlen = max(0,4) = 4

maxlen = max(4,1) = 4

Approach 2

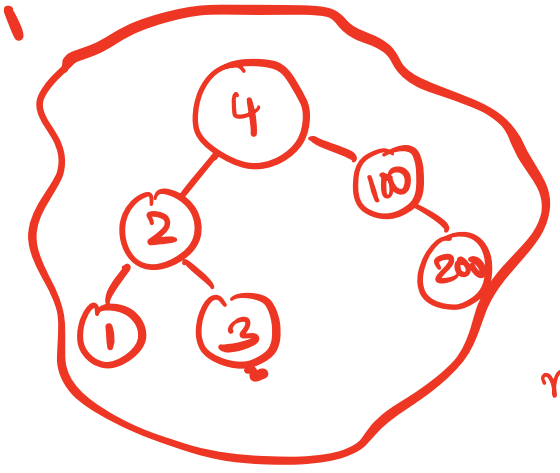
```
nums = [100, 4, 200, 1, 3, 2]
```

```
nums = [1, 0, 1, 2]
```

Iterate in order through set(s): for(num: nums)

num	find(num+1)	seqlen = 1
1	find(2)	2
2	find(3) ✓	3
3	find(4) ✓	4
4	find(5) X	1
100	find(101) X	1
200	find(201) X	1

Set s (balanced BST)



maxlen=0

maxlen= 4

maxlen= 4

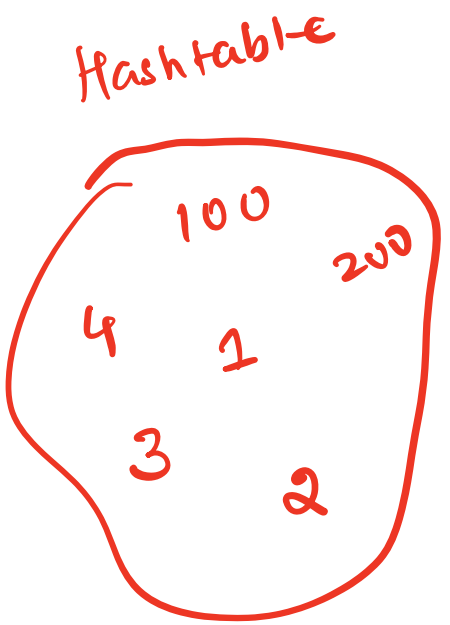
maxlen= 4

Approach 3

No sorting: hop to consecutive keys in a HashTable

```
nums = [100, 4, 200, 1, 3, 2]
```

Iterate through the hashtable.
Figure out what to do in each iteration.



num

4

find(3)
find(2)
find(1) ← start of seq.
find(0) X

seqlen

1

find(2) — 2
find(3) 3
find(4) 4
find(5) 1

Approach 4 ? DFS through an unordered map

nums = [7,4,8,5,3,1,2]

Put all keys in an unordered_map with initial counts = 1

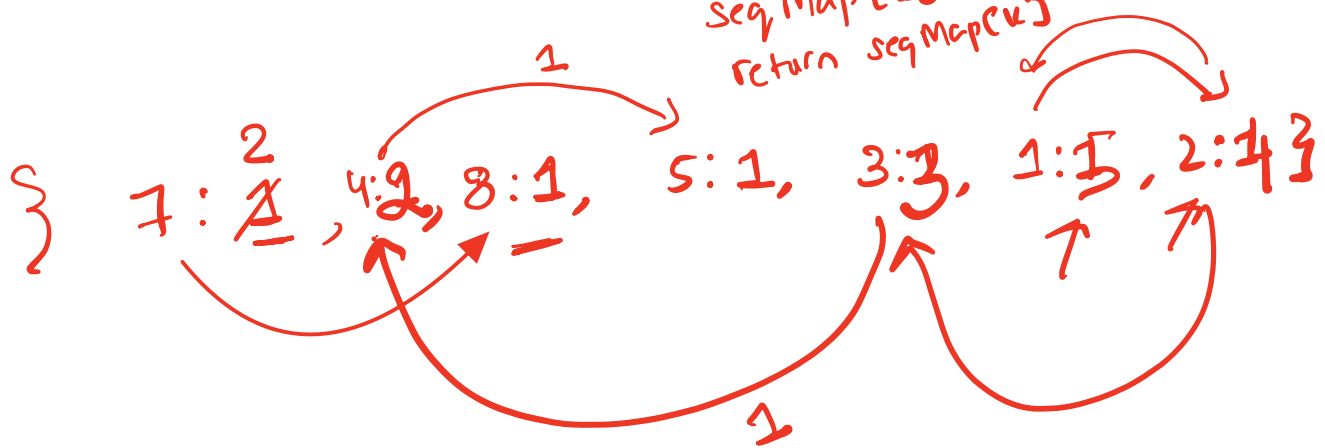
The unordered-map (seqMap) stores key: length of LCS(key)

seqMap = {7:1, 4:1, 8:1, 5:1, 3:1, 1:1, 2:1}

Iterate through the map.

For each key k,

```
lcs( seqMap, k):
    if (seqMap.count(k) == 0) return 0
    if (seqMap[k] > 1) return seqMap[k] // Use precomputed value.
    seqMap[k] = 1+ lcs(k+1)
    return seqMap[k]
```



To compute seqlen starting at key 7 we computed seqlen starting at 8 along the way (DFS) & skipped over 8 in future iterations!

Reflection + Q&A

- Which approach did you try first, and why?
  - Did exploring other methods change how you thought about the problem?

- Which tool or idea resonated most with you?
  - If you had to solve this under time pressure, which approach would you pick and why?

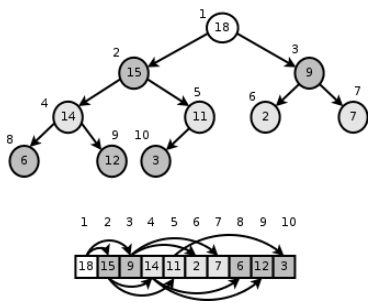
Questions?

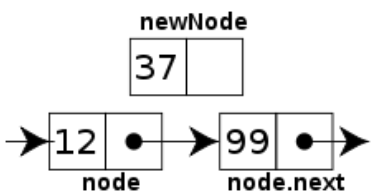
CS24: Learning goals

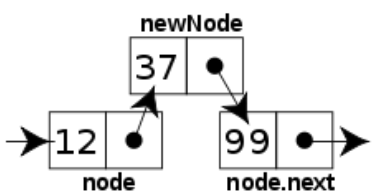
Correct coding, clear thinking, no AI shortcuts

- Design and implement **larger programs** that **run fast**
- Organize **data** in programs using **data structures**
- **Analyze** the **complexity** of your programs
- Prep for **technical interviews**

Grow confident in your problem solving skills!







INSERTION-SORT(A)	*cost*	*times*
1 **for** $j = 2$ **to** $A.length$	c_1	n
2 $key = A[j]$	c_2	$n-1$
3 // Insert $A[j]$ into the sorted sequence $A[1..j-1]$.	0	$n-1$
4 $i = j-1$	c_4	$n-1$
5 **while** $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
6 $A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n}(t_j-1)$
7 $i = i-1$	c_7	$\sum_{j=2}^{n}(t_j-1)$
8 $A[i+1] = key$	c_8	$n-1$

Data Structures and C++

Complexity Analysis

Break: Please take a moment to fill the course evaluations!

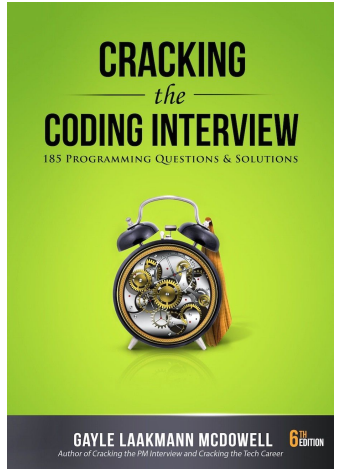
PROBLEM SOLVING II

Student-FO

https://go.blueja.io/WD3FDfJ2QUqKjETiskpCNw

To access the evaluation, scan this QR code with your mobile phone.

Tips for Technical Interviews



1. Listen carefully
2. Draw an example
3. State the brute force or a partially correct solution
    - then work to get at a better solution
4. Optimize:
    - Make time-space tradeoffs to optimize runtime
    - Precompute information: Reorganize the data e.g. by sorting
5. Solidify your understanding of your algo before diving into writing code.
6. Start coding!

Thank you and all the best !