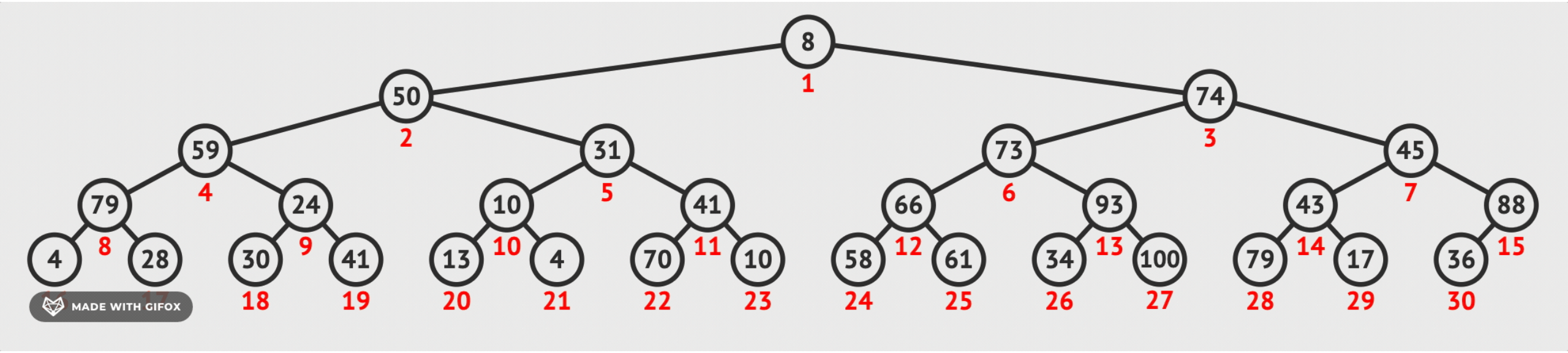
HEAP CODING DEMO INTERVIEW PRACTICE

Problem Solving with Computers-II

Heapify: A fast way to turn an arbitrary vector to a heap



High-level approach: Given an arbitrary vector of keys. Starting from the internal node with the largest index in the vector, and moving upwards in the tree through all the internal nodes (level by level), sift the root of each subtree downward as in the **bubble-down process** until the **heap property** is restored.

Heapify the vector below to convert it into a max-heap (3 min)

1	5	3	6	4	1	7	8	4

What is the resulting vector?

A. 8 7 6 5 4 4 3 1 1

B. 8 1 7 5 4 1 3 6 4

C. 8 6 7 5 4 1 3 1 4

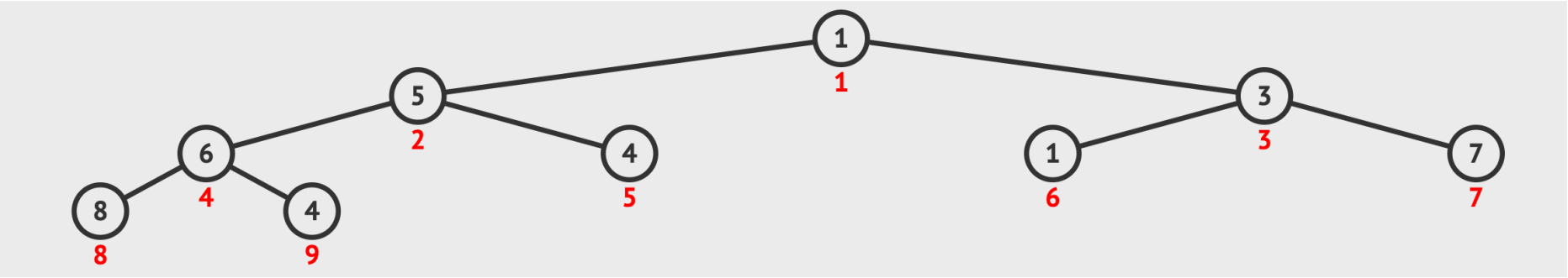
D. Something else

Heap Sort Algorithm

- Step 1: Heapify the input vector with n keys
- Step 2: Let S be the number of keys in the heap. Extract the max element (root key) by swapping it with the last key in the vector. Reduce the size of the heap by 1. At this point, the first (S - 1) keys in the vector represent the heap and the remaining are the sorted portion of the vector Finally, restore the heap property of the root using the bubble down process
- Repeat step 2 while the size of the heap is greater than 1.

Activity 3: Running time of heapify (10 min)

In a complete binary tree of height h, how many total nodes are there in levels 0 to ℓ



Assume $\ell <= h - 1$

A. 2^{ℓ}

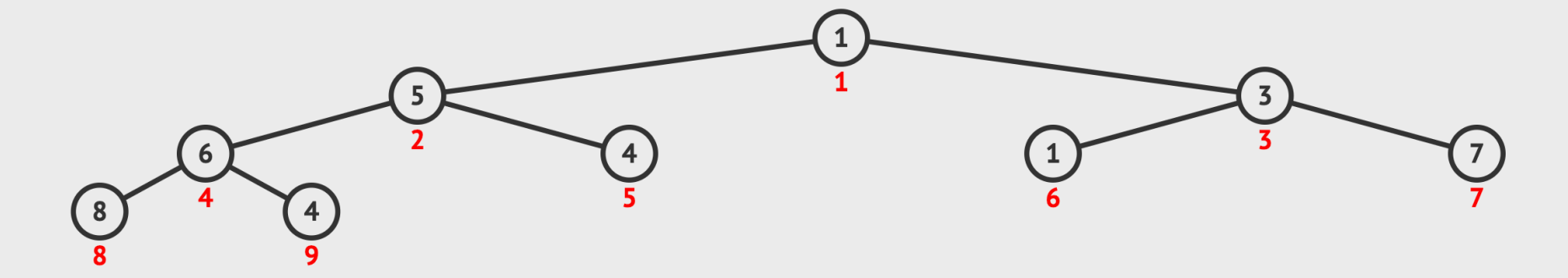
B. $2^{\ell} + 1$

C. $2^{\ell} + 1 - 1$

D. $2^{h} + 1 - 1$

E. Something else

In a complete binary tree of height h, what is the index of the last node at level ℓ



Assume $\ell <= h - 1$

A. $2^{\ell} - 1$

B. $2^{\ell + 1} - 1$

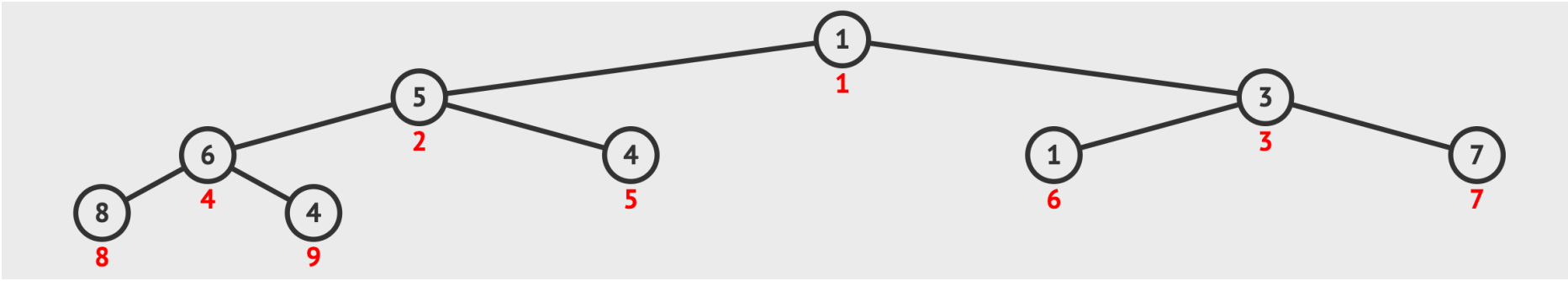
C. $2^{\ell + 1} - 2$

D. $2^{h + 1} - 2$

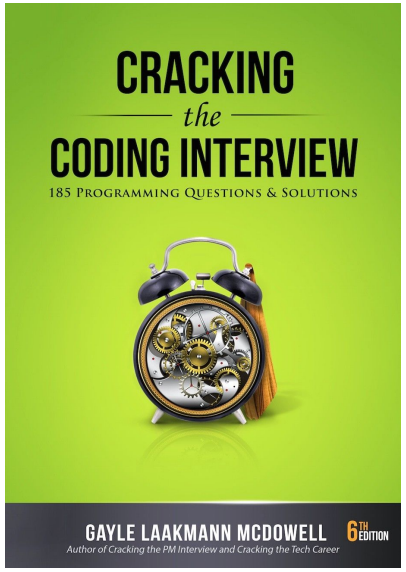
E. Something else

In a complete binary tree of height h, what is the index of the first node at level ℓ

Assume $\ell <= h - 1$



Tips for Technical Interviews

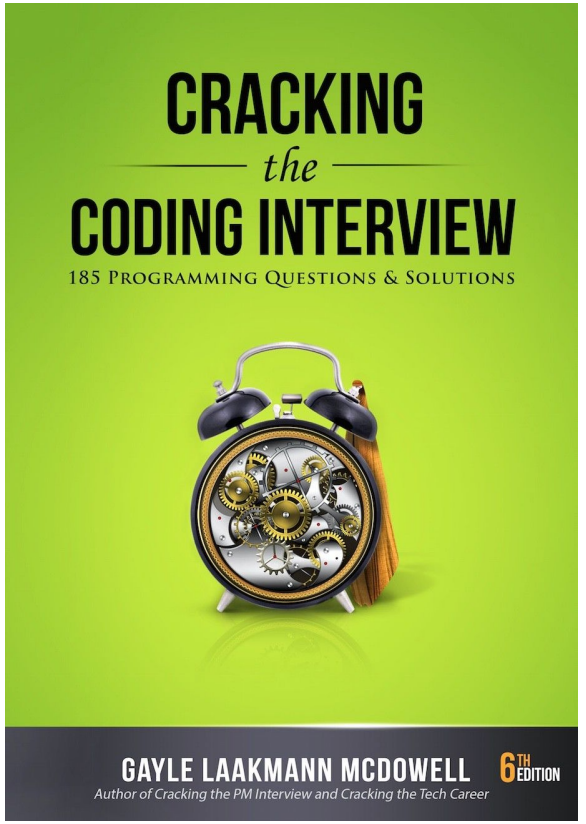


1. Listen carefully
2. Draw an example
3. State the brute force or a partially correct solution
    - then work to get at a better solution
4. Optimize:
    - Make time-space tradeoffs to optimize runtime
    - Precompute information: Reorganize the data e.g. by sorting
5. Solidify your understanding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

Interview practice!

Write a ADT called minStack that provides the following methods

- push() // inserts an element to the “top” of the minStack
- pop() // removes the last element that was pushed on the stack
- top () // returns the last element that was pushed on the stack
- min() // returns the minimum value of the elements stored so far



Practice the interview tips:

- Draw/solve a small example! (2 min)
  - Think of the most straightforward approach (1 min)
  - Evaluate its performance (1 min)
  - Think of another approach and evaluate it (5 min)
    - Can you trade off space/memory for better runtime?
- Pick the most promising approach and start coding! (10 min)