HEAP CODING DEMO
INTERVIEW PRACTICE

Problem Solving with Computers-II

```
#include <iostream>
using namespace std;

int main(){
    cout<<"Hola Facebook\n";
    return 0;
}
```

How is PA 2 going?

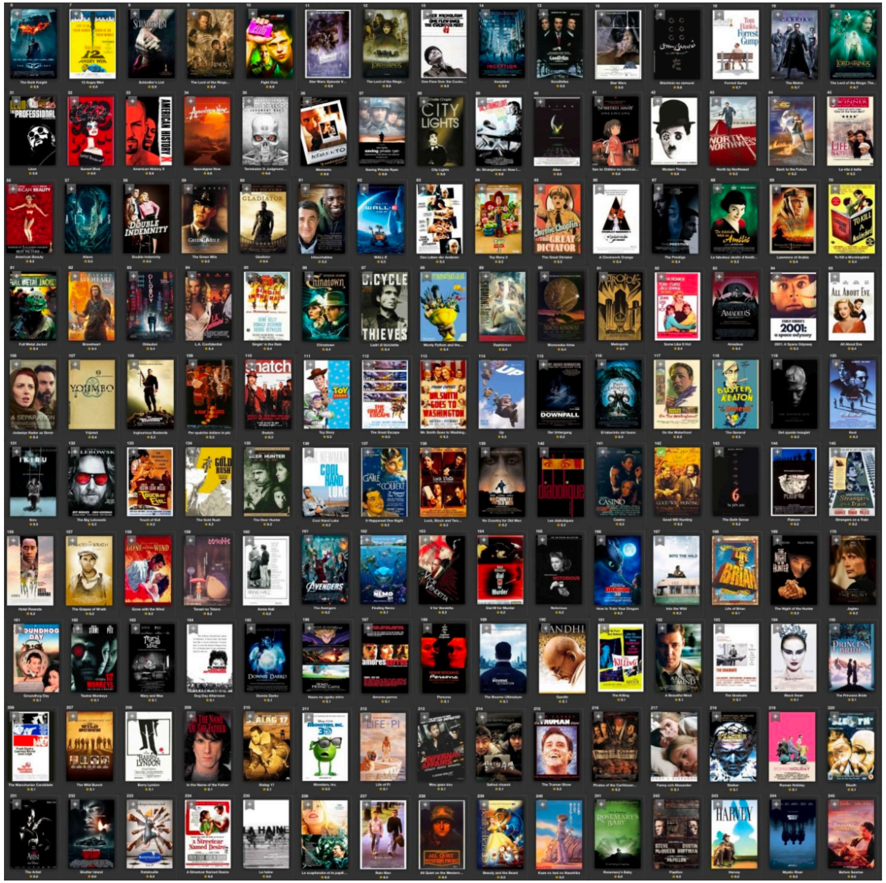
A. Done! **B.** On track to finish and having fun. **C.** On track to finish but struggling (a bit).

D. Falling behind and struggling a lot. **E.** Haven't read the assignment.

I can deal with pressure, and deadlines.

DO YOU HAVE AN IDEA FOR YOUR STORY YET?
NO, I'M WAITING FOR INSPIRATION.
YOU CAN'T JUST TURN ON CREATIVITY LIKE A FAUCET. YOU HAVE TO BE IN THE RIGHT MOOD.
WHAT MOOD IS THAT?
LAST-MINUTE PANIC.

Reminder: PA 2 (Due Fri: 12/08)



Learning goals:

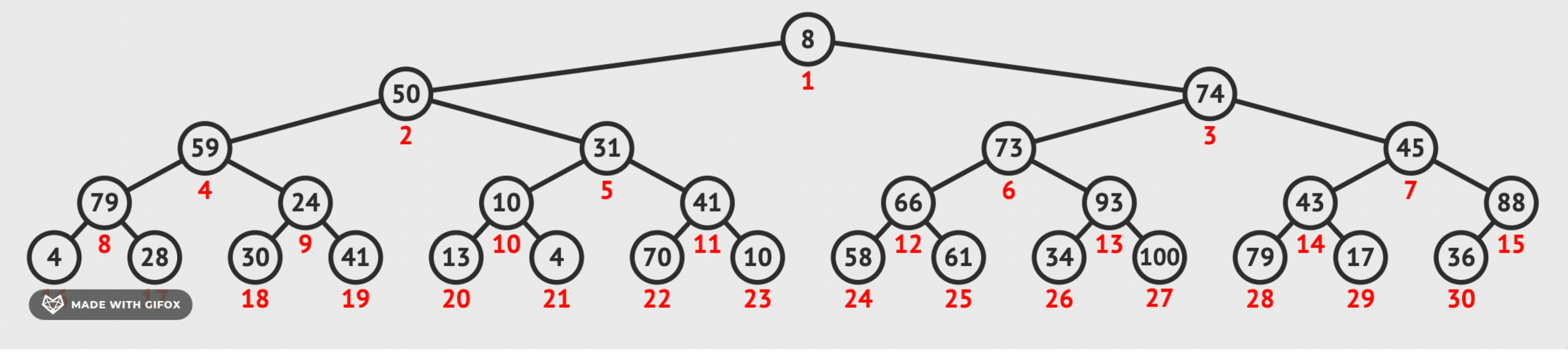
- Answer questions about a movie data set by selecting suitable data structure(s)
- Evaluate the time and space complexity of your solution(s)
- Use the C++ Standard Template Library (STL)
- Think creatively to improve running time of your solution

Helpful links for PA 2

- Find information by reading C++ documentation:
  - **set**: https://www.cplusplus.com/reference/set/set/set/
  - **vector**: https://www.cplusplus.com/reference/vector/vector/?kw=vector
  - **list**: https://cplusplus.com/reference/list/list/?kw=list
  - **stack**: https://cplusplus.com/reference/stack/stack/?kw=stack
- Evaluate the time and space complexity of your solutions
  - https://www.cplusplus.com/reference/set/set/find/

    https://www.cplusplus.com/reference/set/set/lower_bound

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



See lecture code

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

See lecture code

- 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)

Let n be the no. of nodes

Let h be the height of the tree representing the heap

Note: if our count is off by a constant term, the final answer doesn't change still $O(n)$

$T(n)$: Running time of heapify

$$= \sum_{l=0}^{h-1} 2^{l} \cdot (h-l+1)$$

Let $j = h-l$

$$= \sum_{j=1}^{h} 2^{h-j} \cdot (j+1)$$

$$= 2^{h} \sum_{j=1}^{h} j \cdot 2^{-j} + 2^{h} \underbrace{\sum_{0}^{h} 2^{-j}}_{\text{converges to } 2}$$

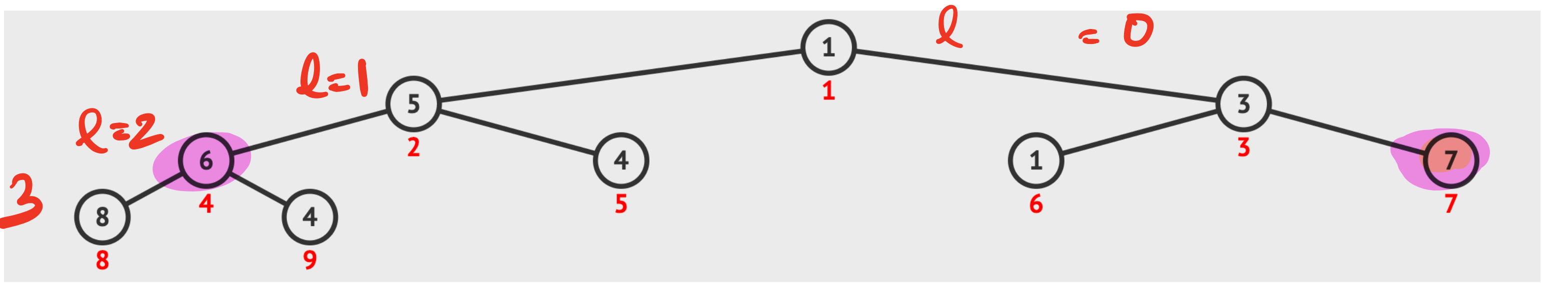
$\leq 2^h \, 2 + 2 \cdot 2^h$

$\leq 4 \cdot n.$ Since: $2^h < n$ (No. of nodes at the last level is less than the total no. of nodes)

$= O(n)$

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

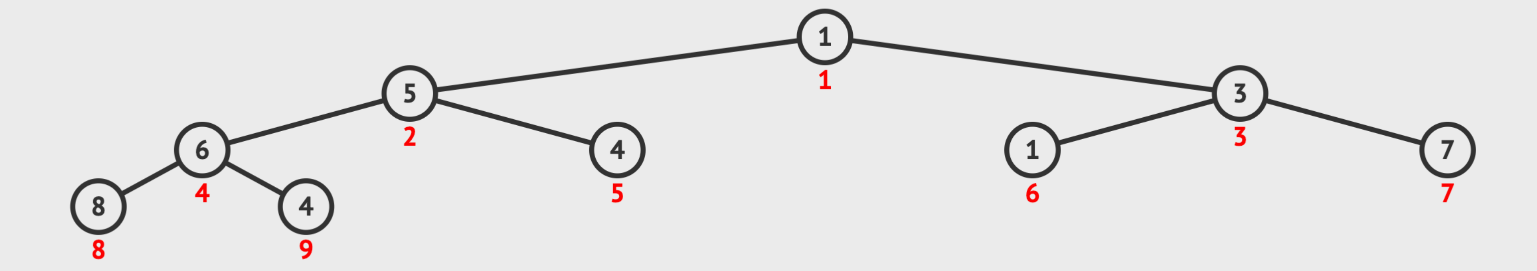
$h = 3$

# of node from 0 to l

when every level is complete

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

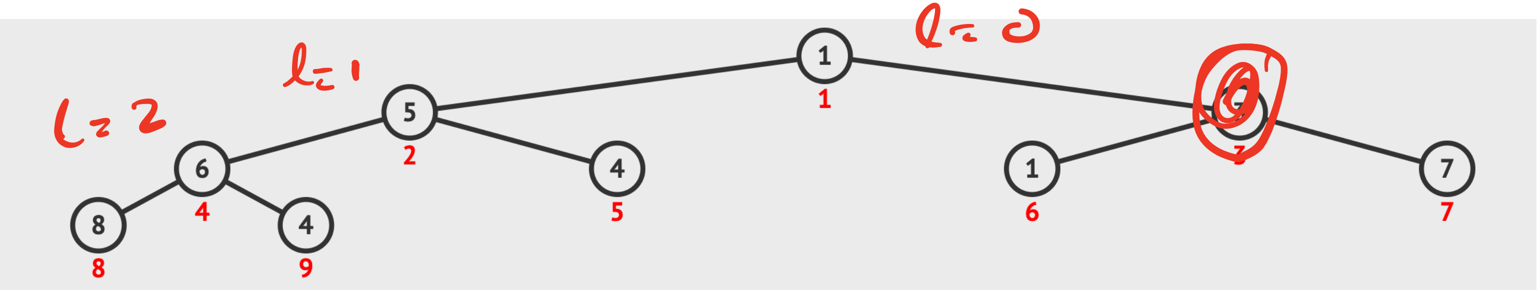
D. $2^{h + 1} - 2$

E. Something else

This is just one less than the total no. of nodes from level 0 to ℓ (which from the last slide is $2^{\ell+1} - 1$)

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

Assume $\ell <= h - 1$



A. 2^{ℓ}

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

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

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

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

$$= 2^{\ell} - 1$$

We used this result to code heapify another way which corresponds more directly to how we analyzed the run time.