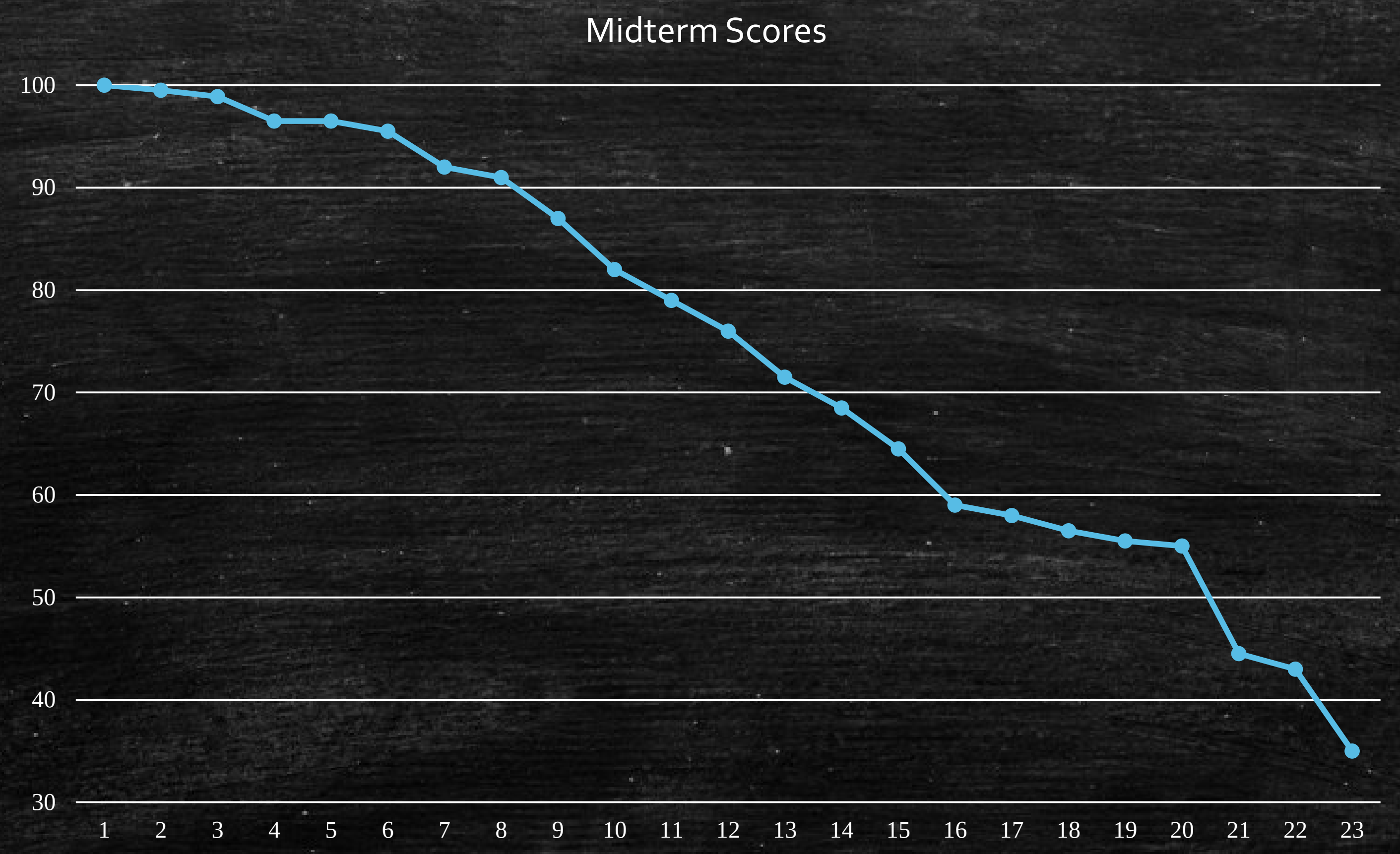
# CS 457, Fall 2019

Drexel University, Department of Computer Science

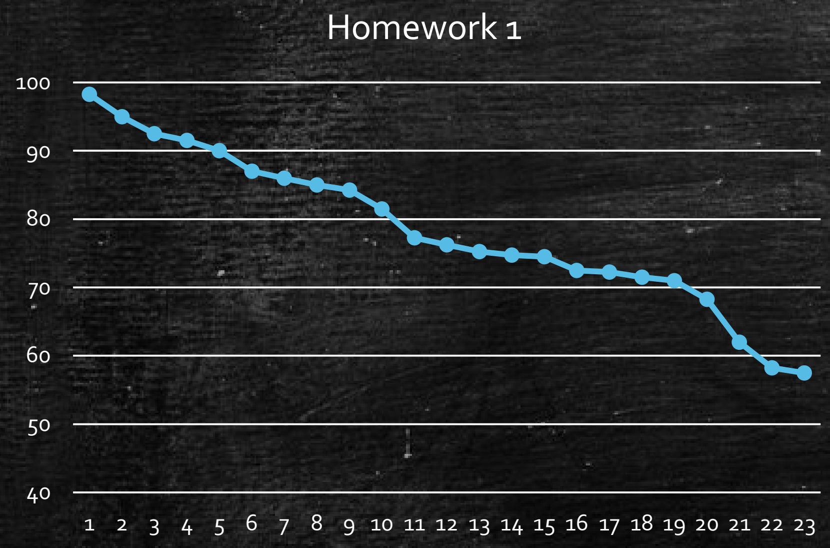
Lecture 11

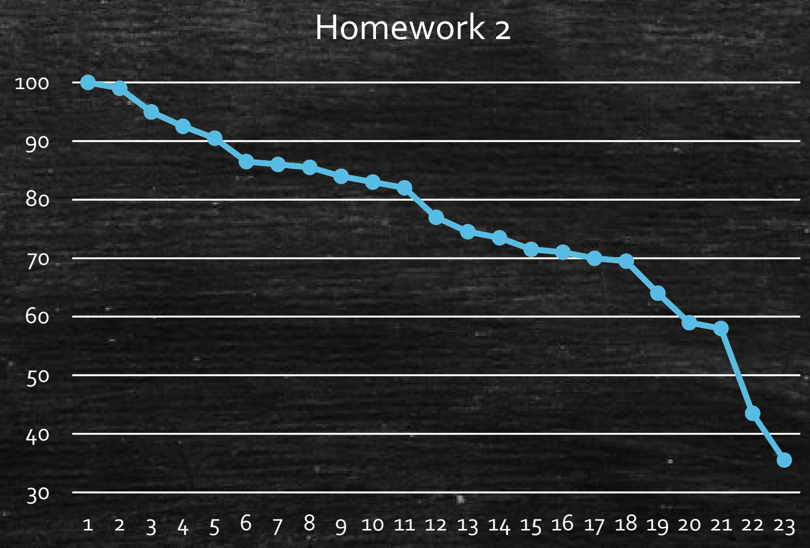
# Midterm results

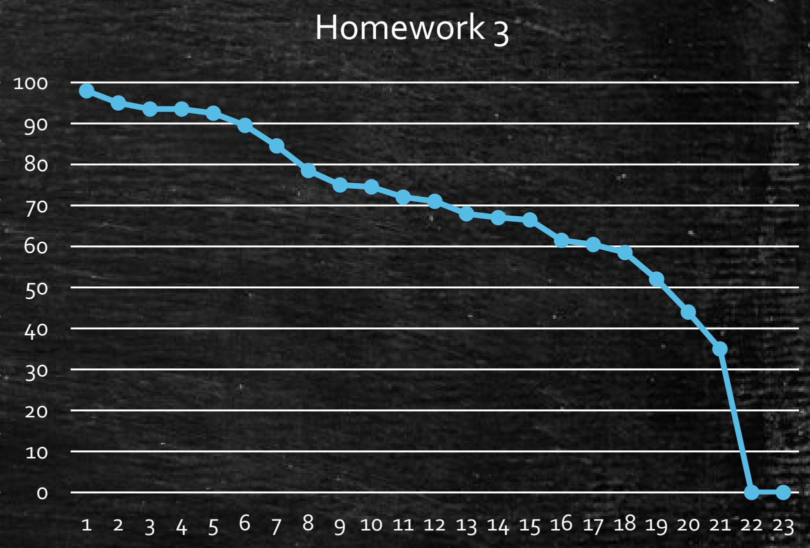
- Midterm results:



# Homework results



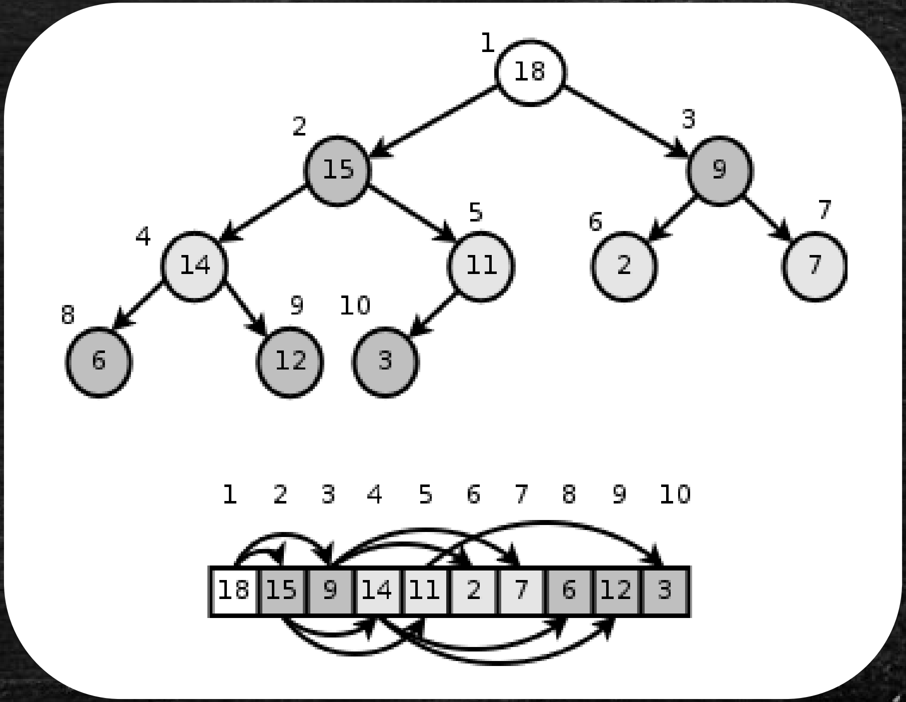



# Binary Heaps

What is a binary heap?

Heap data structure:

PARENT ($i$)
    return $\lfloor i/2 \rfloor$
LEFT ($i$)
    return $2i$
RIGHT ($i$)
    return $2i + 1$




Max-heap property: A[PARENT($i$)] $\geq$ A[$i$]

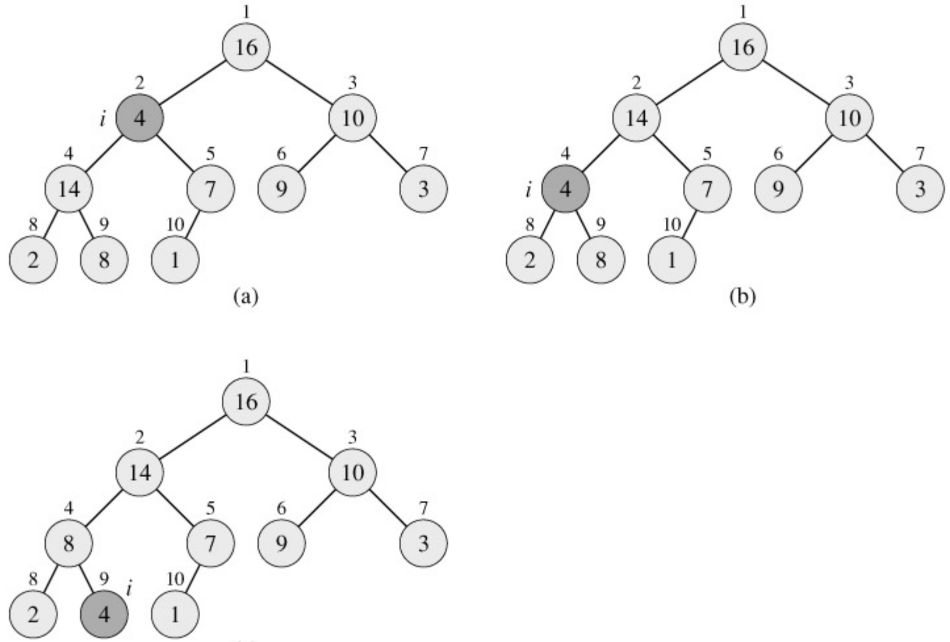
# Max-Heapify

Running time is $O(h)$, where $h$ is height of node $i$

**Max-Heapify (A, $i$ )**

```
l = left(i)
r = right(i)
if l ≤ A.heap-size and A[l] > A[i]
        largest = l
else largest = i
if r ≤ A.heap-size and A[r] > A[largest]
        largest = r
if largest ≠ i
        exchange A[i] with A[largest]
        Max-Heapify (A, largest)
```

Call to **Max-Heapify (A,2)**



(a) (b) (c)

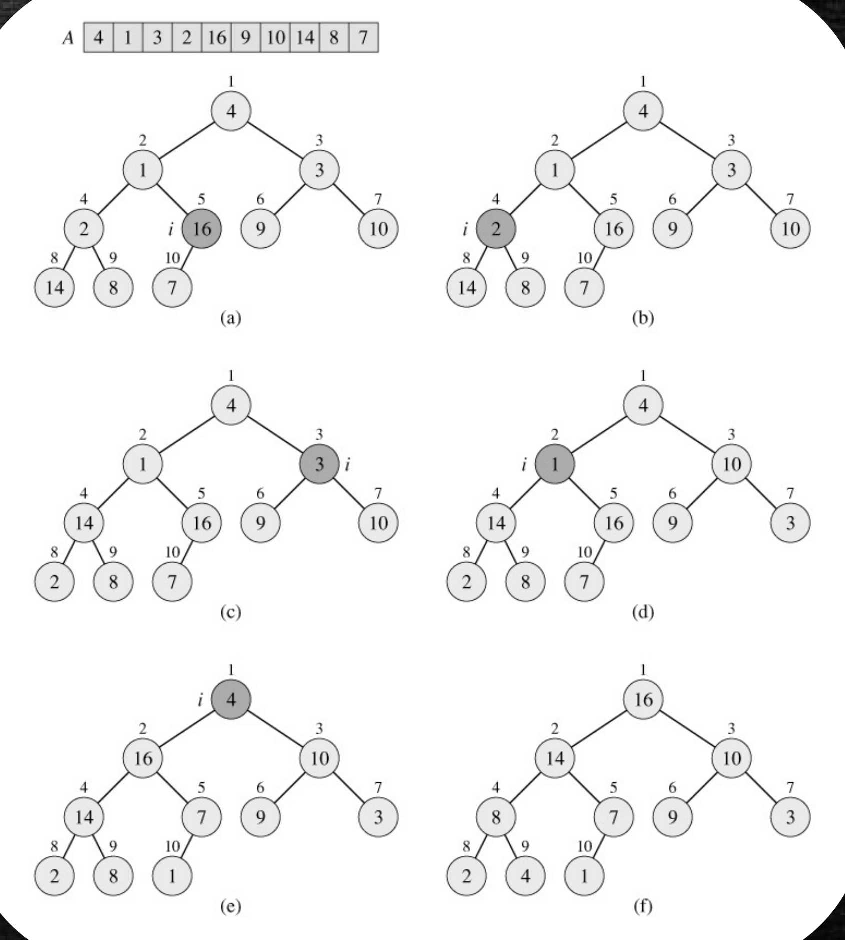
# Build-Max-Heap

**Build-Max-Heap (A)**

1. A.heap-size = A.length
2. **for** $i = \lfloor \text{A.length}/2 \rfloor$ **down to** 1
3. Max-Heapify(A, $i$)

What is the running time of Build-Max-Heap?

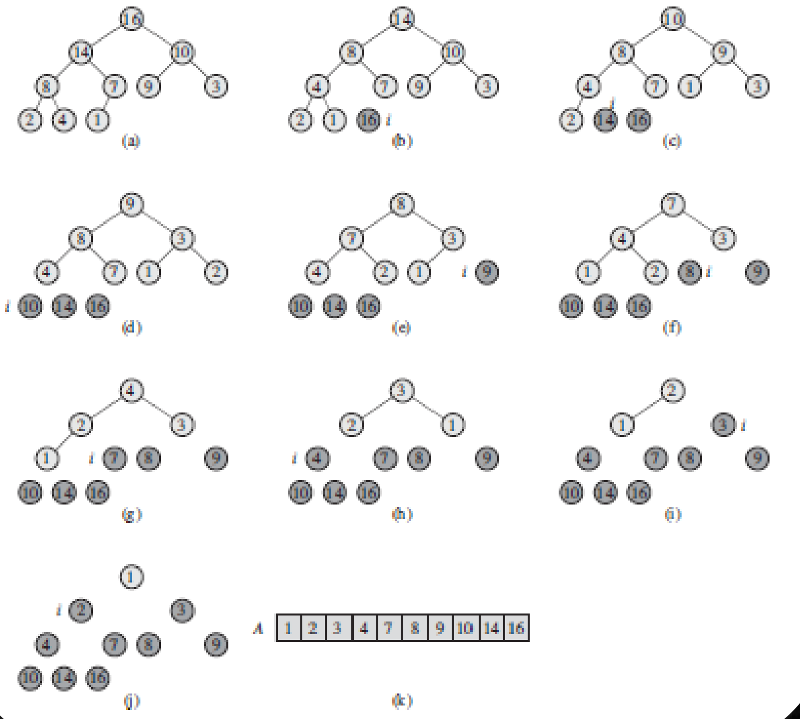
$$T(n) = \sum_{h=0}^{\lfloor \log n \rfloor} \left\lceil \frac{n}{2^{h+1}} \right\rceil O(h) = O\left( n \sum_{h=0}^{\lfloor \log n \rfloor} \frac{h}{2^h} \right) = O(n)$$



# Heapsort

**Heapsort(A)**

```
Build-Max-Heap(A)
for i = A.length down to 2
        exchange A[1] with A[i]
        A.heap-size = A.heap-size - 1
        Max-Heapify(A,1)
```



# Today’s Lecture

- Binary trees (this is a good time to read through Appendix B!)
  - Structural induction
  - Search trees

# Binary Trees and Induction

- Binary trees play a major role in computer science
  - E.g., heaps, binary search trees, red-black trees
  - It is important to have good intuition regarding their properties
- What is a complete (rooted) binary tree?
  - A binary tree in which all leaves have the same depth
  - How many leaves does a complete binary tree of height $h$ have?
  - Any complete binary tree of height $h$ has $2^h$ leaves. Can you prove this using induction?
  - How many nodes does a complete binary tree of height $h$ have?
  - Any complete binary tree of height $h$ has $2^{h+1}-1$ nodes. Can you prove this using induction?
- What is a full (rooted) binary tree?
  - A binary tree in which every node is either a leaf or has degree* exactly 2
  - How many leaves does a full binary tree with $n$ nodes have?
  - Show that there are at most $\lceil n/2^{h+1} \rceil$ nodes of height $h$ in any $n$-node full binary tree

# Structural Induction Problem

**Question:** A rooted binary tree is a rooted tree in which each node has at most two children. A node of a tree is full if it contains a non-empty left child and a non-empty right child. Prove, using induction, that for any rooted binary tree, the number of full nodes is one less than the number of leaves.

**Solution:** For the base case, we consider the set of trees with 1 node. Clearly, there exists only one such tree that comprises of a single node and no edges. For this tree, the number of leaves is 1 and the number of full nodes is 0, so the number of full nodes is indeed exactly one less than the number of leaves in this case.