

Chapter 3, Case Studies:

- ▶ Linear-time sorting algorithms (last week Wednesday and Friday)
- ▶ Disjoint sets and array forests (Monday)
- ▶ Priority queues (**Wednesday and Friday**)
- ▶ N -sets and bit vectors (Thursday in lab)
- ▶ (Start graphs next week Monday)

Today:

- ▶ Recent HW and quiz problems
- ▶ Worklist algorithms
- ▶ Priority queue ADT (problem statement)
- ▶ Inefficient solutions
- ▶ Abstractions for the heap data structure
- ▶ Heap implementation details, part 1
- ▶ Excursion: heap sort
- ▶ Heap implementation details, part 2
- ▶ Analysis and optimization

```
public class ArrayList<E>
    implements List<E> {
    private E[] internal;
    private int size;
    ...
}
```

```
public class ArrayStack<E>
    implements Stack<E> {
    private E[] internal;
    private int size;
    ...
}
```

Informal: The positions of `internal` in range $[0, \text{size})$ contain the elements in the conceptual list in order.

Formal:

- (a) $\text{size} \leq \text{internal.length}$
- (b) `size` equals the number of calls to `insert()` and `add()` minus the number of (non-empty) calls to `remove()`.

Informal: `size` is the number of items in the conceptual stack, which appear in order of arrival in range $[0, \text{size})$

Formal:

- (a) $\text{size} \leq \text{internal.length}$
- (b) `size` equals the number of calls to `push()` minus the number of (non-empty) calls to `pop()`.
- (c) For all $i \in [0, \text{size} - 1)$, the item in `internal[i]` was pushed before the item in `internal[i + 1]`.

```

private class PlainFind implements FindStrategy {
    public int find(int p) {
        while (p != parents[p]) p = parents[p];
        return p;
    }
}

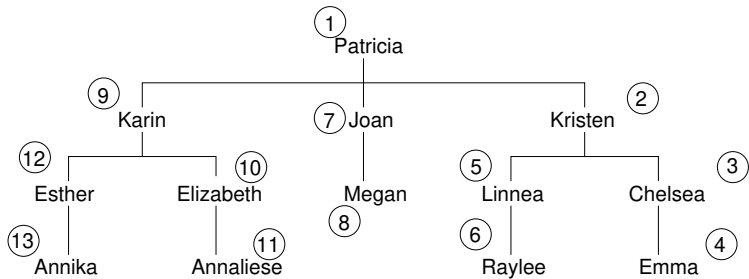
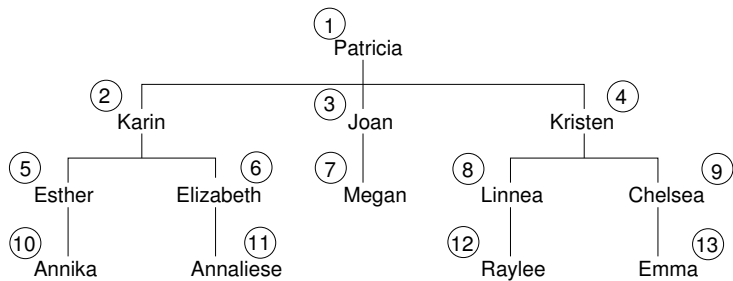
```

Let p_{orig} be the original value of p . Let i be the number of iterations completed.

- (a) $0 \leq p < N$.
- (b) $\text{leader}(p) = \text{leader}(p_{\text{orig}})$
- (c) $p = \overbrace{\text{parents}[p_{\text{orig}}]}^{i \text{ times}}$

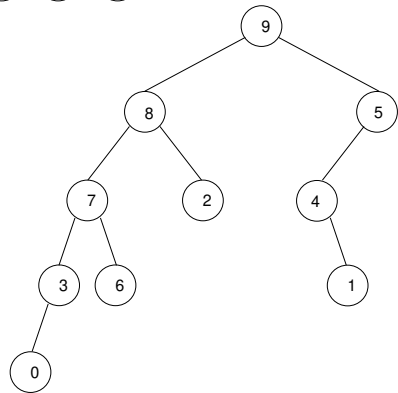
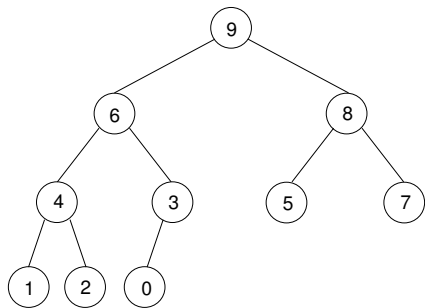
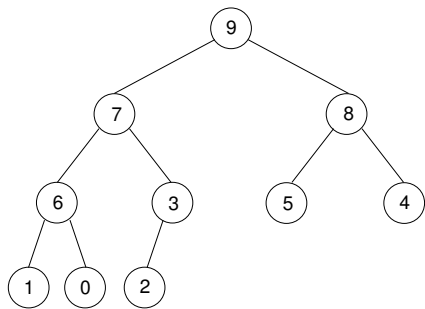
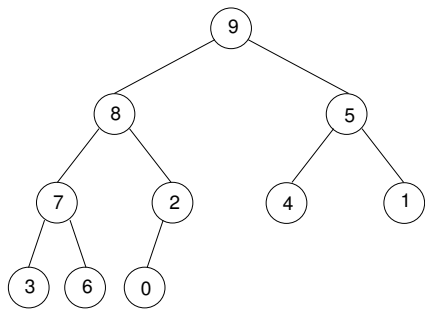
- ▶ An array forest is an example of what kind of data structure?
It contains both array-based and linked concepts.
- ▶ For each, is it best considered a data structure, abstraction, or abstract data type?

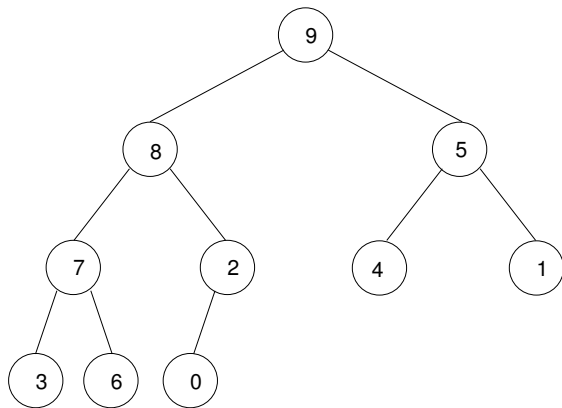
Array	Data structure (simple)
Array Forest	Data structures (hybrid/advanced)
Disjoint set	Abstract data type
Forest	Abstraction
- ▶ What design pattern was (explicitly) used in the implementation of disjoint sets?
Strategy



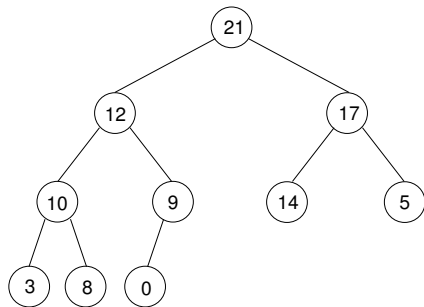
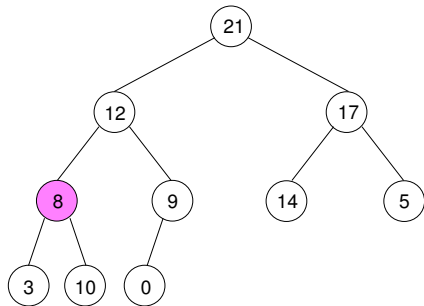
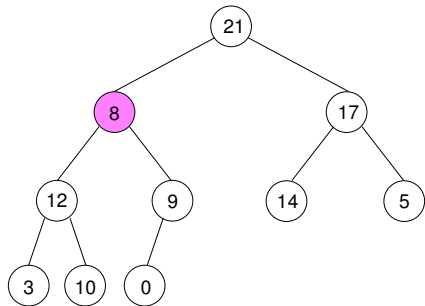
	ListPriorityQueue	SortedListPriorityQueue
--	-------------------	-------------------------

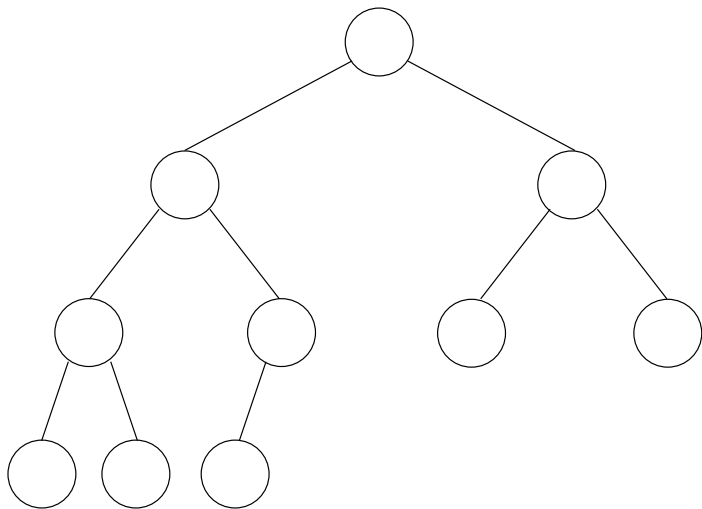
Initialize empty	$\Theta(1)$	$\Theta(1)$
Initialize populated	$\Theta(n)$	$\Theta(n^2)$
insert	$\Theta(1)$	$\Theta(n)$
max	$\Theta(n)$	$\Theta(1)$
extractMax	$\Theta(n)$	$\Theta(1)$
contains	$\Theta(n)$	$\Theta(n)$
increaseKey	$\Theta(1)$	$\Theta(n)$
decreaseKey	$\Theta(1)$	$\Theta(n)$





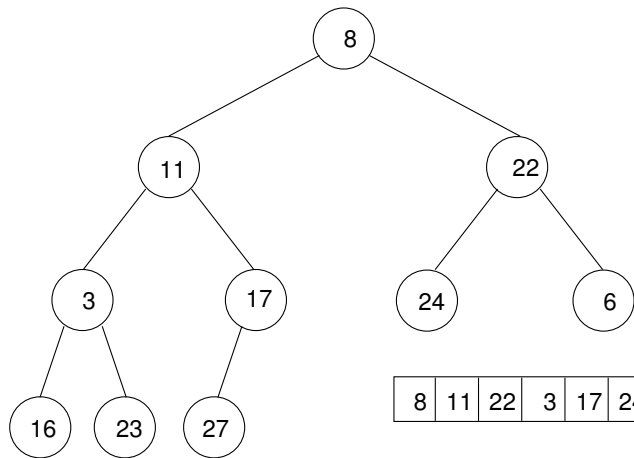
9	8	5	7	2	4	1	3	6	0
---	---	---	---	---	---	---	---	---	---



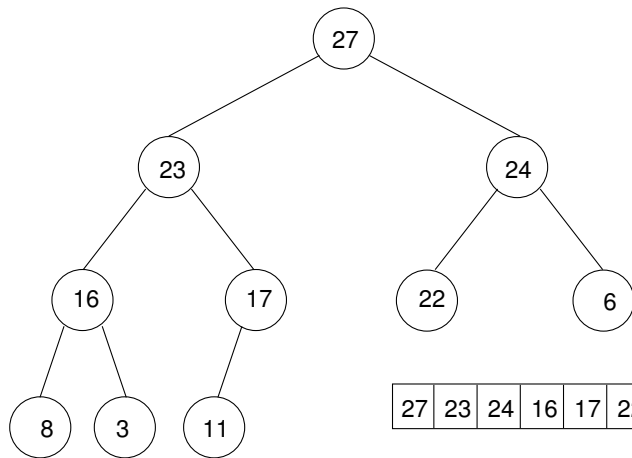


--	--	--	--	--	--	--	--	--	--

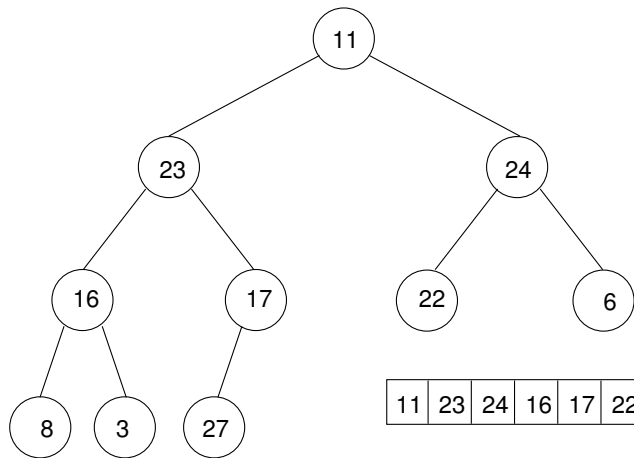
$$\begin{aligned}
\sum_{i=0}^{h-1} 2^i (h-1-i) &= (h-1) \sum_{i=0}^{h-1} 2^i - \sum_{i=0}^{h-1} i 2^i \\
&= (h-1)(2^h - 1) - 2 - (h-2)2^h \\
&= h2^h - h - 2^h + 1 - 2 - h2^h + 2 \cdot 2^h \\
&= 2^h - h - 1 \\
&= 2^{\lg(n+1)} - \lg(n+1) - 1 \\
&= n+1 - \lg(n+1) - 1 \\
&= n - \lg(n+1)
\end{aligned}$$



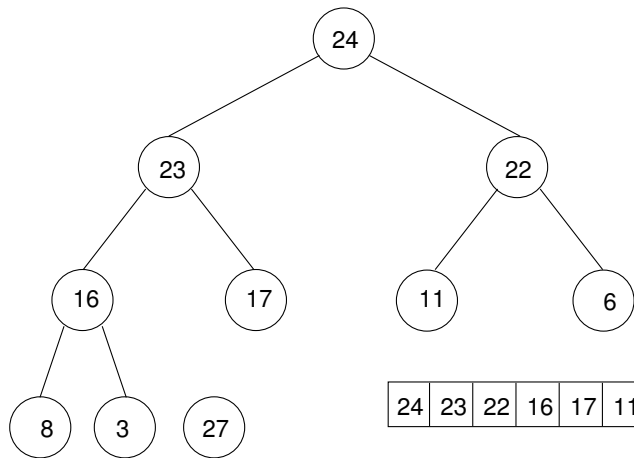
8	11	22	3	17	24	6	16	23	27
---	----	----	---	----	----	---	----	----	----



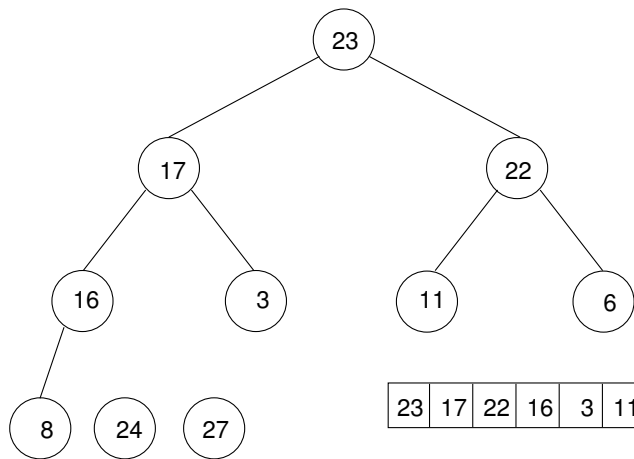
27	23	24	16	17	22	6	8	3	11
----	----	----	----	----	----	---	---	---	----

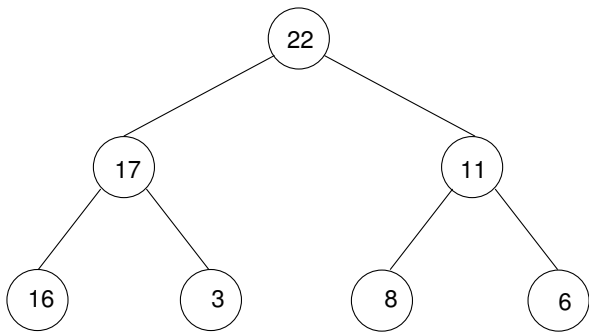


11	23	24	16	17	22	6	8	3	27
----	----	----	----	----	----	---	---	---	----

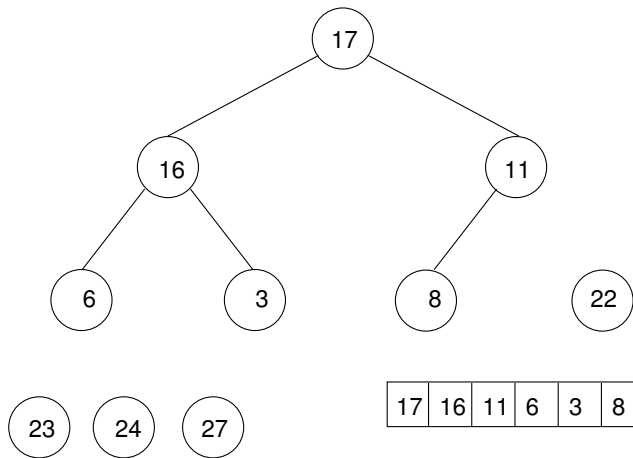


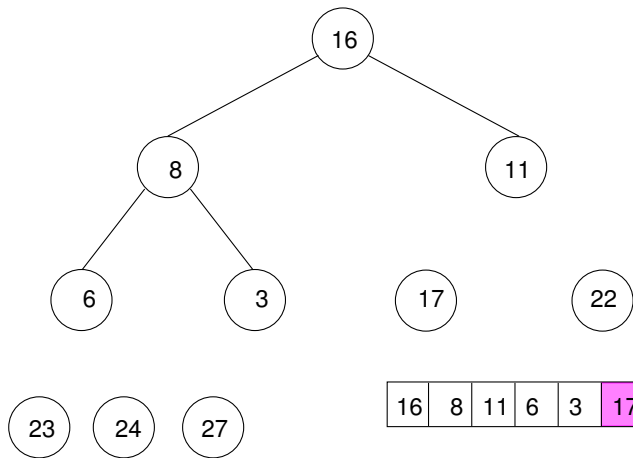
24	23	22	16	17	11	6	8	3	27
----	----	----	----	----	----	---	---	---	----



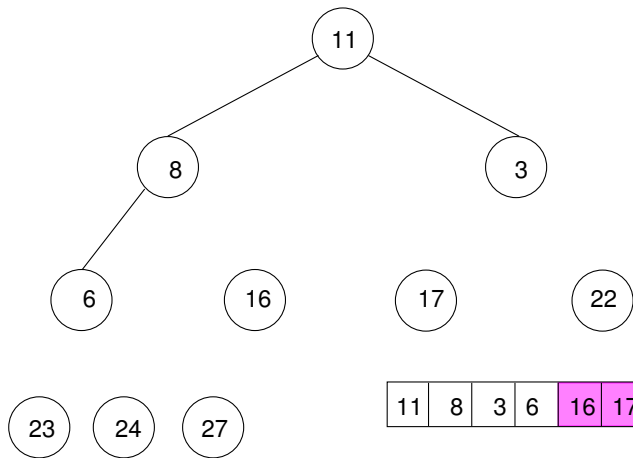


22	17	11	16	3	8	6	23	24	27
----	----	----	----	---	---	---	----	----	----

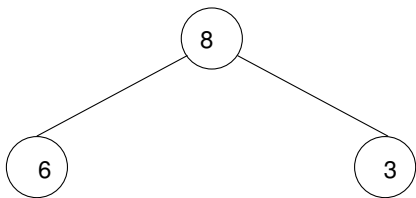




16	8	11	6	3	17	22	23	24	27
----	---	----	---	---	----	----	----	----	----



11	8	3	6	16	17	22	23	24	27
----	---	---	---	----	----	----	----	----	----



11

16

17

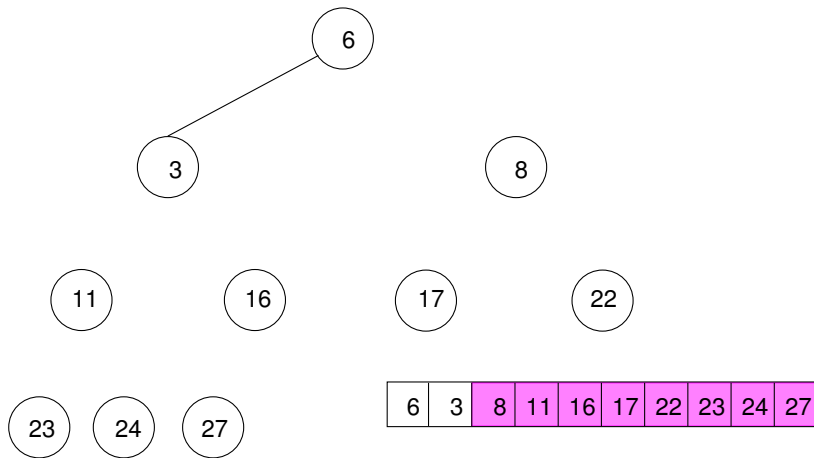
22

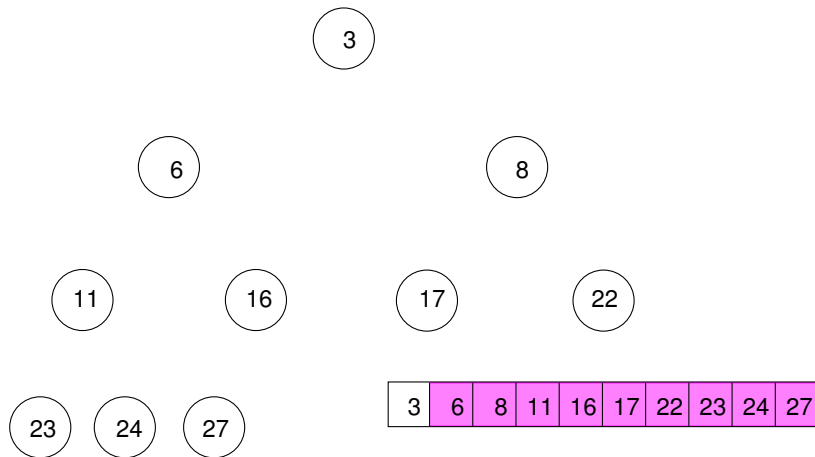
23

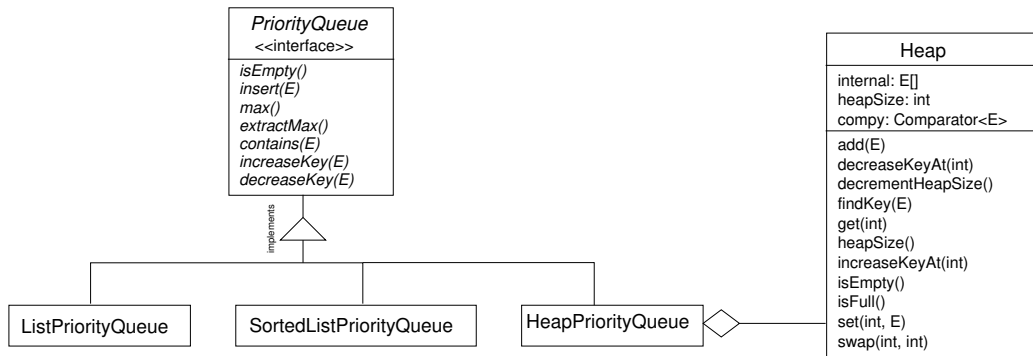
24

27

8	6	3	11	16	17	22	23	24	27
---	---	---	----	----	----	----	----	----	----







	ListPriorityQueue	SortedPriorityQueue	HeapPriorityQueue
Initialize empty	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$
Initialize populated	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n)$
insert	$\Theta(1)$	$\Theta(n)$	$\Theta(\lg n)$
max	$\Theta(n)$	$\Theta(1)$	$\Theta(1)$
extractMax	$\Theta(n)$	$\Theta(1)$	$\Theta(\lg n)$
contains	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$
increaseKey	$\Theta(1)$	$\Theta(n)$	$\Theta(n)$
decreaseKey	$\Theta(1)$	$\Theta(n)$	$\Theta(n)$

Coming up: (all end-of-day)

*Do **linear sorting** project (due today, Wed, Sept 24)*

*Do **heaps and priority queue** project (Fri, Oct 3)*

*Due **Thurs, Sept 25:***

Read Section 3.4 (N-sets and bit vectors)

Do Exercises 3.(26 & 27).

Take N-sets quiz

*Due **Fri, Sept 26:***

Read Section 3.3 (heaps and priority queues)

(no exercises)

Take heap/pq quiz