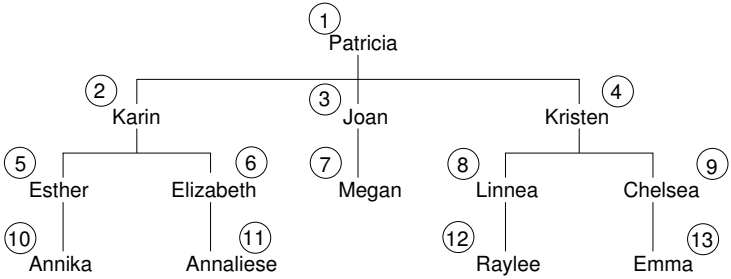
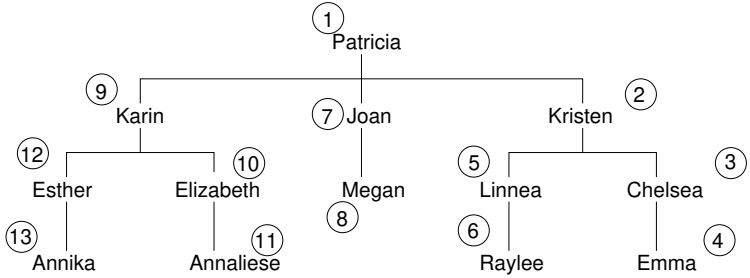
Chapter 3, Case Studies:

- Linear-time sorting algorithms (last week Wednesday and Friday)
- Disjoint sets and array forests (Monday)
- Priority queues (**Today**)
- $N$-sets and bit vectors (Friday )
- (Start graphs Monday)

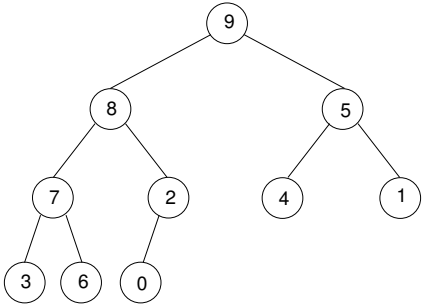
Today:

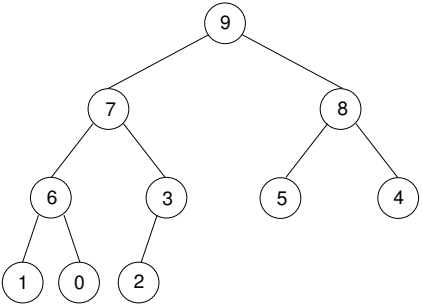
- Worklist algorithms
- Priority queue ADT (problem statement)
- Inefficient solutions
- Abstractions for the heap data structure
- Heap implementation details, part 1
- Excursion: heap sort
- Heap implementation details, part 2
- Analysis and optimization

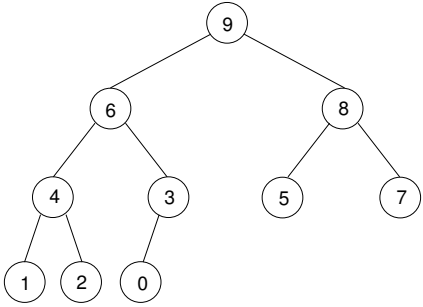

1
Patricia
2
Karin
3
Joan
4
Kristen
5
Esther
6
Elizabeth
7
Megan
8
Linnea
9
Chelsea
10
Annika
11
Annaliese
12
Raylee
13
Emma

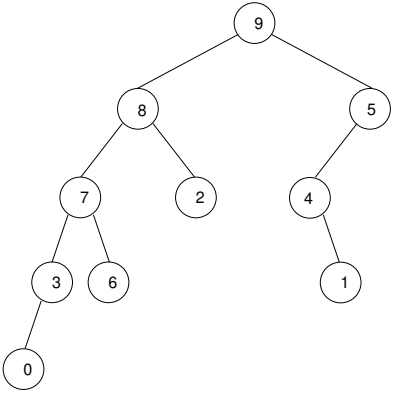

1
Patricia
9
Karin
7
Joan
2
Kristen
12
Esther
10
Elizabeth
Megan
8
5
Linnea
3
Chelsea
13
Annika
11
Annaliese
6
Raylee
4
Emma

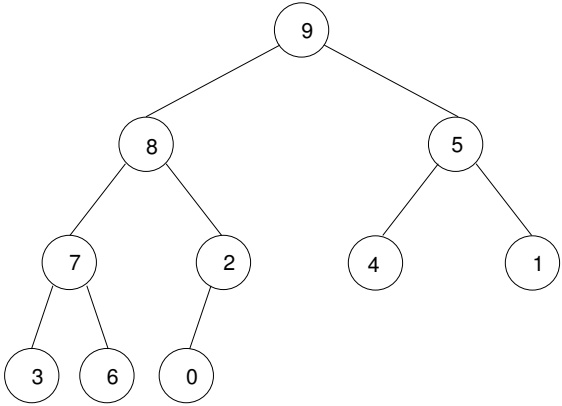
| | `ListPriorityQueue` | `SortedListPriorityQueue` |
|---|---|---|
| Initialize empty | $\Theta(1)$ | $\Theta(1)$ |
| Initialize populated | $\Theta(n)$ | $\Theta(n^2)$ |
| `insert` | $\Theta(1)$ | $\Theta(n)$ |
| `max` | $\Theta(n)$ | $\Theta(1)$ |
| `extractMax` | $\Theta(n)$ | $\Theta(1)$ |
| `contains` | $\Theta(n)$ | $\Theta(n)$ |
| `increaseKey` | $\Theta(1)$ | $\Theta(n)$ |
| `decreaseKey` | $\Theta(1)$ | $\Theta(n)$ |


9
8
5
7
2
4
1
3
6
0

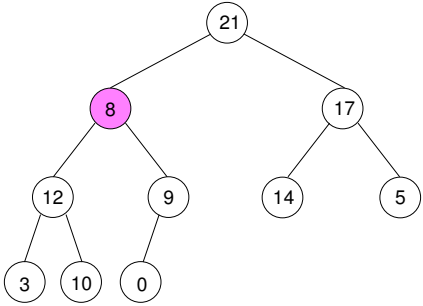

9
7
8
6
3
5
4
1
0
2

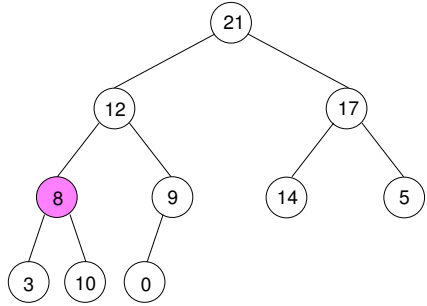

9
6
8
4
3
5
7
1
2
0

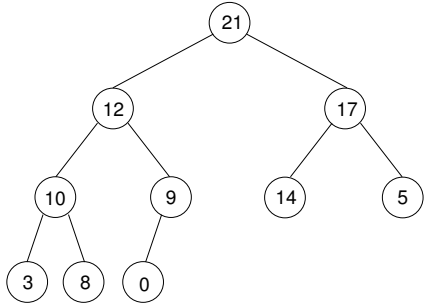

9
8
5
7
2
4
3
6
1
0

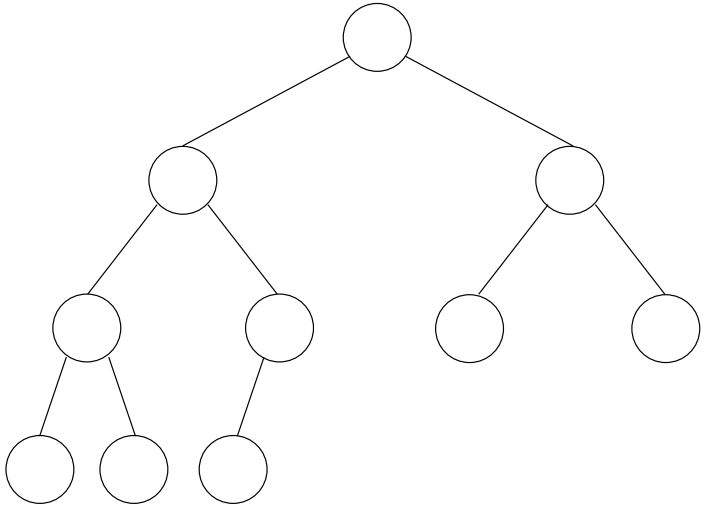


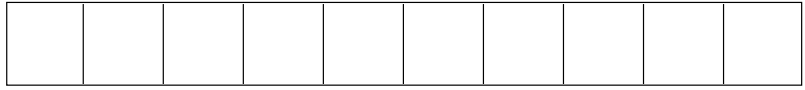

| 9 | 8 | 5 | 7 | 2 | 4 | 1 | 3 | 6 | 0 |
|---|---|---|---|---|---|---|---|---|---|


21
8
17
12
9
14
5
3
10
0

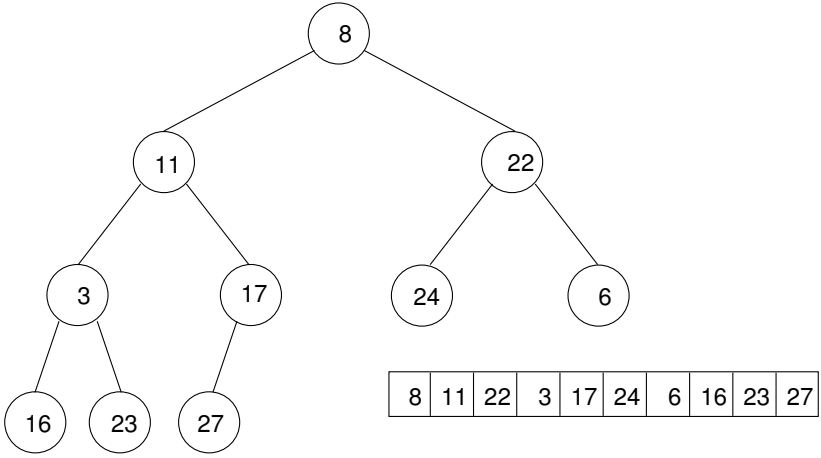

21
12
17
8
9
14
5
3
10
0

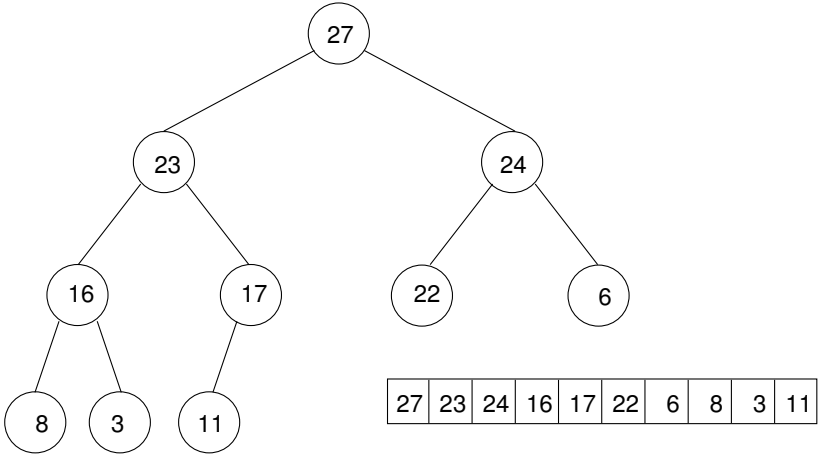

21
12
17
10
9
14
5
3
8
0



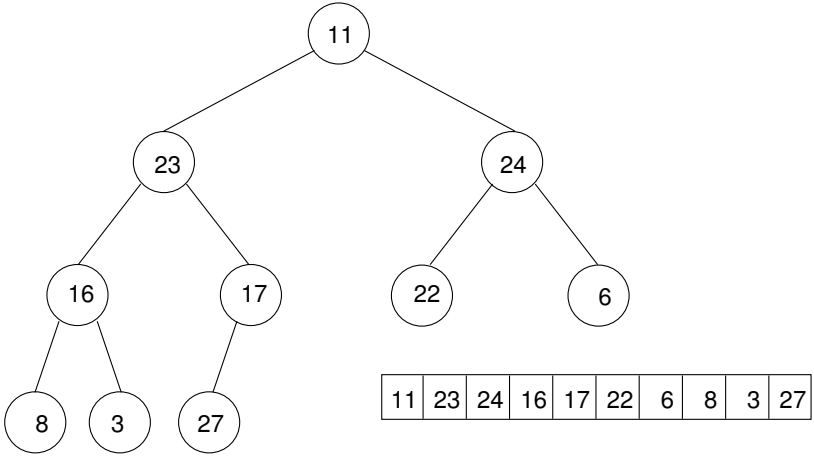


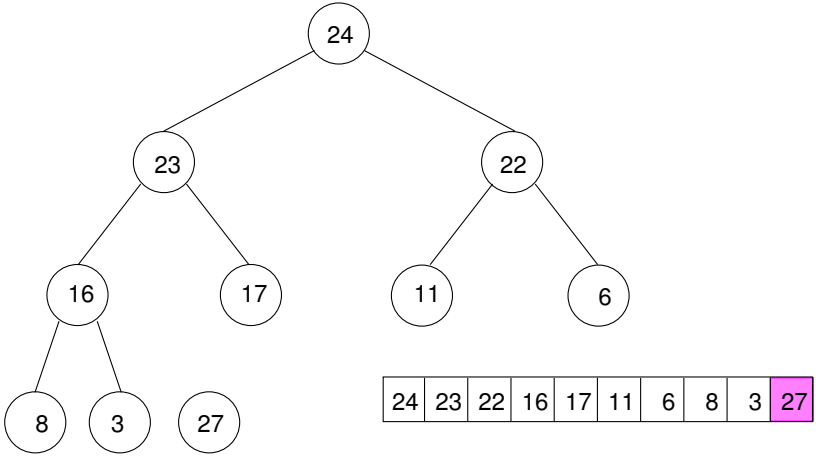
$$
\begin{aligned}
\sum_{i=0}^{h-1} 2^i(h-1-i) &= (h-1)\sum_{i=0}^{h-1} 2^i \;-\; \sum_{i=0}^{h-1} i2^i \\
&= (h-1)(2^h-1) - 2 - (h-2)2^h \\
&= h2^h - 2^h - h + 1 - 2 - h2^h + 2\cdot 2^h \\
&= 2^h - h - 1 \\
&= 2^{\lg(n+1)} - \lg(n+1) - 1 \\
&= n + 1 - \lg(n+1) - 1 \\
&= n - \lg(n+1)
\end{aligned}
$$

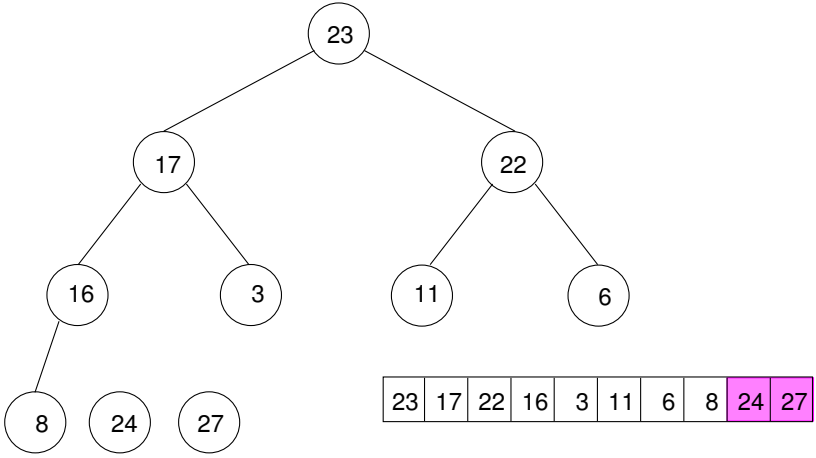

8
11
22
3
17
24
6
16
23
27
8 11 22 3 17 24 6 16 23 27

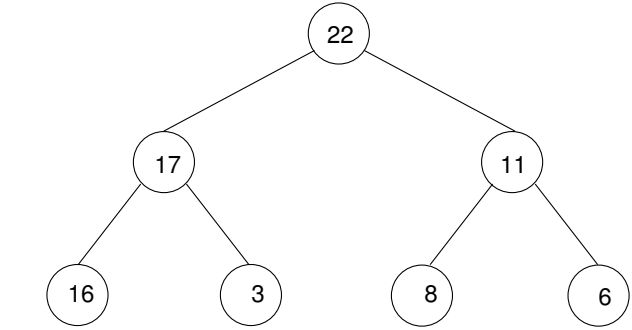



| 27 | 23 | 24 | 16 | 17 | 22 | 6 | 8 | 3 | 11 |
|---|---|---|---|---|---|---|---|---|---|

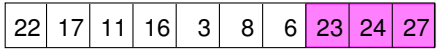

11
23
24
16
17
22
6
8
3
27
11 23 24 16 17 22 6 8 3 27

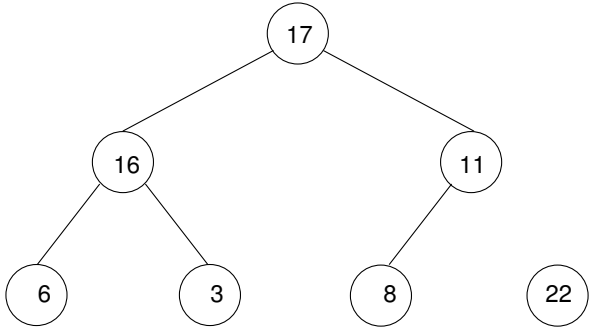

24
23
22
16
17
11
6
8
3
27
24 23 22 16 17 11 6 8 3 27


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

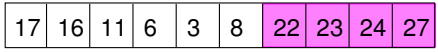

22
17
11
16
3
8
6

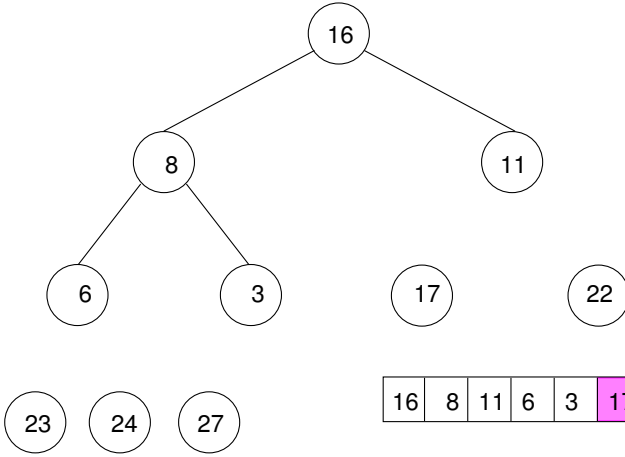

23
24
27


22
17
11
16
3
8
6
23
24
27

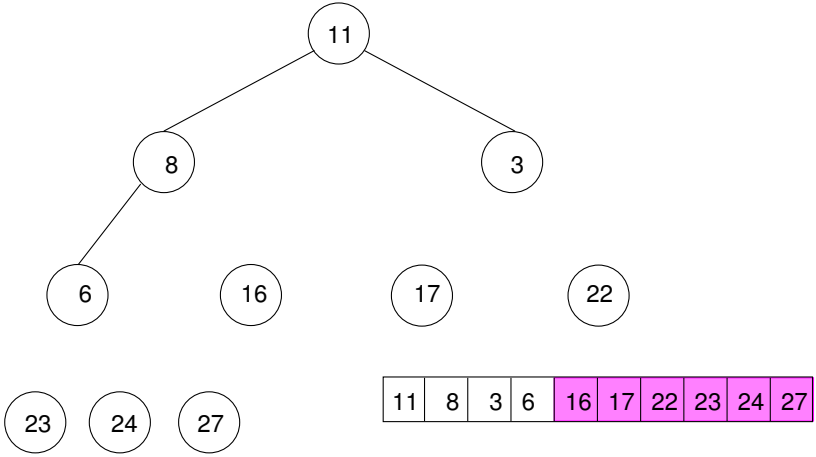

17
16
11
6
3
8
22

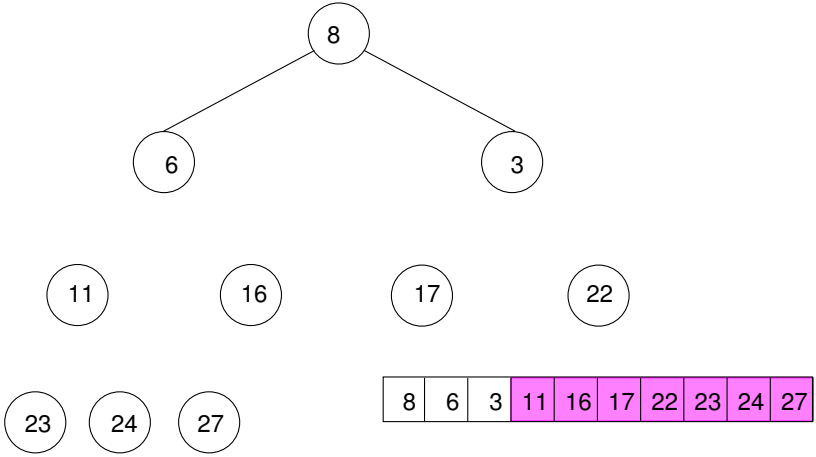

23
24
27

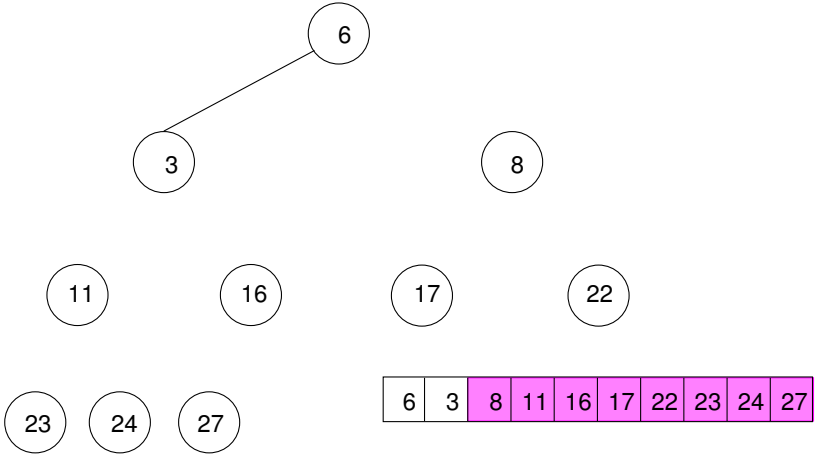

17
16
11
6
3
8
22
23
24
27

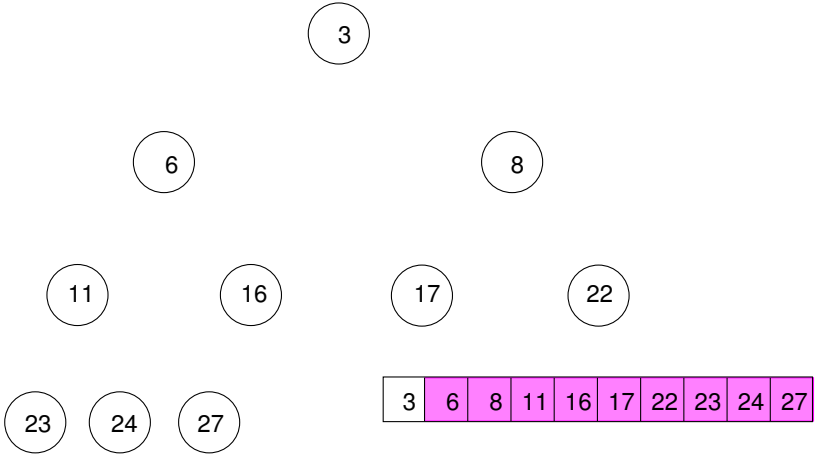


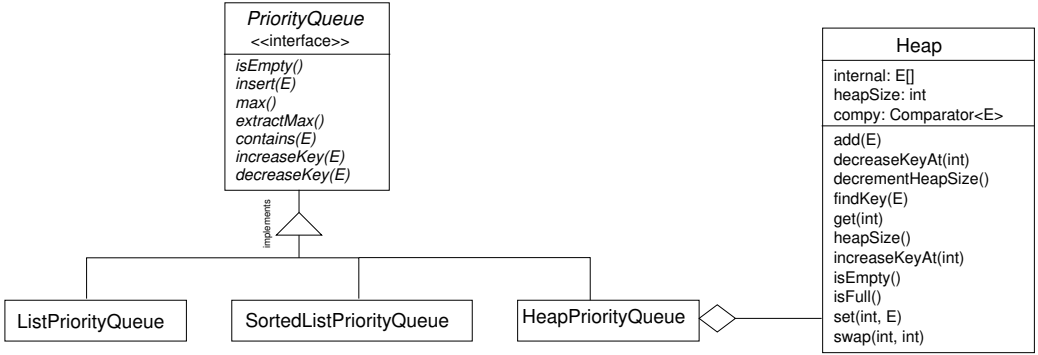

| 16 | 8 | 11 | 6 | 3 | 17 | 22 | 23 | 24 | 27 |
|---|---|---|---|---|---|---|---|---|---|


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


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


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


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


PriorityQueue
<<interface>>
isEmpty()
insert(E)
max()
extractMax()
contains(E)
increaseKey(E)
decreaseKey(E)
implements
Heap
internal: E[]
heapSize: int
compy: Comparator<E>
add(E)
decreaseKeyAt(int)
decrementHeapSize()
findKey(E)
get(int)
heapSize()
increaseKeyAt(int)
isEmpty()
isFull()
set(int, E)
swap(int, int)
ListPriorityQueue
SortedListPriorityQueue
HeapPriorityQueue

| | `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 (suggested by today)*
*Do* **heaps and priority queue** *project (suggested by Wed, Sept 28)*

*Due* **Today***:*
*Read Section 3.3 (heaps and priority queues)*

*Due* **Thursday***:*
*Take heap/pq quiz*

*Due* **Friday***:*
*Read Section 3.4*
*Do Exercises 3.(26 & 27).*
*Take N-sets quiz*