Naive Solution:
- Store items in linear list
- Order?

Insert order - fast insert/slow extract
Priority order - fast extract/slow insert

Heap: Tree-based structure

(min) heap order: for all nodes, parent's key ≤ node's key

[Reverse: max-heap order]

Many variants:
- Binary, leftist, binomial, Fibonacci, pairing, quake, skew... heaps
- Decrease key: O(log n) meld or merge: O(1)

Binary Heap:
- Simple, elegant, efficient
- Old (1964): J.W.J. Williams
- Basic: insert/extract: O(log n)
  build: O(n)

Priority Queue:
- Stores key-value pairs
- Key = priority
- Ops: insert(x, x) - insert value v with key x
  extract-min - remove/return pair with min key value

Priority Queues + Heaps I

Heap: Tree-based structure

[min] heap order: for all nodes, parent's key ≤ node's key

Many variants:
- Binary, leftist, binomial, Fibonacci, pairing, quake, skew... heaps
- Decrease key: O(log n) meld or merge: O(1)

Binary Heap:
- Simple, elegant, efficient
- Old (1964): J.W.J. Williams
- Basic: insert/extract: O(log n)
  build: O(n)

Priority Queue:
- Stores key-value pairs
- Key = priority
- Ops: insert(x, x) - insert value v with key x
  extract-min - remove/return pair with min key value

Heap: Tree-based structure

[min] heap order: for all nodes, parent's key ≤ node's key

Many variants:
- Binary, leftist, binomial, Fibonacci, pairing, quake, skew... heaps
- Decrease key: O(log n) meld or merge: O(1)

Binary Heap:
- Simple, elegant, efficient
- Old (1964): J.W.J. Williams
- Basic: insert/extract: O(log n)
  build: O(n)

Priority Queue:
- Stores key-value pairs
- Key = priority
- Ops: insert(x, x) - insert value v with key x
  extract-min - remove/return pair with min key value

Heap: Tree-based structure

[min] heap order: for all nodes, parent's key ≤ node's key

Many variants:
- Binary, leftist, binomial, Fibonacci, pairing, quake, skew... heaps
- Decrease key: O(log n) meld or merge: O(1)

Binary Heap:
- Simple, elegant, efficient
- Old (1964): J.W.J. Williams
- Basic: insert/extract: O(log n)
  build: O(n)
Example:  

```
6
/   \\
7   6
/ \  /\
3  9  9
```

**Binary Heap - Extract Min**  
- Min key at root → save it  
- Copy A[n] to root (A[i]) + decrement n  
- Sift the root key down  
  - if larger, swap with child  
  - Return saved root key

**Leftist Property:**  
Null path length:

\[
npl(v) = \begin{cases} 
-1 & v = \text{null} \\
1 + \min(npl(v.\text{left}), npl(v.\text{right})) & \text{otherwise} 
\end{cases}
\]

**Def:** Leftist Heap is binary tree where:

- Keys are heap ordered  
- A nodes v, npl(v.left) ≥ npl(v.right)

**Priority Queues & Heaps II**

```
int sift-down(int i, Key z)
while (i ≠ null)
    [buggy] u ← left(i), v ← right(i)  
    if (v ≤ n && A[v] < A[u])  
       [buggy] u ← v  
    else break  
return i
```

**Analysis:** Both insert & extract-min take time proportional to tree height  
Tree is complete ⇒ \(O(\log n)\) time
**Class structure:**
```
Leftist Heap <Key> { }
```

**Private class LHNode:**
```
private class LHNode {
  Key x
  LHNode left, right
}
```

**Private LHNode root:**
```
private LHNode root
```

**Public LeftistHeap:**
```
public LeftistHeap() { root = null }
```

**Void insert(Key x):**
```
  void insert(Key x)
```

**Void extractMin():**
```
  void extractMin()
```

**Void mergeWith(LeftistHeap H2):**
```
  void mergeWith(LeftistHeap H2)
```

**Public mergeWith(LeftistHeap H2):**
```
public mergeWith(LeftistHeap H2)
  root ← merge(this.root, H2.root)
  H2.root ← null
```

**Merge helper:** 2 phases
1. Merge right paths by order of keys + update npl's
2. Check leftist property + swap

**Lemma:** A leftist tree with \( r \geq 1 \) nodes along its rightmost path has \( n \geq 2^r - 1 \) nodes

**Proof:** (Sketch - see latex notes)
```
```

**Analysis:** Time \( n \) Rightmost path = \( O(\log n) \)

**Insert + Extract-min?** Exercises

**LHNode merge(LHNode u, LHNode v):**
```
if (u = null) return v
if (v = null) return u
if (u.key > v.key)
  swap u ← v
if (u.left = null) u.left ← v
else
  u.right ← merge(u.right, v)
```

**Priority Queues + Heaps III**

**Phase 1:**
```
merge
```

**Phase 2:**
```
merge
```

**Final tree!**
Announcements - Tue 9/13

- Just added? Check with me ... Piazza, Gradescope, recordings
- Prog Assign 0 due tonight 11:59 pm
- Homework 1??
- Fast-forward recordings
  for Union-Find & Heaps??
- My office hours?? soon