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

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

Priority Queue:
- Stores key-value pairs
- Key = priority
- Ops:
  - insert(x, v) - 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

[Reverse: max-heap order]

Many variants:
*binary, leftist, binomial, fibonacci, pairing, quake, skew... heaps

Binary Heap:
- simple, elegant, efficient
- old (1964) - williams - heapsort
- basic: insert/extract O(log n)
  build - O(n) heapify

Priority Queues + Heaps I

Heap

Binary Heap:
- Simple, elegant, efficient
- Old (1964) - Williams - Heapsort
- Basic: insert/extract O(log n)
  Build - O(n) Heapify
Binary Heap - Extract Min
- Min key at root → save it
  - decrement $n$
- Sift the root key down
  - find smaller of two children
  - if larger, swap with this child
- Return saved root key

Leftist Property: Null path length

$\text{npl}(v) = \text{length of shortest path to null}$

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

Def: Leftist Heap is binary tree where:
- keys are heap ordered
- all nodes $v$, $\text{npl}(v.\text{left}) \geq \text{npl}(v.\text{right})$

Examples

Leftist Heaps: Meldable heaps
- can merge two heaps into single heap
- eg. One processor breaks: Awaiting jobs must be merged with another processor.

Analysis:
Both insert & extract-min take time proportional to tree height
Tree is complete $\Rightarrow O(\log n)$ time
Announcements: Thu 2/9

- Homework 1 - out now (preliminary)
  - Due: Tue, Feb 21, start of class
    → No late submissions
  - Basic data structures
    (+ amortization)
  - Trees
  - Union-find
  - Heaps

Amortization:

Amortized time? → O(1)