History:
1989: Seidel & Aragon
[Explosion of randomized algorithms]
Later discovered this was already known: Priority Search Trees from different context (geometry)
McCreight 1980

Intuition:
- Random insertion into BSTs \(\Rightarrow O(\log n)\) expected height
- Worst case can be very bad \(O(n)\) height
- Treap: A tree that behaves as if keys are inserted in random order

Example: Insert: k, e, b, o, f, h, w (Std. BST) 1 2 3 4 5 6 7

Along any path - Insertion times increase

Randomized Data Structures
- Use a random number generator
- Running in expectation over all random choices
- Often simpler than deterministic

Geometric Interpretation:

Treaps I

Obs: In a standard BST, keys are by inorder + insert times are in heap order (parent < child)

Example:

Treap: Each node stores a key + a random priority. Keys are in inorder. Priorities are in heap order.

? Is it always possible to do both?
Yes: Just consider the corresponding BST
**Insertion:** As usual, find the leaf and create a new leaf node.
- Assign random priority
- On backing out - check heap order and rotate to fix.

**Theorem:** A treap containing $n$ entries has height $O(\log \log n)$ in expectation (averaged over all assignments of random priorities)

**Proof:** Follows directly from BST analysis

**Implementation:** (See pdf notes)
- **Node:** Stores priority + usual...
- **Helpers:**
  - lowest priority ($p$) returns node of lowest priority among:
  - **Restructure:** performs rotation ($p$.left, $p$.right) to put lowest priority node at $p$.
  - **Demotion:** $c$.success in $c$.parent

**Example:**

**Deletion:** (cute solution) Find node to delete. Set its priority to $+\infty$. Rotate it down to leaf level and unlink.

**Example:**

**Treaps II**