Week 8 Summary: AVL Trees, Heaps, Priority Queues, and Hash Tables

Z5007 Programming and Data Structures · MTech DSAI · Odd Semester 2026-27 · Dr. Innocent Nyalala

1. Self-Balancing AVL Trees (Adelson-Velsky and Landis, 1962)

A plain binary search tree degrades into a chain of height $n-1$ on sorted insertions. An AVL tree maintains the invariant $|h(u.\text{left}) - h(u.\text{right})| \leq 1$ at every node u, storing $h(u)$ locally so balance factors $\text{bf}(u) = h(u.\text{left}) - h(u.\text{right})$ update in $O(1)$ time.

Height Bound Proof Sketch

Let $N(h)$ denote the minimum number of nodes in an AVL tree of height h. Because $N(0) = 1$, $N(1) = 2$, and $N(h) = 1 + N(h-1) + N(h-2)$, induction shows $N(h) = F_{h+3} - 1 \geq \varphi^h$ where $\varphi = (1+\sqrt{5})/2$. Consequently, $h < 1.44 \log_2(n+2) - 0.328$, guaranteeing $O(\log n)$ worst-case search, insertion, and deletion.

Case	Trigger Condition	Repair Action	Preserved Inorder
LL	$\text{bf}(z) = +2$ and $\text{bf}(z.\text{left}) \geq 0$	Single right rotation at z	Yes
RR	$\text{bf}(z) = -2$ and $\text{bf}(z.\text{right}) \leq 0$	Single left rotation at z	Yes
LR	$\text{bf}(z) = +2$ and $\text{bf}(z.\text{left}) < 0$	Left at z.left, then right at z	Yes
RL	$\text{bf}(z) = -2$ and $\text{bf}(z.\text{right}) > 0$	Right at z.right, then left at z	Yes

2. Binary Heaps and Priority Queues

A binary max-heap stores a complete binary tree inside a zero-indexed array $A[0..n-1]$ with $\text{parent}(i) = \lfloor (i-1)/2 \rfloor$, $\text{left}(i) = 2i+1$, and $\text{right}(i) = 2i+2$.

- **Insertion (`push`):** Append at index n and sift upward in $O(\log n)$ worst-case swaps.
- **Extraction (`pop`):** Swap $A[0]$ with $A[n-1]$, shrink size by 1, and sift down in $O(\log n)$.
- **Floyd Bottom-Up `heapify`:** Call `sift_down` from $\lfloor n/2 \rfloor - 1$ down to 0. Because $\sum_{h=0}^{\lfloor \log_2 n \rfloor} \lceil n/2^{h+1} \rceil \cdot h \leq 2n$, building a heap runs in $\Theta(n)$ time.

3. Hash Tables: Separate Chaining vs. Open Addressing

Given table capacity m, n stored keys, and load factor $\alpha = n/m$:

- **Separate Chaining:** Each slot stores a linked bucket. Under simple uniform hashing, expected search takes $1 + \alpha$ probes, remaining stable even near $\alpha \approx 1$.
- **Linear Probing:** Probes $h(k), (h(k)+1) \bmod m, \ldots$ in one flat array. Primary clustering increases unsuccessful probe length to $\frac{1}{2}(1 + 1/(1-\alpha)^2)$, so tables rehash around $\alpha \geq 0.65$ and mark deleted slots with tombstones rather than `None`.

4. Worked Exercise

Insert keys $60, 30, 40, 80, 70, 90, 75$ into an empty AVL tree. Inserting 40 triggers LR at 60, rotating into root $40(30, 60)$. Inserting 70 triggers RL at 60, giving $70(60, 80)$. After inserting 90 and 75, deleting 30 triggers RL at 40 with right child 80, leaving root 70, inorder $[40, 60, 70, 75, 80, 90]$, and height 2.