Discussion 2: Hash, Sort

CS101 Fall 2024

CS101 Course Team

Oct 2024

Hash

1. Used for tasks like error handling, IP address lookup, and domain name resolution.
2. Hash function: Deterministic, Equal objects hash to equal values, fast is better.
3. As Hash range in $[0, M)$, hash function should map to an index $0, ..., M-1$
4. Commonly used hash function: $(ax + b) mod M$
5. Load factor: the average number of objects per bin.

$$\lambda = \frac{n}{M}$$

6. Probability of at least one collision: increases very fast as λ increases:

$$1 - \frac{M(M-1)...(M-n+1)}{M^n}$$

Dealing with Collisions

1. **Chained Hash Tables**: Each bin in the hash table points to a linked list of entries that hash to the same index.
2. **Open Addressing**: Stored directly in the array, and when a collision occurs, the algorithm probes for the next available slot.
3. **Primary Clustering**: Primary Clustering is when different keys collide to form one big group. Think of this as "clusters of many colors". Even though these keys are all different, they end up in a giant cluster. In linear probing, we expect to get $O(lgn)$ size clusters.
   - **Linear Probing**: The next available slot is found by moving linearly through the array.
   - **Quadratic Probing**: steps through the array quadratically, no primary clustering.
4. **Double Hashing**: $h_1(k) + i \cdot h_2(k)$

Sort

Algorithm	**Best-case**	**Worst-case**	**Average-case**
Bubble Sort	$O(n)$	$O(n^2)$	$O(n^2)$
Insertion Sort	$O(n)$	$O(n^2)$	$O(n^2)$
Merge Sort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$

Table: Best-, worst-, and average-case time complexities of sorting algorithms.

Bubble Sort

1. Significantly worse than insertion sort.
2. **Flagged Bubble Sort**: If no swap occurs, stop.
3. **Range-Limiting**: Limit loops based on last swap location.
4. **Alternating**: Bidirectional version of bubble sort.

Merge Sort

1. Based on Divide and Conquer!
2. $\Theta(n)$ additional space.
3. Stable, Not in-place.

$$T(n) = \begin{cases} \Theta(1) & n = 1 \\ 2T(\frac{n}{2}) + \Theta(n) & n > 1 \end{cases}$$

4. How to solve recurrence?

Merge Sort

A method without Master Theorem (Cite: *Algorithms*)

$T(n) = 2T(\frac{n}{2}) + \Theta(n) \Rightarrow \exists$constant c, $T(n) \leq 2T(\frac{n}{2}) + cn$.

$$\begin{aligned} T(n) &\leq 2T(\frac{n}{2}) + cn \\ &\leq 2[2T(\frac{n}{4}) + c\frac{n}{2}] + cn = 4T(\frac{n}{4}) + 2cn \\ &\leq 4[2T(\frac{n}{8}) + c\frac{n}{4}] + 2cn = 8T(\frac{n}{8}) + 3cn. \end{aligned}$$

By induction, we obtain: $T(n) \leq 2^k T(\frac{n}{2^k}) + kcn$.
Substitute by $k = \log_2 n$, we got $T(n) \leq nT(1) + cn\log_2 n = O(n\log_2 n)$.