Hashtable vs Dictionary

1. Overview

- Hashtable: Non-generic, older collection, uses object keys/values.
- Dictionary<TKey, TValue>: Generic, type-safe, better performance, widely used today.

2. Internal Structure

- Hashtable: Uses buckets with open addressing (probing).
- Dictionary: Uses buckets with entries linked via 'next' pointers (separate chaining).

3. Collision Handling

- Hashtable: Probing (linear, quadratic, or double hashing). Can cause clustering.
- Dictionary: Separate chaining with linked lists of entries.

4. Time Complexity

- Average Case: O(1) for search, insert, delete.
- Worst Case: O(n) if all elements hash to the same bucket.

5. Safety & Usability

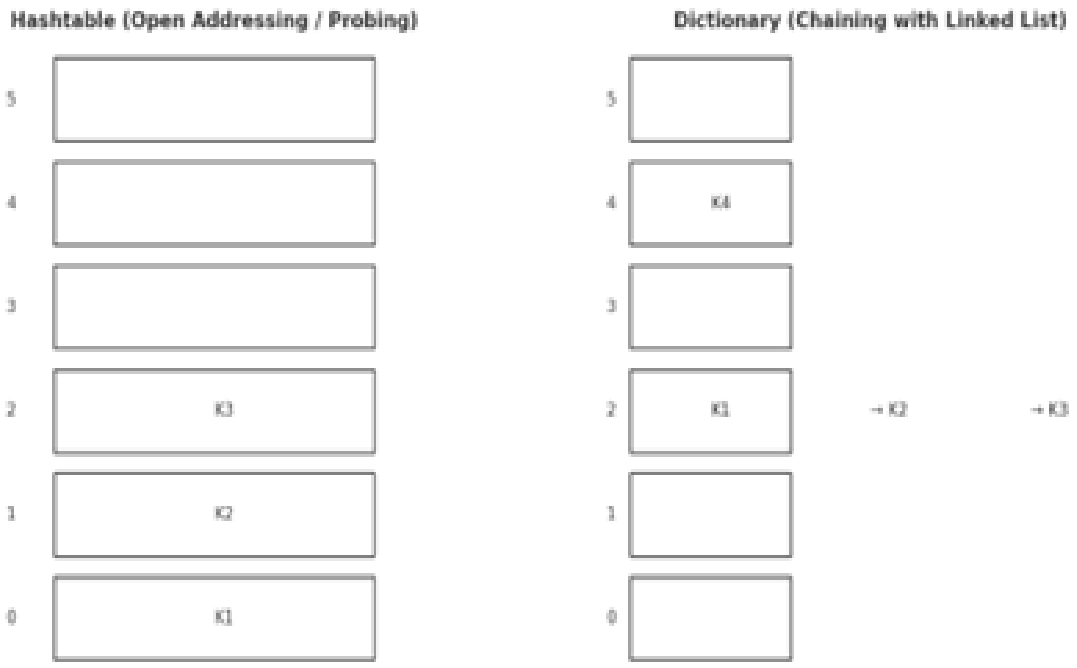
- Hashtable: Stores keys/values as object, requires boxing/unboxing, not type-safe.
- Dictionary: Generic, type-safe, compile-time safety, faster with value types.

6. Performance

- Hashtable: Slower due to boxing/unboxing and probing overhead.
- Dictionary: Faster, memory efficient, less clustering, uses rehashing when load factor exceeds ~0.72.

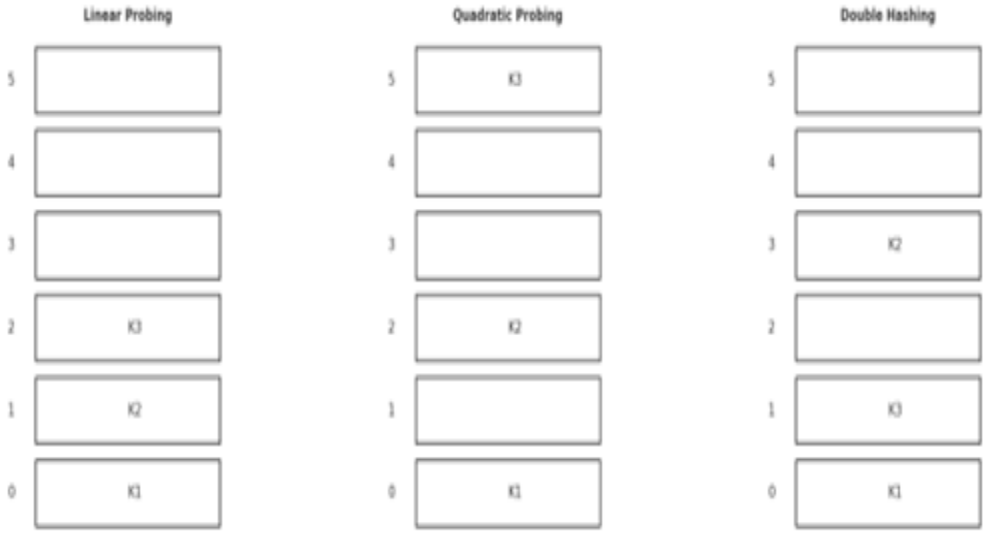
7. Diagrams

1. Hashtable (Open Addressing) vs Dictionary (Separate Chaining).



2. Types of Probing in Hashtable (Linear, Quadratic, Double Hashing).

[Diagram: Types of Probing]



8. Internal Structure Comparison

Collection	Internal Structure	Notes
Hashtable	Array of buckets (probing)	Older, open addressing
Dictionary<TKey,TValue>	Buckets + entries (linked via next)	Separate chaining
HashSet<T>	Buckets + slots (hashCode, value, next)	Unique elements
Array	Fixed array (T[])	Static size
ArrayList	Dynamic array (object[])	Non-generic
List<T>	Dynamic array (T[])	Generic, resizable
SortedList<TKey,TValue>	Parallel arrays (keys[], values[])	Sorted by key
SortedDictionary<TKey,TValue>	Red-Black Tree	Balanced BST

9. Operations Complexity (Average vs Worst)

Collection	Add/Insert	Delete	Search	Random Access	Notes
Array	Insert O(n)	Delete O(n)	Search O(n)	Access O(1)	Fixed size
ArrayList	Add O(1)*	Delete O(n)	Search O(n)	Access O(1)	Boxing/unboxing
List<T>	Add O(1)*	Delete O(n)	Search O(n)	Access O(1)	Generic, type-safe
Hashtable	Insert O(1) avg / O(n) worst	Delete O(1) avg / O(n) worst	Search O(1) avg / O(n) worst	N/A	Open addressing
Dictionary<TKey,TValue>	Insert O(1) avg / O(n) worst	Delete O(1) avg / O(n) worst	Search O(1) avg / O(n) worst	N/A	Separate chaining
HashSet<T>	Insert O(1) avg / O(n) worst	Delete O(1) avg / O(n) worst	Search O(1) avg / O(n) worst	N/A	Unique elements

SortedList<TKey,TValue>	Insert O(n)	Delete O(n)	Search O(log n)	Access by index O(1)	Parallel arrays
SortedDictionary<TKey,TValue>	Insert O(log n)	Delete O(log n)	Search O(log n)	N/A	Red-Black Tree