Samorie Turay Jr Reading 3 Answers

1. Searching/inserting data into sorted arrays is O(n) but Hash Tables are O(1) usually for searching, inserting and deleting operations. O(1) is better than O(n), making Hash tables more efficient.
2. The main strength is that it is easy to implement, overwriting the data. However, the weakness is that overwriting collisions largely leads to data loss and would make hash tables unsuitable because overwriting the data results in a loss of the previous values stored at that index.
3. Separate chaining uses linked lists to store multiple keys in the same bucket, while open addressing searches for alternative slots when collisions occur
4. Collisions are pretty hard to avoid due to hash tables having less slots than the possible number of keys.
5. A hash function that randomly choses table indices are unusable because they get in the way of efficient retrieval. The fact that they prevent the consistent mapping of keys illustrates this.
6. The worst performing hash hash function maps all keys to one singular index. This leads to poor performance because it results in a linked list of all elements in a bucket.
7. A good hash function is efficient to compute, prevents clustering, and distributes keys uniformly.
8. Character order is ignored by the naïve C implementation, which results in poor distribution for comparable strings. The hash table function in the Java standard library uses a polynomial rolling hash to take into account both character values and their locations, improving upon the naive string hashing implementation. This improves consistency and lowers collisions by guaranteeing that the hash value varies according to the character order.