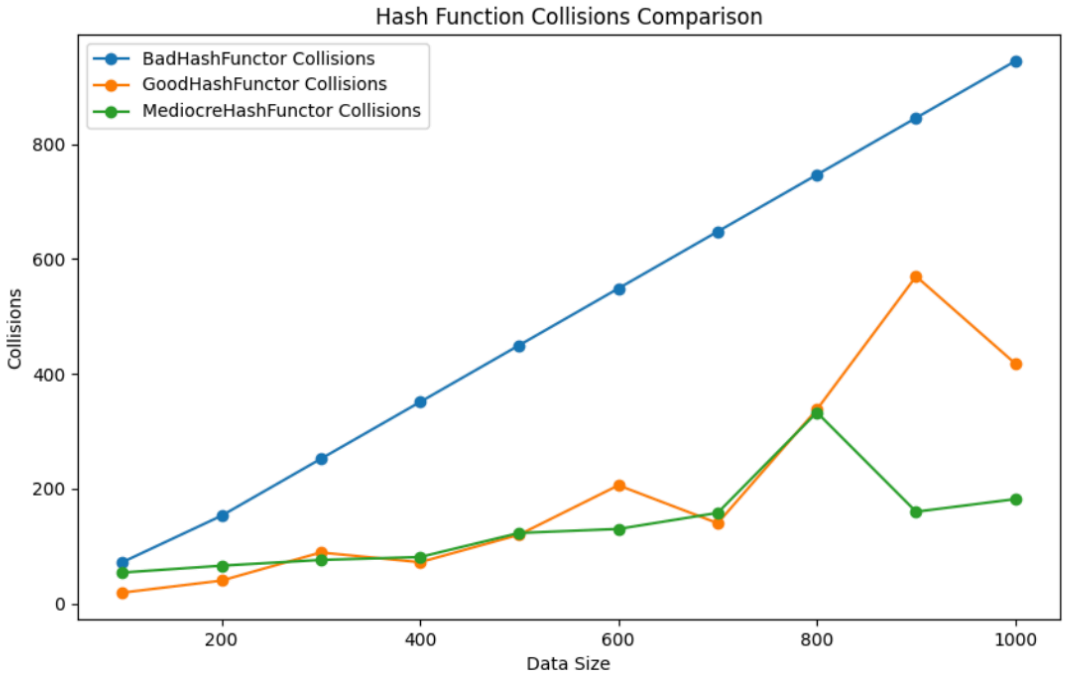


BadHashFunctor:

Hashing Function Explanation: This hash function simply adds the ASCII values of each character in the input string to the hash value. It doesn't perform any complex operations to ensure an even distribution of hash values.
Big-O Notation: O(N), where N is the length of the input string.
Expected Performance: This hash function is expected to perform poorly in terms of distributing hash values evenly, resulting in a higher number of collisions for most inputs.

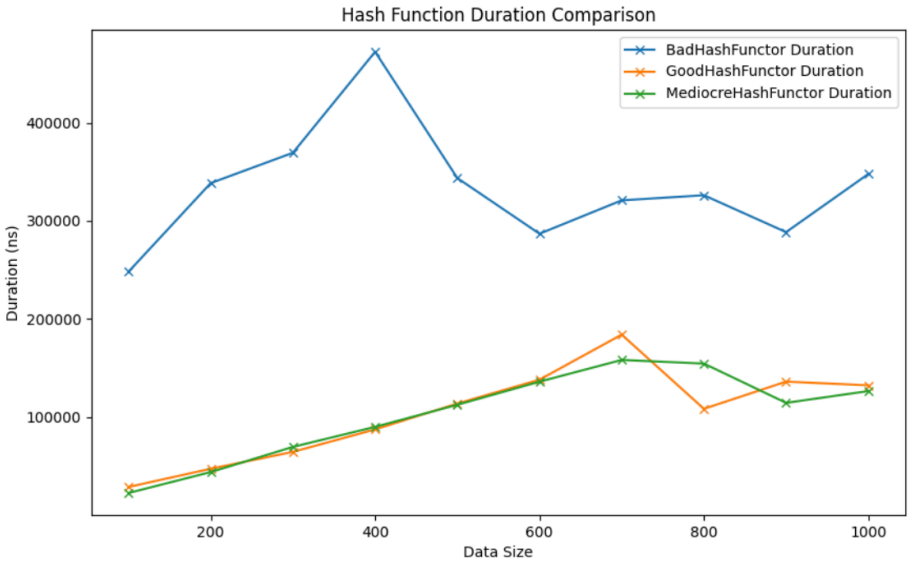
MediocreHashFunctor:

Hashing Function Explanation: This hash function is also implementing the DJB2 algorithm. Like GoodHashFunctor, it uses bitwise manipulation and multiplication to generate hash values. However, it may not be as efficient as GoodHashFunctor in terms of dispersion.
Big-O Notation: O(N), similar to BadHashFunctor and GoodHashFunctor.
Expected Performance: MediocreHashFunctor is expected to perform moderately in terms of distributing hash values, resulting in some collisions for certain inputs.

GoodHashFunctor:

Hashing Function Explanation: This hash function is implementing the well-known DJB2 algorithm. It uses a combination of bitwise manipulation and multiplication to create a more evenly distributed set of hash values for input strings.
Big-O Notation: O(N), similar to BadHashFunctor, as it still operates on each character in the input string.

Expected Performance: GoodHashFunctor is expected to perform well in terms of distributing hash values, resulting in fewer collisions compared to BadHashFunctor and MediocreHashFunctor.



Generally speaking, duration shows similar graph with the collisions. But GoodHashFunctor and MediocreHashFunctor have similar performance, which might be due to efficiency of Hashing Operations: While GoodHashFunctor is designed to provide a better distribution of hash values, it might involve slightly more complex operations, such as bitwise manipulation and multiplication. On the other hand, MediocreHashFunctor also uses similar operations but may not be as efficient in its implementation. As a result, the difference in computational cost between the two hash functions may not be significant.