Advantages of BST over Hash Table

Hash Table supports following operations in Θ(1) time.

1) Search

2) Insert

3) Delete

The time complexity of above operations in a self-balancing Binary Search Tree (BST) (like Red-Black Tree, AVL Tree, Splay Tree, etc) is O(Logn).

So Hash Table seems to beating BST in all common operations. When should we prefer BST over Hash Tables, what are advantages. Following are some important points in favor of BSTs.

1. We can get all keys in sorted order by just doing Inorder Traversal of BST. This is not a natural operation in Hash Tables and requires extra efforts.
2. Doing order statistics, finding closest lower and greater elements, doing range queries are easy to do with BSTs. Like sorting, these operations are not a natural operation with Hash Tables.
3. BSTs are easy to implement compared to hashing, we can easily implement our own customized BST. To implement Hashing, we generally rely on libraries provided by programming languages.
4. With BSTs, all operations are guaranteed to work in O(Logn) time. But with Hashing, Θ(1) is average time and some particular operations may be costly, especially when table resizing happens.