Dictionary and SortedList are both collection types in C# used to store key-value pairs, but they have some significant differences in terms of behavior, use cases, and performance. Here are the key differences between the two:

1. Ordering:

- Dictionary: A Dictionary<TKey, TValue> does not maintain any specific order of elements. Elements are stored based on their hash codes and key comparisons, so the order in which you add elements is not preserved.

- SortedList: A SortedList<TKey, TValue> maintains elements in a sorted order based on their keys. The elements are automatically sorted as you add them, ensuring that they are always in order.

2. Performance:

- Dictionary: Dictionary is generally more efficient for look-up operations because it uses hash tables, which provide fast access times on average. It offers constant-time look-up performance.

- SortedList: SortedList offers fast searching and enumeration of elements due to its sorted order. However, the look-up performance is logarithmic, which means it may be slower than a dictionary for very large datasets.

3. Key Type:

- Dictionary: The key type in a Dictionary must implement proper GetHashCode and Equals methods to provide efficient hash code generation and key comparisons. It doesn't require keys to be comparable.

- SortedList: The key type in a SortedList must be comparable, meaning it must implement the IComparable interface or be used with a custom IComparer to define the sorting order.

4. Order Guarantee:

- Dictionary: A `Dictionary` does not guarantee any specific order of elements. The order can change when elements are added or removed.

- SortedList: A `SortedList` guarantees that elements will be maintained in a sorted order based on the keys. The order is automatically maintained as elements are added or removed.

5. Efficiency for Range Operations:

- Dictionary: A dictionary is more efficient for look-up operations with constant-time complexity, but it is less efficient for range queries where you need elements within a specific key range.

- SortedList: A sorted list is more efficient for range queries because the elements are already in order, making it easier to find elements within a specific range.

6. Memory Overhead:

- Dictionary: `Dictionary` may have lower memory overhead compared to `SortedList` because it stores elements in a hash table, which doesn't guarantee a specific order.

- SortedList: `SortedList` has a slightly higher memory overhead to maintain the sorted order of elements.

In summary, the choice between `Dictionary` and `SortedList` depends on your specific requirements. If you need fast look-up operations, `Dictionary` is typically a better choice. If you require elements to be maintained in sorted order, `SortedList` is more suitable, especially when you need efficient range queries or want to ensure that elements are always sorted.