На отсортированных и случайных данных оба метода — QuickSort и IntroSort — показывают практически одинаковую производительность, так как структура данных позволяет эффективно выполнять разбиение массива.

Однако на массивах, отсортированных в обратном порядке, QuickSort демонстрирует худший случай работы, где время выполнения значительно увеличивается. В таких ситуациях IntroSort благодаря переключению на HeapSort сохраняет стабильное и быстрое выполнение, значительно превосходя QuickSort по скорости.

IntroSort оказывается более универсальным методом, особенно на сложных массивах, таких как обратно отсортированные или почти отсортированные данные. QuickSort же остаётся хорошим выбором для простых и случайных массивов, где он работает на уровне IntroSort.