Oppgave 3:

I oppgave 1 ble alle tallene generert tilfeldig, noe som betyr at sorteringsalgoritmen måtte håndtere en helt usortert liste. I oppgave 2 er de første og siste tredjedelene av listen allerede sortert, mens den midterste tredjedelen inneholder tilfeldige tall. Dette påvirker ytelsen til sorteringsalgoritmen.

Timsort, som brukes av både *Arrays.sort* og *Collections.sort*, er en hybrid sorteringsalgoritme som kombinerer egenskapene til mergesort og insertionsort. Timsort er spesielt effektiv på lister som allerede har noen sorterte sekvenser (runs). I oppgave 2 vil Timsort kunne dra nytte av de allerede sorterte delene av listen, noe som fører til bedre ytelse sammenlignet med en helt usortert liste som i oppgave 1.