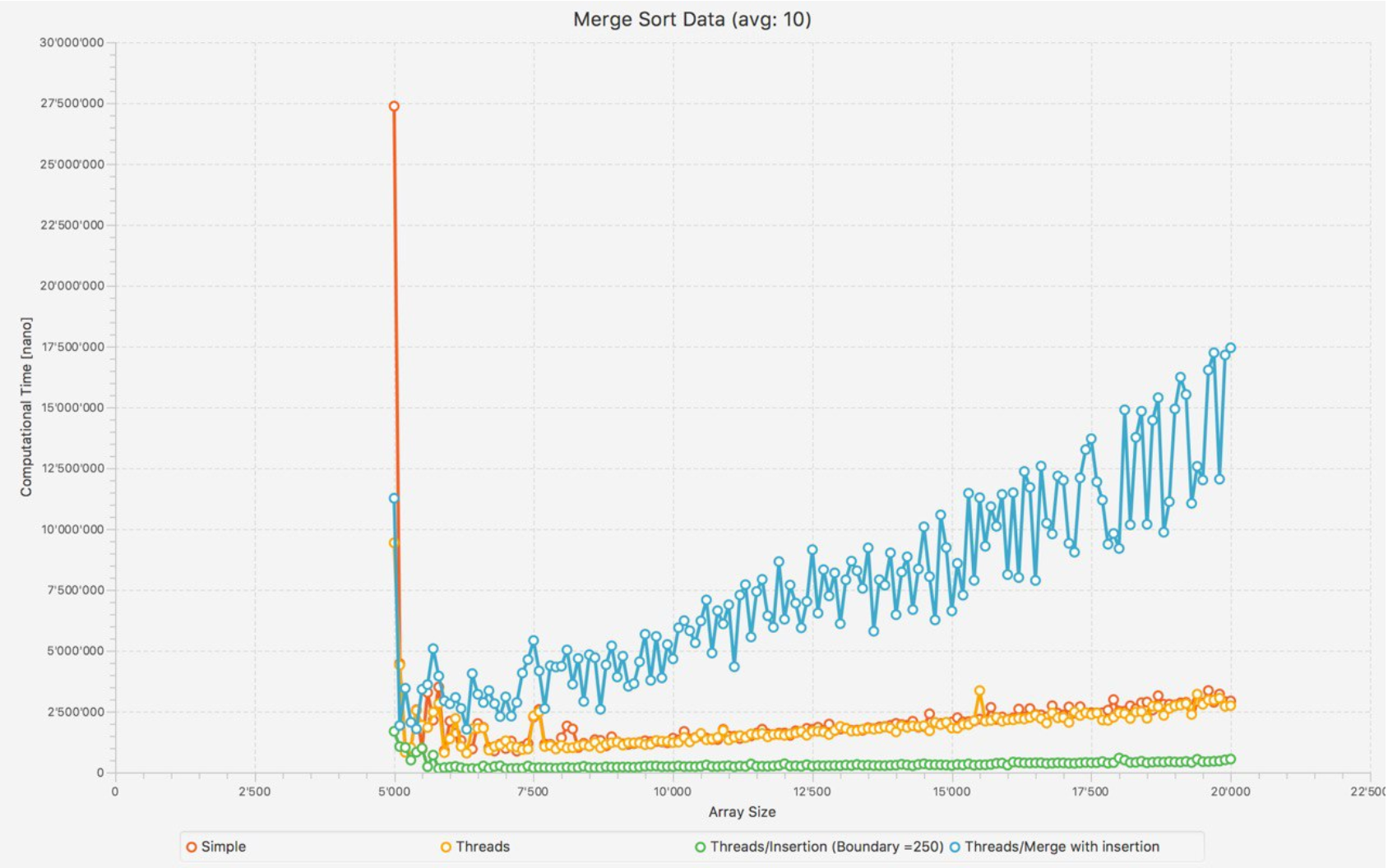
# Merge-Sort: Interpretation der Grafiken



1. Geschwindigkeit Simple gleich wie Geschwindigkeit Threads

**Beobachtung:** Die Simple-Version des Algorithmus, bei welcher Mergesort in der einfachsten Weise und ohne optimierende Eigenschaften implementiert wurde, läuft gleich schnell wie die Implementation mit Multithreading (4 Threads).

**Interpretation:** Folgende Alternativen scheinen möglich zu sein:

* Das Multithreading wurde nicht korrekt implementiert.
* Die Maschine läuft nicht mit mehreren Prozessorkernen.
* Der Aufwand fürs Multithreading gleicht sich mit dessen Gewinn bezüglich der Laufzeit aus.

1. Mergesort mit Insertionsort ab bestimmter Grenze ist am schnellsten

**Beobachtung:** Die grüne Kurve beschreibt die Geschwindigkeit des Mergesort-Algorithmus, bei welchem ab einer Arraygrösse von 250 auf Insertionsort gewechselt wird. Diese Version von Mergesort ist im Vergleich mit den anderen Versionen die schnellste.

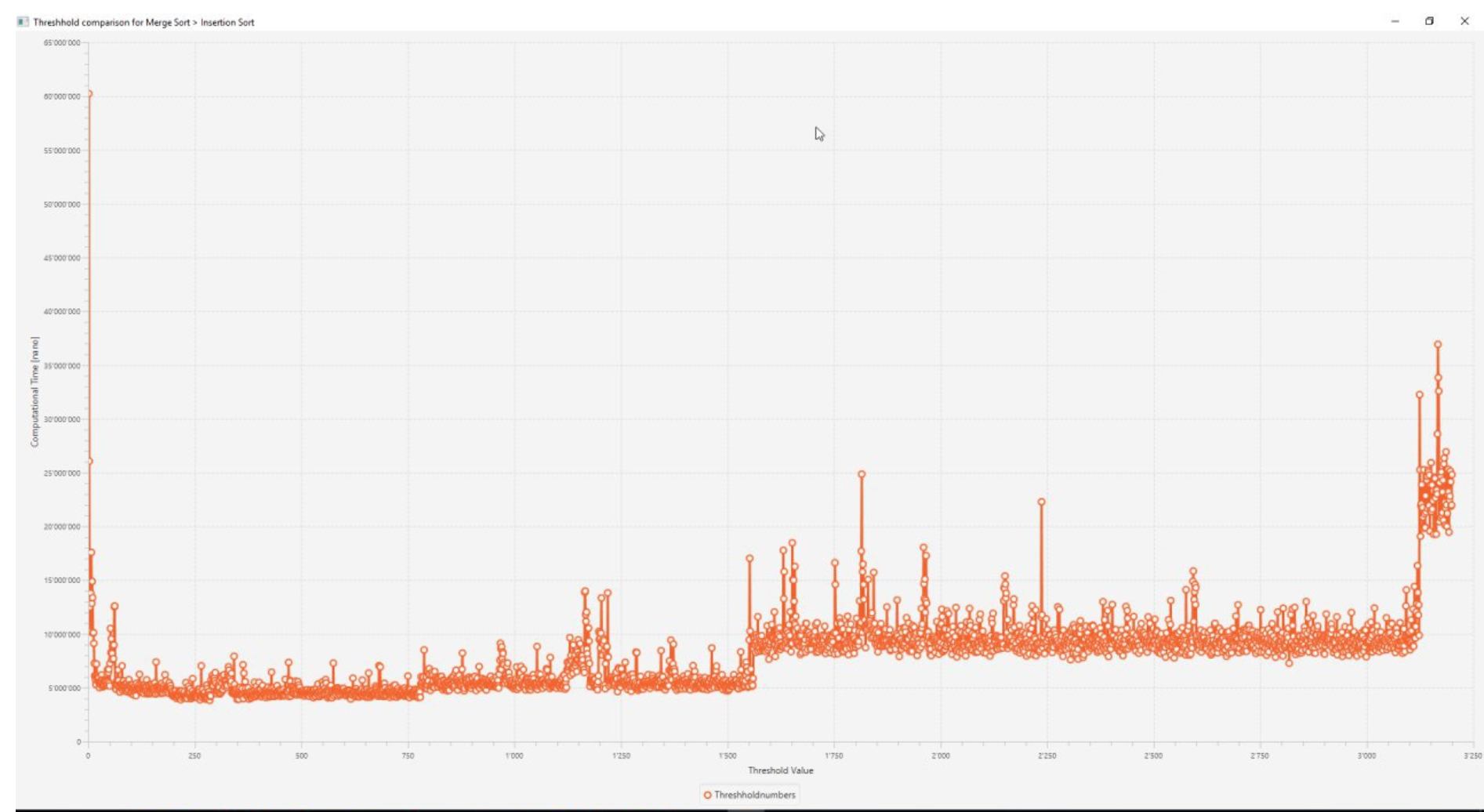
**Interpretation:** Folgende Alternativen scheinen denkbar:

* Ab einer bestimmten Arraygrösse scheint Insertionsort schneller zu sortieren als Mergesort.
* Insertionsort ist grundsätzlich schneller als Mergesort. Hierzu muss angemerkt werden, dass die auf der Grafik dargestellten Tests mit zufälligen Arrays durchgeführt wurden, was dem Average Case entspricht. Für Worstcase-Szenarien könnte die Grafik anders aussehen.

1. Mergesort mit Insertionsort für Recombining langsam

**Beobachtung:** Wenn die Recombining-Funktion von Mergesort durch Insertionsort ersetzt wird, resultiert die verhältnismässig langsamste Version des Mergesort-Algorithmus. (blaue Kurve)

**Interpretation:** Insertionsort sortiert indem ein Teilarray (Postfix), von dem angenommen wird, er sei unsortiert, Element für Element in einen sortierten Teilarray (Prefix) eingefügt wird. Mergesort liefert für das Recombining jedoch bereits sortierte Arrays. Somit wird der Postfix jeweils zweimal sortiert. Erstens beim Aufsplitten durch Mergesort, zweitens beim Recombining durch Insertionsort. Eine Kombination der zwei Algorithmen in der beschriebenen Weise führt somit zu einer höheren Komplexität als die Simple-Variante. Zusätzlich wird der Postfix „vorsortiert“, was ja nicht notwendig ist, da der Insertionsort vom Postfix Element um Element im Prefix am richigen Ort einfügt. Eine nichtnotwendige Arbeit wird somit zeitraubend durchgeführt.



1. Boudaries für Wechsel für Insertionsort

**Beobachtung:** Es sollte untersucht werden, wo die Grenze (Arraygrösse) liegt, ab welcher von Mergesort zu Insertionsort gewechselt werden soll, damit die optimale Laufzeit resultiert. Zu diesem Zweck wurde ein Array mit Grösse 100‘000 sortiert, wobei die Grenze variiert wurde (1 bis 3‘250), ab welcher auf Insertionsort umgestellt wird. Die obige Grafik zeigt die Laufzeit dieser Sortierungen.

**Interpretation:** Bei der Kurve in der obigen Grafik fällt die Abstufung auf. Dies macht Sinn, wenn man die Vorgehensweise von Mergesort bedenkt. Mergesort teilt einen Anfangsarray im Verlauf der Sortierung in immer kleinere Arrays der Grösse auf, wobei x die Rekursionstiefe bezeichnet. Der im beschriebenen Test verwendete 100‘000-Array wird also in Arrays der Grösse 50‘000, 25‘000, 12‘500, 6‘250, 3‘125, usw. geteilt. Wenn nun im Test Grenzen von 1 bis 3‘250 gewählt werden, sind nur diejenigen Werte für die Grenzwahl interessant, die um liegen. Es macht für die Laufzeit der Sortierung keinen Unterschied, ob die Grenze bei 3‘123 oder 3‘124 liegt, da bei einem Anfangsarray mit einer Grösse von 100‘000 in jedem Fall ein Array von 3‘125 geliefert wird.

Um diese Problematik zu verhindern, müssten für diesen Test Anfangsarrays der Grösse gewählt werden. So kann gewährleistet werden, dass die zu testende Grenze tatsächlich überprüft wird.

Je kleiner die Grenze gewählt wird, desto weniger relevant ist die oben beschriebene Problematik. Durch diese Untersuchungen hat sich gezeigt, dass auf dem verwendeten System eine Grenze von etwa 250 für den Wechsel von Mergesort zu Insertionsort optimal ist.