Performance Evaluierung

# Big-O-Notation

## Andrews Algorithm

Andrews Algorithmus besteht aus zwei Hauptteilen:

1. Sortierung der Punkte (Merge sort)

Beim Merge sort wird die Punktmenge rekursiv in kleinere Teillisten zerlegt, bis diese nur noch ein Element enthalten.  
Das Aufteilen selbst benötigt eine Tiefe von O(log n) (da die Liste bei jedem Schritt halbiert wird).  
Das anschließende Zusammenführen (Merge) der Teillisten erfordert für jede Rekursionsebene O(n) Zeit, da jedes Element genau einmal verglichen und eingefügt wird.

Daraus ergibt sich insgesamt eine Laufzeit von:

für den gesamten Sortiervorgang.

1. Bestimmung der konvexen Hülle

Anschließend werden die Punkte einmal durchlaufen, um die untere und obere Hülle zu bestimmen.  
Jeder Punkt wird dabei höchstens zweimal hinzugefügt bzw. entfernt, sodass dieser Teil linear ist, also O(n) benötigt.

Die Gesamtlaufzeit des Algorithmus ergibt sich aus der Summe beider Schritte:

## Quick hull

Analyse

1. Bestimmung der Extrempunkte

Die Suche nach den beiden Extrempunkten (minimale und maximale x-Koordinate) benötigt

1. Rekursive Aufteilung

Ähnlich wie beim Quicksort hängt die Laufzeit stark von der Verteilung der Punkte ab. Bei günstiger (gleichmäßiger) Verteilung werden die Teilmengen gleichmäßig (linear) kleiner.

Im besten Fall halbiert jeder rekursive Schritt die Punktmenge ähnlich wie bei Quicksort und es erfordert Vergleiche (zur Bestimmung des Punktes mit maximalem Abstand).

Rekursionstiefe: .

Wenn alle Punkte auf der konvexen Hülle liegen oder die Rekursion sehr unbalanciert verläuft (ähnlich zu schlechtem Pivot in Quicksort),  
dann wird die Teilmenge nur um einen Punkt reduziert.

Im Allgemeinenen kann man aber von

ausgehen.

# Performance results

Ziel des Tests war der Vergleich der Laufzeiten zweier Algorithmen zur Bestimmung der konvexen Hülle:

* **Quick hull**
* **Andrews Monotone Chain Algorithmus**

Die Messungen wurden mit zufällig generierten Punktmengen unterschiedlicher durchgeführt (von 50 000 bis 10 000 000 Punkten).  
Für jede Konfiguration wurde die durchschnittliche Laufzeit (in Sekunden und Millisekunden) Größe im Backend ermittelt.  
Bei **50 Millionen Punkten** wurde der Andrews-Algorithmus aufgrund der sehr langen Laufzeit abgebrochen.

| **Algorithmus** | **Input Points** | **Hull Points** | **Laufzeit (Sekunden)** | **Bemerkung** |
| --- | --- | --- | --- | --- |
| QuickHull | 50 000 | 22 | **0.065 s** | sehr schnell |
| Andrews | 50 000 | 22 | **0.201 s** | ~3× langsamer |
| QuickHull | 500 000 | 39 | **0.678 s** |  |
| Andrews | 500 000 | 39 | **2.675 s** | ~4× langsamer |
| QuickHull | 5 000 000 | 43 | **7.34 s** |  |
| Andrews | 5 000 000 | 43 | **36.65 s** | ~5× langsamer |
| QuickHull | 10 000 000 | 37 | **16.25 s** |  |
| Andrews | 10 000 000 | 37 | **79.98 s** | ~5× langsamer |
| QuickHull | 50 000 000 | 48 | **124.55 s** |  |
| Andrews | 50 000 000 | — | **abgebrochen** | Laufzeit zu hoch |

**Interpretation**

* Beide Algorithmen liefern die gleiche Anzahl von Hüllpunkten, also korrekte Ergebnisse.
* Quick hull zeigt in allen Tests deutlich bessere Laufzeiten – zwischen dem 3- bis 6-fachen Geschwindigkeitsvorteil gegenüber Andrews.
* Bei großen Datensätzen (>10 Millionen Punkte):
  + Quick hull behält akzeptable Laufzeiten bei (z. B. 124 s bei 50 Mio. Punkten),
  + während Andrews überproportional langsam wird und abgebrochen werden musste.