Ein einfacher Algorithmus für konvexe Hüllen in 2D

Herman Haverkort*

28 November 2018

Eingabe: n unterschiedliche Punkte $S[1..n]$ in der Ebene.
Ausgabe: die Ecken der konvexen Hülle von S, gegen den Uhrzeigersinn

$u \leftarrow$ lexikographisch-kleinster Punkt in S
Entferne u aus S (also noch $n-1$ Punkte übrig)
Sortiere S gegen den Uhrzeigersinn um u (bei gleicher Richtung ansteigend nach Entfernung)
Initialisiere Stapel, push u, push $S[1]$
for $i \leftarrow 2$ **to** $n-1$ **do**
    while zweitoberster Punkt, oberster Punkt, $S[i]$ nicht nach links abbiegt **do**
        pop obersten Punkt
    push $S[i]$
Pop alle Punkte vom Stapel und gib sie in umgekerhter Reihenfolge aus

Implementierung: beim Sortieren, $(p < q) = upq$ biegt nach links ab, oder upq ist kollinear und p liegt zwischen u and q. Die while-Schleife wird abgebrochen wenn nur ein Punkt auf dem Stapel liegt.

Korrektheit: sei $S[j]$ der erste Punkt mit y-Koordinate höher als die y-Koordinate von u. Man kann jetzt nachvollziehen, dass nach dem Schleifendurchlauf mit $i = j$, der Stapel u, $S[j-1]$ und $S[j]$ enthält. Ab dann läuft der Algorithmus so wie das nicht-randomisierte inkrementelle Verfahren in Abschnitt 4.1.2, wobei die konvexe Hülle der bisher betrachteten Punkte immer auf dem Stapel steht und wobei man sich vorstellt, dass z unmittelbar rechts von u liegt. Die Reihenfolge, in der die Punkte hinzugefügt werden, führt dazu, dass wir immer eine Kante entfernen müssen, die eine Ecke oben auf dem Stapel hat—die erste zu entfernende Kante wird also in $O(1)$ Zeit gefunden, ohne dass wir weitere Daten- oder Zeigerstrukturen anlegen müssen.

Laufzeit: nicht mehr Pop-Operationen als Push-Operationen; Gesamtlaufzeit $O(n)$ + Sortieren.

Dieser Algorithmus ist bekannt als “Graham Scan”:
siehe z.B. https://en.wikipedia.org/wiki/Graham_scan.

* Institut für Informatik, Universität Bonn, Germany, cs.herman@haverkort.net