Karlsruher Institut für Technologie
Fakultät für Informatik
ITI Wagner

Algorithmische Geometrie
Sommersemester 2014
Martin Nöllenburg/Benjamin Niedermann

# Übungsblatt 4 - Lineare Programmierung

**Ausgabe:** 06. Mai 2014
**Abgabe:** 13. Mai 2014

## 1 Korrektheitsbeweis

In der Vorlesung wurde der Algorithmus RandomPermutation(A) vorgestellt.

a) Beweisen Sie seine Korrektheit, indem Sie zeigen, dass jede mögliche Permutation von A gleich wahrscheinlich ist.

b) Zeigen Sie außerdem, dass die Aussage aus a) nicht mehr stimmt, wenn man den Ausdruck $r \leftarrow \text{Random}(k)$ durch $r \leftarrow \text{Random}(n)$ ersetzt.

## 2 Paranoia

**Algorithmus 1 :** ParanoidMax

```
Eingabe : Endliche Menge A ⊂ ℝ
Ausgabe : Maximum max_{a∈A} a der Menge
if |A| = 1 then
    return einziges Element a ∈ A
else
    a = zufällig gewähltes Element aus A
    b = ParanoidMax(A \ {a})
    if b ≥ a then
        return b
    else
        prüfe unnötigerweise jedes Element aus A \ {a}, um sicherzugehen, dass a wirklich
        größer ist
        return a
```

Betrachten Sie den Algorithmus 1, der das Maximum einer Menge von Zahlen berechnet. Schätzen Sie die asymptotische Worst-Case-Laufzeit des Algorithmus scharf ab. Beachten Sie die zufällige Auswahl des Elementes $a$ und zeigen Sie, dass die erwartete Laufzeit echt besser ist.

*Bitte umblättern*

## 3 Züge

Gegeben seien $n$ Züge, die auf parallelen Gleisen fahren. Jeder Zug $z_i$ $(i = 1, \ldots, n)$ fahre dabei mit konstanter Geschwindigkeit $v_i$ und nehme zum Startzeitpunkt $t = 0$ (hier startet nur die Zeit, die Züge haben schon die volle Geschwindigkeit) die Position $k_i$ auf der Strecke ein. Geben Sie einen Algorithmus an, der in $O(n \log n)$ Zeit berechnet, welche Züge bis zu einem gegebenen Zeitpunkt $t_{\text{stop}} > 0$ mindestens einmal in Führung waren.

*Hinweis:* Der in der Vorlesung vorgestellte Algorithmus zu Berechnung vom Schnitt mehrerer Halbebenen könnte dabei nützlich sein.

## 4 Rückblick: Polygon Partitionieren

In der dritten Vorlesung wurde ein Verfahren vorgestellt welches, mit Hilfe von einer doppelt verketteten Kantenliste (DCEL), ein einfaches Polygon in monotone Polygone auftrennt. Im Verlauf des Algorithmus werden neue Kanten zur DCEL hinzugefügt, nämlich Diagonalen um Split– oder Merge–Knoten zu entfernen. Dabei wurde angenommen, dass wir eine Kante in konstanter Zeit zur DCEL hinzufügen können.

Argumentieren Sie, warum das Hinzufügen von Kanten in eine DCEL im Allgemeinen *nicht* in konstanter Zeit möglich ist. Zeigen Sie dann, wie man im Fall des in der Vorlesung vorgestellten Algorithmus zur Polygon-Partitionierung Kanten trotzdem in $O(1)$ in die DCEL einfügen kann.