Technische Universität Berlin
Fakultät II – Institut für Mathematik
Prof. Günter M. Ziegler
Axel Werner

Wintersemester 2004/2005
Algorithmische Geometrie

5. Übungsblatt

Abgabe am 2.12.2004

Aufgabe 17: **4 Punkte**

Sei H eine endliche Menge von mindestens drei Halbebenen im $\mathbb{R}^2$. Eine Halbebene $h \in H$ heißt *redundant*, wenn $\bigcap H \neq \emptyset$ und h keine Kante zu $\bigcap H$ beiträgt. Zeige:

(a) Ist $\bigcap H = \emptyset$, dann gibt es schon drei Halbebenen $h_1, h_2, h_3 \in H$, so dass $h_1 \cap h_2 \cap h_3 = \emptyset$ ist.

(b) Für jede redundante Halbebene $h \in H$ existieren zwei andere Halbebenen $h_1, h_2 \in H$, so dass gilt: $h_1 \cap h_2 \subset h$.

Aufgabe 18: **4 Punkte**

Gegeben seien die Zahlen $1, 2, \ldots, n$. Betrachte die beiden folgenden Algorithmen und berechne die erwartete Laufzeit.

(a) Starte mit der Zahl n und ziehe gleichverteilt eine kleinere Zahl aus $\{1, 2, \ldots, n-1\}$. Dann verfahre rekursiv. Wenn 1 erreicht worden ist, brich ab.

(b) Benutze den gleichen rekursiven Algorithmus wie in (a), aber starte mit einer zufällig gewählten Zahl aus $\{1, 2, \ldots, n\}$.

Aufgabe 19: **4 Punkte**

Betrachte folgenden Algorithmus zur Berechnung von zufälligen Permutationen:

Algorithm: Permutation

Input: die identische Permutation $(x_1, x_2, \ldots, x_n) = (1, 2, \ldots, n)$
Output: eine zufällige Permutation π gegeben durch $(x_1, x_2, \ldots, x_n)$, wobei $\pi(i) = x_i$ ist

1. **for** $k = n, n-1, \ldots, 3, 2$
2. ziehe zufällige Zahl r gleichverteilt aus $\{1, \ldots, k\}$
3. vertausche x_k und x_r

a) Beweise, dass dieser Algorithmus jede Permutation mit der gleichen Wahrscheinlichkeit produziert, d.h. Permutationen gleichverteilt zieht.

b) In obigem Algorithmus wird eine zufällige Zahl (gleichverteilt) aus $\{1, \ldots, k\}$ gezogen. Angenommen, man kann nur zufällige (gleichverteilte) Bits (0 oder 1) in konstanter Zeit ziehen. Wie ist dann die erwartete Laufzeit des Algorithmus?

Aufgabe 20: **4 Punkte**

Gegeben seien n paarweise verschiedene reelle Zahlen $x_1, \ldots, x_n$. Der folgende randomisierte Algorithmus berechnet das Maximum unter diesen Zahlen:
Wähle eine zufällige Zahl x_i und untersuche durch Vergleichen mit den anderen x_j, ob x_i die größte ist. Falls ja, so gib x_i als Maximum zurück, ansonsten wiederhole obigen Schritt mit einer andere Zahl x_k. Dabei wird jedesmal ein neues x_k benutzt, d.h. eines, das vorher noch nicht gezogen wurde.
Dieser Algorithmus gibt offenbar immer das korrekte Ergebnis zurück. Berechne die erwartete Laufzeit. Um wieviel schneller ist der Algorithmus als die Methode, jede Zahl gegen jede zu testen?