Prof. Dr. Ir. Joost-Pieter Katoen

Christian Dehnert, Jonathan Heinen, Thomas Ströder, Sabrina von Styp

Hinweise:

- Die Übungsblätter sind in Gruppen von je 3 Studierenden aus der gleichen Kleingruppenübung zu bearbeiten.
- Die Lösungen müssen bis **Freitag, den 13. Juli um 14:00 Uhr** in den entsprechenden Übungskasten eingeworfen werden. Sie finden die Kästen am Eingang Halifaxstr. des Informatikzentrums (Ahornstr. 55). Alternativ können die Lösungen auch zu Beginn der zugehörigen Globalübung im Audimax abgegeben werden.
- Namen und Matrikelnummern der Studenten sowie die Nummer der Übungsgruppe sind auf jedes Blatt der Abgabe zu schreiben. Heften bzw. tackern Sie die Blätter!

Aufgabe 1 (Dynamische Programmierung): (4 + 6 + 5 = 15 Punkte)

a) Schreiben Sie eine Funktion, welche die Länge der Longest Common Subsequence (LCS) für zwei gegebene Zeichensequenzen berechnet. Die Zeichensequenzen seien dabei der Einfachheit halber `int` Arrays `seq1` und `seq2` mit den Längen `l1` and `l2`. Demnach soll Ihre Funktion die folgende Signatur haben:

```
int lcs(int seq1[], int l1, int seq2[], int l2)
```

b) Bestimmen Sie gemäß dem in der Vorlesung vorgestellten Verfahren die LCS der Wörter BACHELOR und AACHEN. Geben Sie hierzu die berechnete Matrix an und kennzeichnen Sie den Pfad, der zur Rekonstruktion des Ergebnisses genutzt wird.

c) Betrachten Sie folgendes Szenario. Ihnen wird eine Klausur gestellt, welche n Aufgaben umfasst. Jede dieser Aufgaben hat eine Punktzahl p_i und eine von Ihnen geschätzte Zeit t_i, die Sie zum Lösen der Aufgabe benötigen ($1 \leq i \leq n$). Geben Sie die maximale Punktzahl, welche Sie in T Zeiteinheiten erreichen können, als Rekursionsgleichung an (inklusive der passenden Argumente).

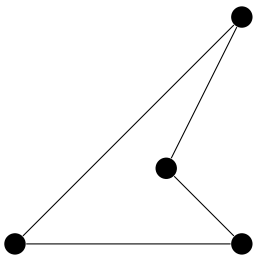
Aufgabe 2 (Geometrie): (8 + 7 = 15 Punkte)

a) Gegeben seien die folgenden Punkte:

(13, 7), (3, 8), (5, 2), (9, 4), (6, 11), (12, 14), (7, 13), (10, 8), (17, 14), (18, 5)

Bestimmen Sie mit Hilfe des Graham-Scans die konvexe Hülle der oben angegebenen Punkte. Geben Sie alle berechneten Determinanten und den jeweils aktuellen Zustand des Stacks an.

b) Schreiben Sie eine Funktion, die zu einem einfachen (aber nicht notwendigerweise konvexen) Polygon und einem Punkt einen Wahrheitswert zurück liefert, der angibt, ob sich der Punkt innerhalb des Polygons befindet (Punkte auf dem Rand des Polygons sind dabei innerhalb des Polygons). Hierbei können Sie davon ausgehen, dass eine Datenstruktur `P` existiert, die zwei ganzzahlige Felder `x` und `y` besitzt. Diese Datenstruktur modelliert Punkte im zweidimensionalen Raum. Solche Punkte können mittels des Vergleichsoperators == auf Gleichheit getestet werden. Das Polygon sei für die Eingabe als Array von Punkten gegeben, wobei jeweils aufeinander folgende Punkte sowie der letzte und erste Punkt miteinander verbunden sind. Als Beispiel ist nachfolgend ein Polygon und seine Repräsentation als Array von Punkten angegeben.



```
polygon[0].x = 0, polygon[0].y = 0
polygon[1].x = 3, polygon[1].y = 0
polygon[2].x = 2, polygon[2].y = 1
polygon[3].x = 3, polygon[3].y = 4
```

Ihre Funktion soll folgende Signatur haben, wobei `n` die Anzahl der Punkte ist, aus denen das Polygon besteht:

```
boolean inPolygon(P point, P polygon[], int n)
```

Sie dürfen davon ausgehen, dass eine Funktion `float winkel(P p1, P p2, P p3)` existiert, welche den Winkel zwischen den Strecken `p2p1` und `p2p3` berechnet. Sie liefert Fließkommazahlen w mit $-180 < w \leq 180$ zurück. Außerdem dürfen Sie annehmen, dass keine Rundungsfehler bei Berechnungen mit Fließkommazahlen auftreten.

Hinweis: Eine mögliche Lösung arbeitet ähnlich wie der Graham-Scan. Überlegen Sie sich, wie Sie Winkel zwischen bestimmten Strecken dazu nutzen können, das Problem zu lösen.