Diskrete Mathematik I

5. Übung

1. Sei G ein gerichteter oder ungerichteter Graph mit Kantengewichten $c : E(G) \to \mathbb{Z}_+$, und seien $s, t \in V(G)$ so gewählt, dass t von s aus erreichbar ist. Zeigen Sie, dass die minimale Länge eines s-t-Pfades in G gleich der maximalen Anzahl von s-t-Schnitten ist, so dass jede Kante e in höchstens $c(e)$ von ihnen enthalten ist. (2 Punkte)

2. Sei G ein gerichteter Graph mit konservativen Kantengewichten $c : E(G) \to \mathbb{R}$, und seien $s, t \in V(G)$ so gewählt, dass t von s aus erreichbar ist. Zeigen Sie, dass die minimale Länge eines s-t-Pfades in G gleich dem Maximum von $\pi(t) - \pi(s)$ über alle zulässigen Potentiale π von (G, c) ist. (2 Punkte)

3. Die Timingrestriktionen eines Logikchips können durch einen gerichteten Graphen G mit Kantengewichten $c : E(G) \to \mathbb{R}_+$ modelliert werden: Die Knoten des Graphen repräsentieren dabei Speicherelemente, die Kanten repräsentieren Pfade durch die kombinatorische Logik, und die Kantengewichte geben Schätzwerte für die Propagierungszeit eines Signals über die entsprechende Kante im schlechtesten Fall an. Eine wichtige Aufgabe im Design von VLSI-Schaltkreisen (VLSI: *"Very Large Scale Integrated"*, höchstintegriert) ist es, optimale Ankunftszeiten für das Taktsignal an jedem Speicherelement zu berechnen, also eine Abbildung $a : V(G) \to \mathbb{R}$, so dass $a(v) + c((v, w)) \leq a(w) + T$ für alle $(v, w) \in E(G)$ und eine möglichst kleine Zahl $T \in \mathbb{R}_+$ (T ist die Zykluszeit des Chips, $a(v)$ die Abgangszeit eines Signals am Speicherelement v und $a(v) + T$ die spätestzulässige Ankunftszeit während eines betrachteten Zyklus).

   a) Reduzieren Sie das Problem, den optimalen Wert T zu finden, auf das Minimum-Mean-Cycle-Problem. (1 Punkt)

   b) Zeigen Sie, wie die Zahlen $a(v)$ einer optimalen Lösung effizient bestimmt werden können. (1 Punkt)

   c) Typischerweise sind einige der Zahlen $a(v)$ fest vorgegeben. Zeigen Sie, wie man das Problem in diesem Fall lösen kann. (2 Punkte)

Abgabe der Aufgaben 1-3 bis spätestens Di, den 25.11.03 vor der Vorlesung.

Abgabe von Aufgabe 4 (s. Rückseite) bis spätestens Di, den 02.12.03 vor der Vorlesung.

4. **Programmieraufgabe:**

a) Implementieren Sie einen Fibonacci-Heap mit den Operationen für das Einfügen von Objekten, das Ändern des Schlüsselwertes eines beliebigen Objektes im Heap und das Auffinden und Löschen des Objektes mit dem kleinsten Schlüsselwert.

(8 Punkte)

c) Implementieren Sie Dijkstra's Kürzeste-Wege-Algorithmus, einmal in der naiven Variante mit quadratischer Laufzeit (ohne Verwendung von Heaps), und einmal unter Verwendung eines Fibonacci-Heaps zum Auffinden eines nicht permanent gelabelten Knotens mit minimalem Label. Vergleichen Sie die erzielten Laufzeiten. Ihr Programm soll einen gerichteten Graphen G mit Kantengewichten sowie Paare von Knoten, zwischen denen ein kürzester Pfad in diesem Graphen gefunden werden soll, aus einer ASCII-Textdatei mit folgendem Format einlesen:

- Die erste Zeile enthält die Anzahl $|V(G)|$ der Knoten.
- Die zweite Zeile enthält die Anzahl $|E(G)|$ der Kanten.
- Die folgenden $|E(G)|$ Zeilen definieren die Kantenmenge $E(G)$ und die Kantengewichte durch Angabe dreier durch Leerzeichen getrennter Zahlen auf jeder Zeile. Die ersten beiden Zahlen liegen in $\{0, \ldots, |V(G)| - 1\}$ und geben zwei Knoten an, die eine vom ersten zum zweiten Knoten orientierte Kante definieren. Die dritte Zahl gibt das Gewicht dieser Kante an.
- Die übrigen Zeilen geben Paare von Knoten an, für die ein kürzester Pfad in G vom ersten zum zweiten angegebenen Knoten zu finden ist.

Beispiel:

```
4
3
2 3 1.4
0 2 3.14159
3 0 2
0 3
1 2
```

Ihr Programm soll für jedes Knotenpaar $s, t \in V(G)$, für das ein kürzester s-t-Pfad zu finden ist, die Länge eines solchen Pfades ausgeben sowie die Folge der Knoten, die er vom Start- bis zum Zielknoten durchläuft, falls t von s aus erreichbar ist, und andernfalls die Information, dass es keinen s-t-Pfad in G gibt. (8 Punkte)

Bitte schicken Sie Ihren Programmcode bis zum umseitig angegebenen Datum an `mueller@or.uni-bonn.de`. Erlaubte Programmiersprachen und sonstige Bedingungen wie in Aufgabe 5 des zweiten Übungsblattes.