# Theorie-Übungsblatt 3

Algorithmen für Routenplanung, Sommer 2010

**Ausgabe** 03. Mai 2010
**Abgabe** keine

## Problem 1: Arc-Flags, Aufwärmübung *

Gegeben sei ein Graph $G = (V, E, \text{len})$ mit einer Partition $\mathcal{P} := \{C_1, \dots, C_k\}$ auf $V$. Das Arc-Flag einer Kante $e \in E$ bezüglich einer Zelle $C \in \mathcal{P}$ sei mit $\text{AF}_C(e)$ bezeichnet.

(a) Es sei $|\mathcal{P}| = 1$. Weiterhin werde die All-Pair-Shortest-Path-Variante für die Vorberechnung benutzt. Unterscheidet sich (wenn ja, worin) der Suchram, das heißt die Anzahl relaxierter Kanten bzw. abgearbeiter Knoten, des Arc-Flags-Algorithmus von DIJKSTRA's Algorithmus? Begründen Sie Ihre Antwort.

(b) Es sei nun $|\mathcal{P}| = n$, das heißt jede Zelle enthält genau *einen* Knoten des Graphen. Zeigen Sie: Eine $s$-$t$-Anfrage besucht nur Knoten entlang des kürzesten $s$-$t$-Weges. Muss dazu der kürzeste Weg eindeutig sein?

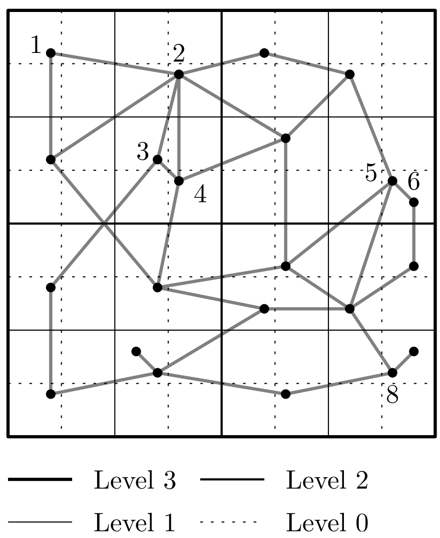
## Problem 2: Multi-Level Partitionierung **

Gegeben sei ein Graph $G = (V, E, \text{len})$. Eine *multi-level Partition* $\mathcal{P}$ mit $k$ Levels sei rekursiv wie folgt definiert:

- $\mathcal{P}_k := \{V\}$
- $\mathcal{P}_i = \bigcup_{C \in \mathcal{P}_{i+1}} P_C$ wobei $P_C$ eine Partition der Zelle $C$ ist. Jede Zelle $C$ der Partition $\mathcal{P}_{i+1}$ wird also "verfeinert".

*Bemerkung*: Arc-Flags $\text{AF}_C^{(i)}(e)$ auf Level $i$ beziehen sich auf Zellen $C \in \mathcal{P}_{i-1}$.

Weiterhin sei der *gemeinsame Level* zweier Knoten $u$ und $v$ das kleinste Level $i$ für das eine Zelle $C \in \mathcal{P}_i$ existiert die sowohl $u$ als auch $v$ enthält.




(a) Betrachten Sie den Graphen aus obiger Abbildung. Geben Sie das gemeinsame Level der Knotenpaare $(1, 2)$, $(1, 4)$, $(5, 6)$ und $(3, 8)$ an.

(b) Geben Sie ein effizientes (algorithmisches) Verfahren zur Bestimmung des gemeinsamen Levels zweier Knoten $u$ und $v$ an.

*Hinweis:* Benutzen Sie eine geschickte Nummerierung der Zellen auf dem untersten Level, die Sie an die Knoten speichern.

## **Problem 3:** Multi-Level Arc-Flags ***

Gegeben sei ein Graph $G = (V, E, \text{len})$ und eine multi-level Partition $\mathcal{P} = \{\mathcal{P}_0, \ldots, \mathcal{P}_k\}$.

(a) Sei $P := [s, v_1, v_2, \ldots, v_k, t]$ ein kürzester $s$-$t$-Weg in $G$. Zeigen Sie: Der gemeinsame Level der Knoten $v_i$ und $t$ nimmt mit zunehmendem $i$ nicht notwendigerweise monoton ab.

(b) Welche Konsequenz ergibt sich daraus für die Vorberechnung von Arc-Flags basierend auf Randknoten?

(c) Geben Sie den Algorithmus zur Vorberechnung basierend auf Randknoten in Pseudo-Code an. Sie können dabei Dijkstra's Algorithmus als Baustein verwenden.

(d) Erweitern Sie Dijkstra's Algorithmus zu einer multi-level Arc-Flags-Query. Beschreiben Sie die notwendigen Änderungen in Pseudo-Code.

(e) Zeigen Sie die Korrektheit Ihres Algorithmus aus Aufgabe (d) wenn Sie die Vorberechnung aus Aufgabe (c) benutzen.

## **Problem 4:** Self-Bounding Bidirectional Reach-Algorithmus ***

Gegeben sei ein Graph $G = (V, E, \text{len})$. Der *reach* eines Knotens $u$ bezüglich eines kürzesten Weges $P := [s, \ldots, u, \ldots, t]$ ist definiert als $r_P(u) := \min(\text{dist}(s, u), \text{dist}(u, t))$. Der reach in $G$ von $u$ ist das Maximum aller reach-Werte von $u$ bezüglich aller kürzesten Wege durch $u$, also

$$r(u) := \max\{r_P(u) \mid P \text{ ist ein kürzester Weg der } u \text{ enthält}\}.$$

Studieren Sie den Self-Bounding Bidirectional Reach-Algorithmus aus der Vorlesung.

(a) Zeigen Sie, dass der kürzeste Weg unter Benutzung des Abbruchkriteriums des normalen bidirektionalen Dijkstra-Algorithmus nicht notwendigerweise im Suchraum enthalten ist.

(b) Zeigen Sie die Korrektheit des Abbruchkriteriums aus der Vorlesung.