# Algo2 Altklausuren

Stand: WS2019/20

## Aufgabe 1

8x vorgekommen
Beschreiben Sie jeweils eine Lösung für das Union-Find-Problem mit Laufzeit

1. $O(\log\ n)$ (amortisiert) für UNION und O(1) für FIND
2. O(1) für UNION und O(log$n$) für FIND

wobei $n$ die Anzahl der ELemente ist. Begründen Sie in beiden Fällen die entsprechenden Laufzeiten.

## Aufgabe 2

8x vorgekommen
Entwickeln Sie eine Datenstruktur zur Speicherung von $n$ Schlüsseln aus dem Universum $\{1, ..., N\}$(wobei $n << N$), die eine Zugriffszeit von O(1) garantiert. Sie dürfen dabei $O(n^2)$ Speicherplatz verwenden.

### Zusatzfrage

5x vorgekommen
(Perfektes Hashing) Verbessern Sie die Datenstruktur aus Aufgabe , so dass nur noch Speicherplatz $O(n)$ benutzt wird.

## Aufgabe 3

7x vorgekommen
Beschreiben Sie die Technik der amortisierten Analyse einer Folge von Operationen auf einer Datenstruktur $D$. Demonstrieren Sie diese Technik am Beispiel einer Folge von Increment-Operationen auf einem binären Zähler.

## Aufgabe 4

7x vorgekommen
Sei $G$ ein planarer Graph mit $n$ Knoten und $m$ Kanten. Folgern Sie aus dem Satz von Euler, dass $m \leq 3n - 6$ und dass $G$ einen KNoten vom Grad$\leq 5$ besitzt.

### Zusatz

1x vorgekommen
Zeigen Sie, dass für bipartite planare Graphen $m \leq 2n - 4$ gilt.

# Aufgabe 5

5x vorgekommen
Das $Split-Find$-Problem ist wie folgt definiert: Verwalte eine Einteilung der Zahlen $\{1, ..., n\}$ in disjunkte Intervalle, die am Anfang nur aus dem Intervall $[1, n]$ besteht, unter folgenden Operationen:

- $FIND(i)$: liefert das Intervall, das die Zahl $i$ enthält.
- $SPLIT(i)$: ersetze das Intervall $[a, b]$ und $[i + 1, b]$

Entwickeln Sie eine Datenstruktur die jede FIND-Operation in Zeit O(1) und jede Folge von SPLIT-Operationen möglichst effizient unterstützt.

# Aufgabe 6

2x vorgekommen
Geben Sie den Algorithmus von Dijkstra im Pseudo-Code an und analysieren Sie die Laufzeit.

1x vorgekommen
Geben Sie den Algorithmus von Dijksatra im Pseudo Code an und analysieren Sie die Laufzeit – FÜR Dijkstra mit Fibonachi Heap UND Binärheap. Geben Idee an: Wie müsste man den Code ändern um Probleme (z.B. keine negativen Zyklen Erkennung) zu beheben.

# Aufgabe 7

2x vorgekommen
Geben Sie ein planaren Graphen an, der verschiedene planare Einbettungen besitzt. Geben Sie die entsprechenden Einbettungen an. Für welche Graphen ist die planare Einbettung eindeutig?

Geben Sie ein planaren Graphen an, der verschiedene planare Einbettungen besitzt. Geben Sie die entsprechenden Einbettungen jeweils durch Auflistung der Face-Zyklen an. Für welche Graphen ist die planare Einbettung eindeutig?

# Aufgabe 8

2x vorgekommen
Geben Sie einen effizienten Algorithmus zur Berechnung dere *starken Zusammenhangskomponenten* eines gerichteten Graphen an.

Geben Sie den Pseudo-Code für einen effizienten Algorithmus zur Berechnung der starken Zusammenhangskomponenten eines gerichteten Graphen und analysieren Sie die Laufzeit.

# SS 2004

Im SS 2004 gab es ganz andere Fragen, die aber alle nicht noch mal vorgekommen sind:

Fachbereich IV - Abteilung Informatik
Algorithmen und Datenstrukturen
Universität Trier

Prof. S. Näher Tel.: (0651)201-3275

Ausgewählte Kapitel aus „Algorithmen und Datenstrukturen“

Sommersemester 2004

# Klausur

**Aufgabe 1:** **(5 Punkte)**
Verwenden Sie DFS (Tiefensuche) zur Berechnung einer topologischen Sortierung eines Graphen. Ihr Algorithmus sollte für zyklische Graphen einen Fehler melden. Geben Sie den vollständigen Pseudo-Code an.

**Aufgabe 2:** **(5 Punkte)**
Sei $G = (V, E)$ ein gerichteter Graph und $s \in V$ ein Knoten. Verwenden Sie BFS (Breitensuche), um für jeden Knoten $v$ die minimale Länge (Anzahl der Kanten) eines Pfades von $s$ nach $v$ zu berechnen.

**Aufgabe 3:** **(5 Punkte)**
Wie kann man das Problem aus Aufgabe 2 effizient lösen, wenn jeder Kante $e \in E$ eine *nicht-negative* reelle Zahl $c[e]$ als Kosten zugeordnet ist und für jeden Knoten $v$ jeweils die Kosten eines billigsten Pfades von $s$ nach $v$ berechnet werden sollen.

**Aufgabe 4:** **(5 Punkte)**
Wie lösen Sie das Kürzeste-Wege-Problem aus Aufgabe 3, wenn auch negative Kosten erlaubt sind?

**Aufgabe 5:** **(5 Punkte)**
Erklären Sie die Grundidee des in der Vorlesung behandelten Algorithmus zur Berechnung eines maximalen Matchings eines bipartiten Graphen.

**Aufgabe 6:** **(6 Punkte)**
Geben Sie jeweils eine Lösung für das Union-Find-Problem mit Laufzeit
a) $O(\log n)$ (amortisiert) für UNION und $O(1)$ für FIND
b) $O(1)$ für UNION und $O(\log n)$ für FIND
c) $O(1)$ für UNION und $O(\alpha(m, n))$ (amortisiert) für FIND.
Dabei ist $n$ die Anzahl der Elemente und $m \geq n$ die Zahl der Find-Operationen.

**Aufgabe 7:** **(4 Punkte)**
Sei $G = (V, E)$ ein ungerichteter Graph. Verwenden Sie Union-Find zum Aufbau einer Datenstruktur zur Beantwortung von Anfragen $SAME\_COMPONENT(v, w)$, die genau dann *true* liefern, wenn $v$ und $w$ in derselben Zusammenhangskomponente von $G$ liegen. Wie lange dauert der Aufbau der Struktur und was kostet die Beantwortung einer Frage?

**Aufgabe 8:** **(5 Punkte)**
Definieren Sie das Union-Split-Find-Problem und erklären Sie, wie eine Lösung dieses Problems zur Implementierung von Warteschlangen über einem beschränkten Universum verwendet werden kann.