| |
|---|
| Name: |
| Vorname: |
| Matrikelnummer: |

Karlsruher Institut für Technologie
Institut für Theoretische Informatik

Prof. Dr. P. Sanders

ID 5

14.3.2014

# Klausur Algorithmen I

| | | |
|---|---|---|
| Aufgabe 1. | Kleinaufgaben | 16 Punkte |
| Aufgabe 2. | $k$-Cores | 10 Punkte |
| Aufgabe 3. | Selektieren und Sortieren | 8 Punkte |
| Aufgabe 4. | Palindrome | 11 Punkte |
| Aufgabe 5. | Minimale Spannbäume | 8 Punkte |
| Aufgabe 6. | Hashing mit linear Probing | 7 Punkte |

Bitte beachten Sie:

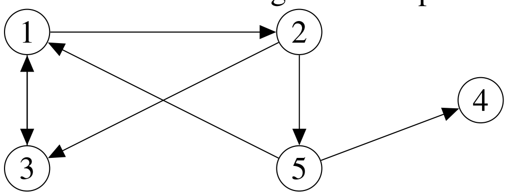
- Als Hilfsmittel ist nur **ein** DIN-A4-Blatt mit Ihren **handschriftlichen** Notizen zugelassen.
- **Schreiben** Sie auf **alle** Blätter Ihren Namen und Ihre Matrikelnummer.
- Merken Sie sich Ihre Klausur-ID 5 für den Notenaushang.
- Die Klausur enthält 15 Blätter.
- Die durch Übungsblätter gewonnenen Bonuspunkte werden erst nach Erreichen der Bestehensgrenze hinzugezählt. Die Anzahl Bonuspunkte entscheidet nicht über das Bestehen.

| Aufgabe | | 1 | 2 | 3 | 4 | 5 | 6 | Summe |
|---|---|---|---|---|---|---|---|---|
| max. Punkte | | 16 | 10 | 8 | 11 | 8 | 7 | 60 |
| Punkte | EK | | | | | | | |
| | ZK | | | | | | | |

| | | |
|---|---|---|
| Bonuspunkte: | Summe: | **Note:** |

**Aufgabe 1.** Kleinaufgaben [16 Punkte]

**a.** Stellen Sie den folgenden Graphen als Adjazenzfeld dar: [2 Punkte]




| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $V$ | | | | | | |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| $E$ | | | | | | | |

**b.** Nennen Sie zwei Operationen, die auf einer einfach verketteten Liste $O(1)$ Zeit, auf einem unbeschränkten Feld hingegen $O(n)$ Zeit benötigen. [1 Punkt]

(weitere Teilaufgaben auf den nächsten Blättern)

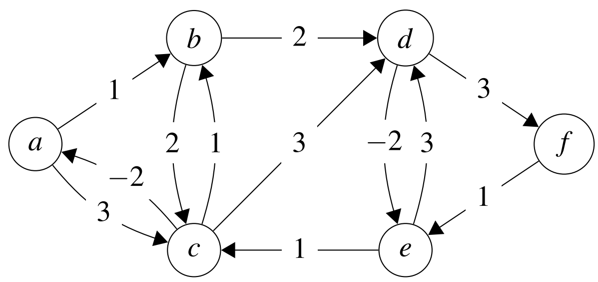
**Fortsetzung von Aufgabe 1**

**c.** Lösen Sie die drei folgenden Rekurrenzen im Θ-Kalkül:

$$\begin{aligned} V(n) &= 2014n + V(n/8), & V(1) &= 42 \\ W(n) &= n/2014 + 9W(n/8), & W(1) &= 5 \\ X(n) &= 8X(n/8) + V(n) & X(1) &= 1 \end{aligned}$$

mit $n = 8^k$ und $k \in \mathbb{N}_{>0}$. [3 Punkte]

**d.** Wieviele kürzeste Wege von $a$ nach $f$ enthält folgender gerichtete gewichtete Graph? Begründen Sie kurz. [2 Punkte]




(weitere Teilaufgaben auf den nächsten Blättern)

**Fortsetzung von Aufgabe 1**

**e.** Zeigen Sie oder widerlegen Sie, dass $3^n = \Omega(3^{2n})$ gilt. [2 Punkte]

**f.** Gegeben sei eine Vorfahrendatenbank in Form eine Folge $D$ von Paaren $(\text{ElternID}, \text{KindID})$, wobei Eltern und Kinder durch Personen-IDs aus $\mathbb{N}$ dargestellt werden.

Geben Sie einen Algorithmus an, der in erwartet $O(|D|)$ Zeit feststellt, ob die Person mit ID $a \in \mathbb{N}$ Vorfahre der Person mit ID $b \in \mathbb{N}$ ist? Korrekte Lösungen die dies noch in $O(|D| \log |D|)$ Zeit erreichen erhalten noch 2 Punkte. [3 Punkte]

(weitere Teilaufgaben auf den nächsten Blättern)

**Fortsetzung von Aufgabe 1**

**g.** Gegeben ist das folgende Parent-Array aus dem *Basis-Algorithmus* Union-Find der Vorlesung (kein Union-by-Rank, ohne Pfadkompression) :

| $v$ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| parent[v] | 4 | 2 | 2 | 4 | 2 | 1 |

Geben Sie eine Folge von *Union Operationen* des Basis-Union-Find Algorithmus an, so dass das gegebene parent-Array erzeugt wird. Geben Sie außerdem nach jeder Union Operation an, welche *link* Operation dadurch ausgeführt wird und wie sich das Parent-Array dadurch ändert.
*Hinweis:* In der Vorlesung steht $link(i, j)$ $\{$ $parent[i] := j$ $\}$. [3 Punkte]

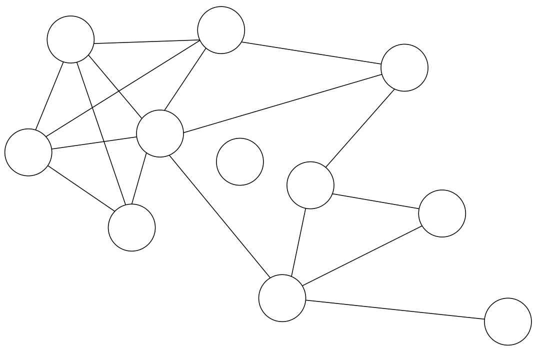
| $v$ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| parent[v] | 1 | 2 | 3 | 4 | 5 | 6 |
| parent[v] | | | | | | |
| parent[v] | | | | | | |
| parent[v] | | | | | | |
| parent[v] | | | | | | |
| parent[v] | | | | | | |
| parent[v] | | | | | | |

**Aufgabe 2.** *k*-Cores [10 Punkte]

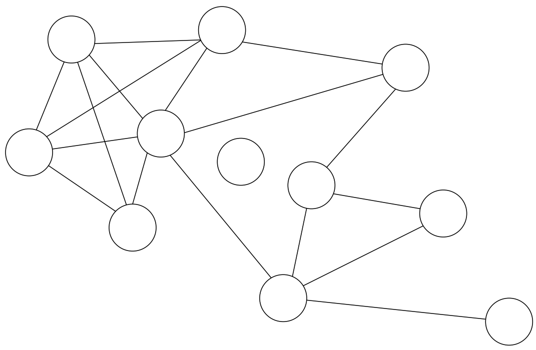
Der $k$-Core eines ungerichteten Graphen $G = (V, E)$ ist gegeben durch die größte Teilmenge $V' \subseteq V$, so dass alle Knoten im knoteninduzierten Teilgraph[1] $G' := G[V']$ einen Knotengrad $deg_{G'}(v)$ größer gleich $k$ haben. Die Core-Struktur $\mathscr{C} : V \to \mathbb{N}_{\geq 0}$ gibt für jeden Knoten $v$ die größte Nummer $x$ an, so dass $v$ noch im $x$-Core enthalten ist.

**a.** Geben Sie für folgenden Graphen die Core-Struktur $\mathscr{C}$ an. Schreiben Sie dazu in jeden Knoten $v$ den Wert $\mathscr{C}(v)$.

Für die **Lösung:**



Kopie zum **Rechnen:**



[2 Punkte]

**b.** Geben ist ein ungerichteter Graph $G = (V, E)$. Geben Sie einen Algorithmus an, der die Core-Struktur $\mathscr{C}$ in Zeit $O(|V| + |E|)$ berechnet und ausgibt. Begründen Sie kurz das Laufzeitverhalten ihres Algorithmus.
*Hinweis:* Lösungen die die Aufgabe in Zeit $\Theta((|V| + |E|) \log |V|)$ lösen erhalten 6 Punkte. Beschreiben Sie Datenstrukturen die nicht in der Vorlesung behandelt wurden genau. [8 Punkte]

[1] $G[V'] = (V', E' := \{\{u, v\} \in E \mid u, v \in V'\})$

(mehr Platz zum Schreiben finden Sie auf dem nächsten Blatt)

**Fortsetzung von Aufgabe 2**

**Aufgabe 3.** Selektieren und Sortieren [8 Punkte]

**a.** Gegeben sei die Sequenz $S = \langle 65, 28, 50, 33, 21, 56, 22, 95, 50, 12, 90, 53, 28, 77, 39 \rangle$. Bestimmen Sie mit Hilfe des *Quickselect*-Algorithmus aus der Vorlesung das Element mit Rang $k = 8$. Geben Sie jeweils für jeden Rekursionsschritt die aktuell betrachtete Teilsequenz $s$, den aktuellen Wert für $k$, sowie die Sequenzen $a := \langle e \in s : e < p \rangle$, $b := \langle e \in s : e = p \rangle$ und $c := \langle e \in s : e > p \rangle$ an. Das Pivot-Element $p$ ist für jeden Schritt vorgegeben. [4 Punkte]

| $s$ | $k$ | $p$ | Sequenzen |
|---|---|---|---|
| $\langle 65, 28, 50, 33, 21, 56, 22, 95, 50, 12, 90, 53, 28, 77, 39 \rangle$ | **8** | **28** | $a$ :<br>$b$ :<br>$c$ : |
| | ☐ | **90** | $a$ :<br>$b$ :<br>$c$ : |
| | ☐ | **39** | $a$ :<br>$b$ :<br>$c$ : |
| | ☐ | **56** | $a$ :<br>$b$ :<br>$c$ : |
| | ☐ | **50** | $a$ :<br>$b$ :<br>$c$ : |

(weitere Teilaufgaben auf den nächsten Blättern)

**Fortsetzung von Aufgabe 3**

**b.** Was ist die worst-case Laufzeit und die erwartete Laufzeit von Quickselect? [2 Punkt]

**c.** Ist Quick**sort** *in-place*? Wenn nein, mit wie viel zusätzlichem Platz kommt man in der besten Variante aus der Vorlesung aus? [2 Punkt]

**Aufgabe 4.** Palindrome [11 Punkte]

Ein *Palindrom* ist ein Wort $w$, das rückwärts geschrieben das gleiche Wort bildet. Beispiele hierfür sind 'anna', 'kayak' und 'reittier'.
Jedes Wort $w$ kann in eine *Sequenz von Palindromen* zerlegt werden: 'ababba' lässt sich in 'a|b|abba', 'aba|bb|a', 'a|bab|b|a' oder 'a|b|a|b|b|a' zerlegen, also in 3–6 Palindrome.
Wir bezeichnen die *minimale Anzahl* von Palindromen in die $w$ zerlegt werden kann mit $p(w)$.

**a.** Zerlegen Sie das Wort $w =$'abbababaabac' in eine Sequenz von $p(w)$ Palindrome und notieren Sie $p(w)$. Markieren Sie deutlich welche Zerlegung zu werten ist. [2 Punkte]

$w =$ a b b a b a b a a b a c

$p(w) = \boxed{\phantom{xx}}$

$w =$ a b b a b a b a a b a c

**b.** Entwerfen Sie einen Algorithmus, der für ein Wort $w$ die Zahl $p(w)$ in $O(n^2)$ berechnet, wobei $n$ die Länge von $w$ ist. Begründen Sie kurz seine Korrektheit und analysieren Sie die Laufzeit. Die Zerlegung selbst brauchen Sie nicht ausgeben.

*Hinweise*: Sei $w[i..j]$ das Teilwort von $w$, das den $i$-ten bis $j$-ten Buchstaben enthält. Konstruieren Sie zuerst ein Array $L[i, j]$, das angibt, ob $w[i..j]$ ein Palindrom ist.
Lösungen in $\omega(n^2)$ geben höchstens 5 Punkte. [9 Punkte]

(mehr Platz zum Schreiben finden Sie auf dem nächsten Blatt)

**Fortsetzung von Aufgabe 4**

**Aufgabe 5.** Minimale Spannbäume [8 Punkte]

**a.** Betrachten Sie zuerst zwei Spannbäume $T_1, T_2 \subseteq E$ in einem ungerichteten Graphen $G = (V,E)$. Zeigen Sie, dass es für jede Kante $e_1 \in T_1$ eine Kante $e_2 \in T_2$ gibt, so dass sowohl $(T_1 \setminus \{e_1\}) \cup \{e_2\}$ als auch $(T_2 \setminus \{e_2\}) \cup \{e_1\}$ ein Baum ist. [4 Punkte]

**b.** Nennen und *beweisen* Sie die Schnitteigenschaft für minimale Spannbäume. [4 Punkte]

**Aufgabe 6.** Hashing mit linear Probing [7 Punkte]

Wir betrachten in dieser Aufgabe Hashtabellen mit $n$ Buckets und zugehörigen Hashfunktionen,

$$h_n(x) = x \text{ MOD } n\,.$$

Beispielsweise ist $h_7(42) = 0$. Zur Kollisionsauflösung wird lineare, zyklische Suche angewendet. Folgende Hashtabelle hat die Größe $n = 10$ und Hashfunktion $h_{10}$:

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 99 | 10 | | 43 | 33 | 35 | 63 | | | 49 |

**a.** Sei nun eine leere Hashtabelle mit $n = 10$ und Hashfunktion $h_{10}$ gegeben.
Geben Sie eine Folge von *insert*-Operationen an, so dass die Tabelle nach Ausführen dieser Operationsfolge den obigen Zustand hat. Wie viele Lesezugriffe auf das Array werden bei Ihrer Operationsfolge ausgeführt? [2 Punkte]

(Teilaufgaben **b.** und **c.** auf dem nächsten Blatt)

**Fortsetzung von Aufgabe 6**

**b.** Geben Sie für eine leere Hashtabelle der Größe $n$ mit Hashfunktion $h_n$ eine Folge von zuerst $n$ **verschiedenen** *insert* Operationen und anschließend $n$ **verschiedenen** *find* Operationen an, so dass jede einzelne *insert* Operationen zwar nur genau einen Lesezugriff auf die Tabelle benötigt, die Laufzeit jeder der *find* Operationen aber linear in der Tabellengröße ist. Begründen Sie kurz, warum Ihre Folge das gewünschte Verhalten liefert. [3 Punkte]

**c.** Nennen Sie zwei Vorteile von Hashing mit linearer Suche gegenüber Hashing mit verketteten Listen. [2 Punkte]

**Konzeptpapier (Abgabe freiwillig)**