Technische Universität Chemnitz Wintersemester 2001/02
Professur Theoretische Informatik
Prof. Dr. A. Goerdt, Dipl.-Inf. F. Schädlich

1. Übung Theoretische Informatik I
Lösungsvorschläge

Aufgabe 1.1:
Beweis des Handshaking-Lemma durch vollständige Induktion über die Anzahl $|E|$ der Kanten:

Sei $|E| = 0$ (keine Kante vorhanden!). Der Grad jedes Knotens ist die Anzahl der Kanten, die mit ihm verbunden sind. Wenn keine Kante da ist, ist der Grad jedes Knotens gleich Null, somit auch die Summe der Grade der Knoten gleich 0 gleich $2|E|$.

Für $|E| = k$ sei das Lemma bewiesen. Sei nun $|E| = k + 1$. Wir nehmen aus dem Graphen eine Kante (a, b) heraus (es existiert eine Kante!). Der Graph $(V, E') = (V, E \setminus \{(a, b)\})$ hat dann nur noch k Kanten, und für ihn gilt die Induktionsvoraussetzung. $2|E|$ ist um 2 größer als $2|E'|$, wir haben zu zeigen, daß die Summe der Grade in E um 2 größer ist als in E'. Für jedes von a und b verschiedene v ist der Grad von v in E gleich dem Grad von v in E', da die Kante (a, b) hier nicht mitgezählt wird. Der Grad von a ist in E um 1 größer als in E', weil hier genau die Kante (a, b) hinzukommt. Dasselbe gilt für den Grad von b. Die Summe der Grade aller Knoten erhöht sich beim Übergang von E' zu E tatsächlich um 2, und das war zu beweisen.

Anwendung auf das konkrete Beispiel: Hier handelt es sich um einen vollständigen Graphen (jeder gibt jedem die Hand) von n Knoten (Besuchern). Der Grad jedes Knoten ist $n - 1$ (jeder Besucher gibt jedem, außer sich selbst, die Hand. Die Summe der Grade der n Knoten ist damit $n(n - 1)$, und das ist die Summe der Anzahlen der Handschläge jedes Mitarbeiters. Hinweis: Jedes Händeschütteln ist hier doppelt gezählt, für jeden der beiden beteiligten Mitarbeiter. Die Anzahl der Handschläge auf der Party, nach der nicht gefragt ist, ist halb so groß: $n(n - 1)/2$. Es wird unterstellt, daß nur Mitarbeiter die Party besucht haben.

Aufgabe 1.2:
Jeder Knoten u ist von sich selbst aus erreichbar: (u) ist ein Weg der Länge 0 von u nach u. Wenn es einen Weg $(u_1, u_2, \ldots, u_n)$ von $u_1 = u$ nach $u_n = v$ gibt, ist $(u_n, u_{n-1}, \ldots, u_1)$ ein Weg von v nach u. Wenn $(u_1, u_2, \ldots, u_n)$ ein Weg von $u_1 = u$ nach $u_n = v$ und $(v_1, v_2, \ldots, v_m)$ ein Weg von $v_1 = v$ nach $v_m = w$ ist, ist $(u_1, u_2, \ldots, u_n, v_2, v_3, \ldots, v_m)$ ein Weg von u nach w.

Für gerichtete Graphen gelten die Beweise für Reflexivität und Transitivität unverändert, die Symmetrie ist dort nicht gegeben. Sei beispielsweise $G = (\{a, b\}, \{(a, b)\})$. Hier gibt es einen Weg von a nach b, aber nicht von b nach a.

Aufgabe 1.3:

a) Adjazenzlistendarstellung:

```
+-+
|1| --> 4 --> 2 --> NIL
+-+
|2| --> 5 --> NIL
+-+
|3| --> 5 --> 6 --> NIL
```

```
+-+
|4| --> 2 --> NIL
+-+
|5| --> 5 --> NIL
+-+
|6| --> NIL
+-+
```

Adjazenzmatrix:

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Array:

```
  1   2   3   4   5   6   7   8
+---+---+---+---+---+---+---+---+
|4:7|5:0|5:8|2:0|4:0|0:0|2:0|6:0|
+---+---+---+---+---+---+---+---+
   |       |              ^   ^
   |       |              |   |
   |       +------------- | --+
   |                      |
   +----------------------+
```

b) Schritt 0: Alle Knoten weiß. Grauschlange leer. Suchbaum leer.

Schritt 1: Ausgangsknoten 3 wird grau. Grauschlange= (3). Suchbaum besteht aus Startknoten 3 mit Tiefe 0.

Schritt 2: Die Nachfolger 5 und 6 von 3 werden grau mit Tiefe 1, Suchbaum erhält die Kanten von 3 nach 5 und 6. 3 wird schwarz und aus der Grauschlange entfernt, Grauschlange= $(5, 6)$.

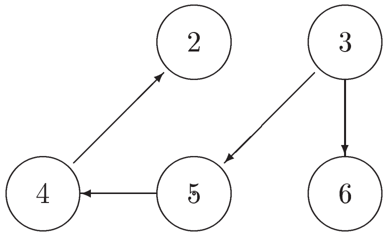
Schritt 3: Der Nachfolger 4 von 5 wird grau mit Tiefe 2, Suchbaum erhält die Kante von 5 nach 4. 5 wird schwarz. Grauschlange= $(6, 4)$.

Schritt 4: 6 hat keinen Nachfolger, wird schwarz und aus der Grauschlange entfernt. 6 bleibt Blatt im Suchbaum, wird schwarz. Grauschlange= (4).

Schritt 5: 4 hat den Nachfolger 2, dieser wird grau mit Tiefe 3. Grauschlange = (3). 4 wird schwarz.

Schritt 6: 2 hat Nachfolger 5, dieser ist schon grau. 2 wird schwarz ohne Ergänzung des Suchbaums. 1 blieb weiß mit Tiefe inf. Grauschlange leer.

Suchbaum:



Aufgabe 1.4:
Schritt 0: Alle Knoten weiß.

Schritt 1: Knoten u wird grau. Grauschlange$=(u)$. Tiefe von u gleich 0.

Schritt 2: Nachfolger t und y von u werden grau mit Tiefe 1, u wird schwarz. Grauschlange$=(t, y)$.

Schritt 3: Nachfolger w und x von t werden grau mit Tiefe 2, t wird schwarz. Grauschlange$=(y, w, x)$.

Schritt 4: y hat keinen weißen Nachfolger, wird schwarz. Grauschlange $=(w, x)$.

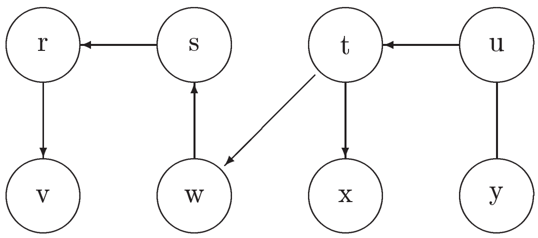
Schritt 5: Nachfolger von w sind s und x. x ist nicht weiß und bleibt unberücksichtigt. s wird grau mit Tiefe 3, Schlange$=(x, s)$.

Schritt 6: Nachfolger von x sind sämtlich nicht weiß. Schlange$=(s)$.

Schritt 7: Nachfolger von s ist r mit Tiefe 4. Schlange$=(r)$.

Schritt 8: Nachfolger von r ist v mit Tiefe 5. Schlange wird leer.

Suchbaum:



Aufgabe 1.5:
Statt der Adjazenzliste eines Knotens muss man eine Zeile in der $n \times n$ Adjazenzmatrix vollständig durchlaufen und alle n Einträge auf 0 oder 1 prüfen (Nachbarknoten?). Damit ergibt sich eine Laufzeit von $O(n^2)$.

Aufgabe 1.6:
Wir verwenden die Breitensuche:

1. Einen beliebigen ungefärbten Knoten $s \in V$ auswählen.
2. Breitensuche starten, dabei für jeden gefundenen Knoten x festhalten, ob die Weglänge zu s gerade oder ungerade ist.

   Existieren Wege gerade *und* ungerader Länge von s nach x, so ist der Graph nicht bipartit.
3. Gehe zu 1, sofern möglich, zur Betrachtung weiterer Zusammenhangskomponenten.

   Dann führt bei bipartiten Graphen jede Kante von einem Knoten mit gerader Weglänge zu einem Knoten mit ungerader Weglänge und umgekehrt. Dies kann leicht in Zeit $O(1)$ in (2.) getestet werden. Der Aufwand jeder einzelnen (modifizierten) Breitensuche ist damit linear in der Anzahl der beteiligten Kanten und Knoten. Damit ist die Gesamtlaufzeit ebenfalls linear, beträgt also $O(|V| + |E|)$.

Um tripartite Graphen $G = (V, E)$ zu erkennen, kann man (naiv) wie folgt vorgehen: Sei $G = (V, E)$ mit $V = \{1, \dots, n\}$ gegeben. Eine Tripartition $V = V_0 \uplus V_1 \uplus V_2$ kann durch einen String x der Länge n mit Einträgen der Menge $\{0, 1, 2\}$ beschrieben werden. Alle Tripartitionen werden (etwa in der Reihenfolge der zu x gehörenden Ternärzahlen) getestet. Ist $V = V_0 \uplus V_1 \uplus V_2$ eine solche Partition, testet man für jede Kante $\{u, v\} \in E$, ob u und v in verschiedenen Mengen V_0, V_1, V_2 liegen. Gilt dieses für alle Kanten $e \in E$, so ist der Graph tripartit, ansonsten wird die nächste Tripartition getestet. Ist alles erfolglos, so ist der Graph nicht tripartit. Das Verfahren hat eine Laufzeit von $\Omega(|E| \cdot 3^n)$, da es maximal 3^n Tripartitionen gibt und für jede solche maximal alle Kanten betrachtet werden.