Prof. Dr. Ir. Joost-Pieter Katoen

Sebastian Junges, Benjamin Kaminski, David Korzeniewski, Tim Quatmann

Übung 9

Hinweise:

- Die Lösungen müssen bis **Donnerstag, den 28. Juni um 16:00 Uhr** in den entsprechenden Übungskasten eingeworfen werden. Sie finden die Kästen am Eingang Halifaxstr. des Informatikzentrums (Ahornstr. 55).
- Die Übungsblätter **müssen** in Gruppen von je 3 Studierenden aus der gleichen Kleingruppenübung abgegeben werden.
- Drucken Sie ggf. digital angefertigte Lösungen aus. Abgaben z.B. per Email sind nicht zulässig.
- Namen und Matrikelnummer sowie die **Nummer der Übungsgruppe** sind auf jedes Blatt der Abgabe zu schreiben. Abgaben, die aus mehreren Blättern bestehen **müssen geheftet bzw. getackert** werden! Die **Gruppennummer muss sich auf der ersten Seite oben links** befinden.
- **Bei Nichtbeachten der obigen Hinweise müssen Sie mit erheblichen Punktabzügen rechnen!**

Aufgabe 1 (Graphen Terminologie): (10 mal 2 Punkte = 20 Punkte)

a) Wie viele gerichtete Graphen mit genau $n \in \mathbb{N}$ Knoten gibt es? Begründen Sie Ihre Antwort kurz.

b) Wie viele ungerichtete Graphen mit genau $n \in \mathbb{N}$ Knoten gibt es? Begründen Sie Ihre Antwort kurz.

c) Wie viele einfache Pfade der Länge genau $k \in \{0, 1, \ldots, n-1\}$ hat ein vollständiger ungerichteter Graph mit $n \in \mathbb{N}$ Knoten? Begründen Sie Ihre Antwort kurz.

d) Wie viele Zykel der Länge höchstens 4 hat ein vollständiger ungerichteter Graph mit $n \in \mathbb{N}$ Knoten? Begründen Sie Ihre Antwort kurz.

e) Sei $G = (V, E)$ ein gerichteter Graph mit $G^T = (V, E')$. Betrachte den Graphen $\hat{G} = (V, \hat{E})$ mit $\hat{E} = E \cup E'$. Beweisen oder widerlegen Sie folgende Aussagen:

i) $\hat{G}$ ist symmetrisch.

ii) Falls $\hat{G}$ stark zusammenhängend ist, dann ist G oder G^T stark zusammenhängend.

iii) Falls G oder G^T stark zusammenhängend ist, dann ist auch $\hat{G}$ stark zusammenhängend.

iv) G ist schwach zusammenhängend genau dann, wenn G^T schwach zusammenhängend ist.

f) Beweisen oder widerlegen Sie folgende Aussage: Jeder knotengewichteter DAG hat genau einen kritischen Pfad.

Aufgabe 2 (Tiefensuche): (4+4=8 Punkte)

Betrachten Sie folgende Implementierung einer Tiefensuche. Beachten Sie, dass der Graph als Adjazenzmatrix gegeben ist, wobei `adj[v][w]` genau dann wahr ist, wenn es eine Kante von `v` nach `w` gibt.

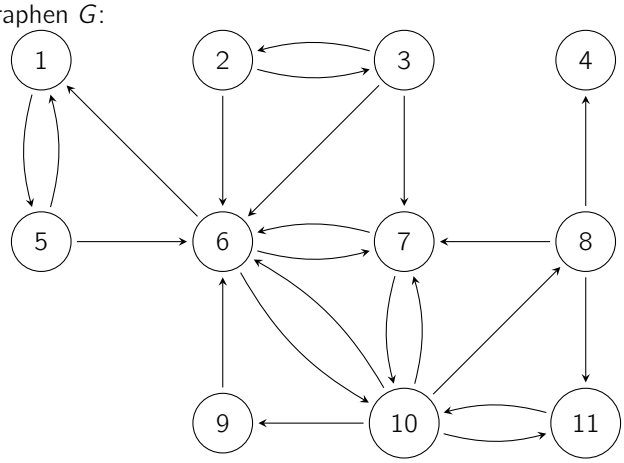
a) Bestimmen Sie die asymptotische Laufzeit der Methode `completeDFS` in Abhängigkeit der Knotenanzahl $n := |V|$ und der Kantenanzahl $m := |E|$ des gegebenen Graphen $G = (V, E)$. Nehmen Sie dazu an, dass nur **Vergleiche**, **Zuweisungen** und `print` **Anweisungen** jeweils eine Zeiteinheit benötigen. Begründen Sie ihre Antwort kurz.

b) Bestimmen und begründen Sie die asymptotische Laufzeit der Methode `completeDFS` analog zu Aufgabenteil a). Nehmen Sie dieses mal jedoch an, dass lediglich `print` **Anweisungen** eine Zeiteinheit benötigen.

```
1  void DFS(bool adj[n][n], int v, int &color[n]) {
2      print("Faerbe Knoten " + toString(v) + " grau.");
3      color[v] = GRAY;
4      for (int w = 0; w < n; w++) {
5          if (adj[v][w] == true) {
6              if (color[w] == WHITE) {
7                  DFS(adj, w, color);
8              }
9          }
10     }
11     print("Faerbe Knoten " + toString(v) + " schwarz.");
12     color[v] = BLACK;
13 }
14
15 void completeDFS(bool adj[n][n], int n) {
16     int color[n] = WHITE;
17     for (int v = 0; v < n; v++) {
18         if (color[v] == WHITE) {
19             DFS(adj, v, color);
20         }
21     }
22 }
```

Aufgabe 3 (SCCs in gerichteten Graphen): (13+3=16 Punkte)

Betrachten Sie folgenden Graphen G:



a) Wenden Sie den *Kosaraju-Sharir Algorithmus* aus der Vorlesung an, um die starken Zusammenhangskomponenten des Graphen G zu finden. Geben Sie dabei folgende Informationen an:

- In Phase 1 (Zeile 16 und 17) soll das Array `color` und der Stack `S` am Ende jeder Schleifeniteration, in der `DFS1` ausgeführt wurde, angegeben werden.
- In Phase 3 (Zeile 20 bis 23) soll das Array `color`, der Stack `S` und das Array `scc` am Ende jeder Schleifeniteration, in der `DFS2` ausgeführt wurde, angegeben werden.

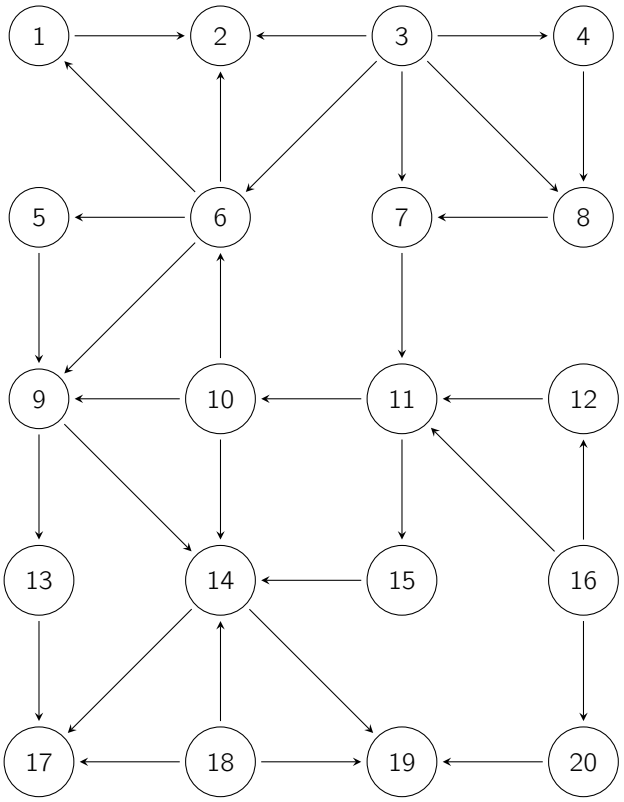
Nehmen Sie hierbei an, dass `scc` initial mit Nullen gefüllt ist und die Knoten im Graphen und in allen Adjazenzlisten aufsteigend nach ihren Schlüsselwerten sortiert sind, also der Knoten mit Schlüssel 1 vom Algorithmus als erstes berücksichtig wird usw.

b) Geben Sie den Kondensationsgraph $G{\downarrow}$ an.

Aufgabe 4 (Topologische Sortierung): (9+15=24 Punkte)

Wir betrachten den Algorithmus `topoSort` aus der Vorlesung (Vorlesung14+15, Folie 69).

a) Bestimmen Sie eine *topologische Sortierung* unter Verwendung des in der Vorlesung vorgestellten Algorithmus für den folgenden Graphen. Im gesamten Algorithmus werden Knoten in aufsteigender Reihenfolge ihrer Schlüssel berücksichtigt (d.h. die Adjazenzlisten sind aufsteigend nach Ihrem Knotenschlüssel sortiert). Als Ergebnis geben Sie an, welcher Knoten v welchen Topologiewert `topo`(v) erhält.



Ergebnis:

`topo`(v)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
`Knoten` v	**2**	**1**																		

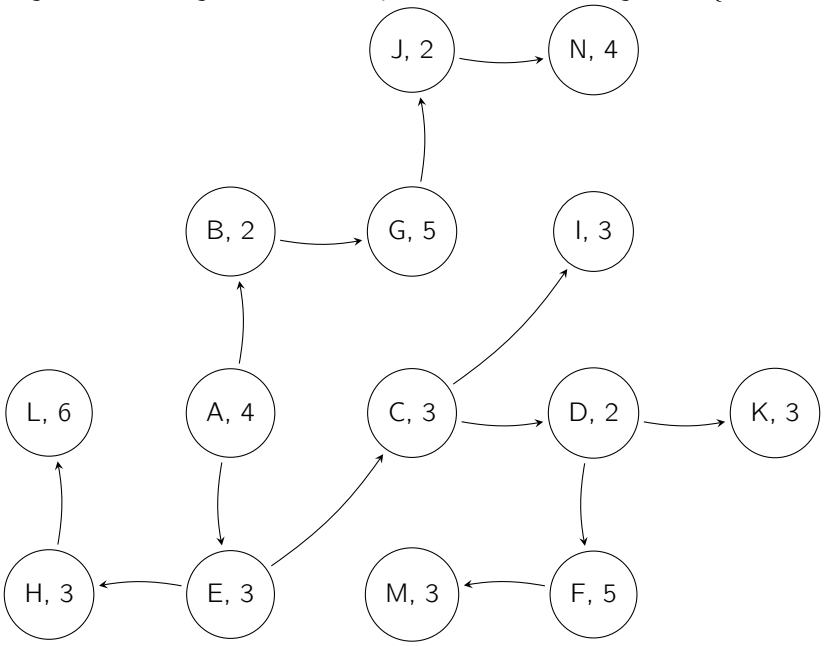
b) Geben Sie eine äquivalente Implementierung zum Algorithmus `topoSort` an, die keine Rekursion benutzt. Ihre Implementierung soll exakt die gleichen Topologiewerte generieren, wie die aus der Vorlesung bekannte Implementierung.

Hinweise:

- Sie können davon ausgehen, dass der gegebene Graph azyklisch ist.
- Sie dürfen die Funktionssignatur von `topoSort(List adj[n], int n, int &topo[n])` nicht verändern.
- Sie dürfen die Methode `List reverse(List l)` benutzen, welche die gegebene Liste `l` in umgekehrter Reihenfolge zurückgibt.

Aufgabe 5 (Kritischer Pfad): (16 Punkte)

Betrachten Sie den folgenden knotengewichteten Graphen mit Knotenmenge $V = \{A, B, \ldots, N\}$.



Nutzen Sie den in der Vorlesung vorgestellten Algorithmus um einen kritischen Pfad in obigen Graphen zu finden. Im gesamten Algorithmus werden dabei Knoten in aufsteigender alphabetischer Reihenfolge ihrer Schlüssel berücksichtigt.
Geben Sie die vom Algorithmus ermittelten frühesten Start- und Endzeitpunkte (`est` bzw. `eft`) sowie die kritische Abhängigkeit (`critDep`) für jeden Knoten an. Geben Sie außerdem an, in welcher Reihenfolge der Algorithmus die Knoten schwarz färbt. Geben Sie am Ende die ermittelte Gesamtdauer so wie den kritischen Pfad an.

Knoten	A	B	C	D	E	F	G	H	I	J	K	L	M	N
`est`														
`critDep`														
`eft`														

Reihenfolge Schwarzfärbung:

Gesamtdauer:

Kritischer Pfad:

Aufgabe 6 (Bipartite Graphen): (16 Punkte)

Ein ungerichteter Graph $G = (V, E)$ heißt **bipartit**, falls es zwei Mengen U und W gibt mit

- $U \cup W = V$
- $U \cap W = \emptyset$
- $E \subseteq \big\{\{v, v'\} \mid v \in U \text{ und } v' \in W\big\}$, d.h. es gibt keine Kanten zwischen zwei Knoten in U bzw. zwei Knoten in W.

Aufgabe: Schreiben sie eine Funktion `bool isBipartit(List adj[n])`, die für einen ungerichteten Graphen (gegeben als Adjazenzlisten) entscheidet, ob dieser bipartit ist. Ihre Funktion soll auf Breitensuche oder Tiefensuche basieren. Begründen Sie die Korrektheit Ihrer Funktion. Ein formaler Korrektheitsbeweis ist jedoch nicht notwendig.

Hinweise:

- Sie dürfen folgenden Satz aus der Globalübung verwenden:
  Satz: Ein ungerichteter Graph ist bipartit genau dann, wenn er keine Zykel ungerader Länge enthält.