Vorlesung: PD Dr. Arne Meier
Übung: Nicolas Fröhlich

Bearbeitungszeitraum 26.10.23 - 09.11.23

Datenstrukturen und Algorithmen
Hausübung 1 (Landausymbole, Graphen, Stacks & Tiefensuche)

Relevant aus dem Skript: Kapitel 1-4

Für alle Lösungen sind Begründungen anzugeben!

Organisatorisches:

1. Bearbeiten Sie dieses Übungsblatt in Gruppen von 2-4 Leuten. (Das bedeutet insbesondere, dass Sie **keine Einzelabgaben** abgeben sollen!)

2. Schreiben Sie die Namen und Matrikelnummern **aller Gruppenmitglieder** sowohl in die Python-Datei als auch auf die PDF.

3. Geben Sie Ihre Lösung **bis spätestens** am 09.11.2022 23:59 im ILIAS Kurs unter folgendem Link ab: `https://ilias.uni-hannover.de/goto.php?target=crs_175670_rcodeYPKATwaXYQ`

Wenn Sie Ihre Lösung in LaTeX erstellt haben und die PDF- und die Python-Datei zusammen in einer ZIP-Datei abgeben, bekommen Sie einen Extrapunkt.

Aufgabe 1 (8 + 2 Punkte)

a) Ordnen Sie die folgenden Funktionen anhand ihres asymptotischen Wachstums. Begründen Sie ihre Anordnung, indem Sie für jede Funktion eine passende kleinste obere Schranke angeben.

   1. $f_1(n) = 42n + 17 + 120n^2 + 23n^3$
   2. $f_2(n) = 4^{1+\log(n^2)} + 2^{4\cdot\log(2^n)}$
   3. $f_3(n) = \left(\sqrt{3n} + \sqrt{12n}\right) \cdot \left(-\sqrt{3n} + \sqrt{12n}\right)$
   4. $f_4(n) = \log(9n^2) + \log(2n^5)$
   5. $f_5(n) = \frac{6n^2-18n-10}{(2n+1)(n-5)}$
   6. $f_6(n) = \log\left(\sqrt{2\pi n}\left(\frac{n}{e}\right)^n\right)$

b) Beweisen Sie den folgenden Satz aus der Vorlesung:
   Satz 1.22 Seien f, g Funktionen. Falls $g \in O(f)$, dann folgt $f + g \in \Theta(f)$.

   Hinweis: f und g haben $\mathbb{N}$ als Definitions- und Wertebereich.

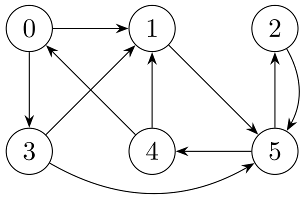
Aufgabe 2 (2 + 6 + 2 Punkte)

Sei $G = (V, E)$ ein Graph und $s \in V$. Ein *Tiefensuche-Baum von* (G, s) ist ein Wurzelbaum $T = ((V, E'), s)$, der entsteht, wenn man G beginnend in s via Tiefensuche durchläuft und genau dann eine Kante (v, w) zu T hinzufügt, wenn der Knoten w im Tiefensuchedurchlauf von G vom Knoten v aus erreicht wurde.

Das bedeutet also, $T = ((V, E'), s)$ mit

$$E' = \left\{ (v, w) \;\middle|\; \begin{array}{l} w \text{ wird im Tiefensuchedurchlauf von } G \text{ be-} \\ \text{ginnend in } s \text{ direkt von } v \text{ aus erreicht.} \end{array} \right\}$$

a) Gegeben ist der folgende Graph. Geben Sie einen Tiefensuche-Baum von $(G, 0)$ an.



b) Implementieren Sie den Algorithmus `dfs_tree` in Python, welcher einen Graphen G in Form einer Adjazenzliste und einen Startknoten s als Eingabe bekommt und einen Tiefensuchebaum von (G, s) in Form einer Adjazenzliste ausgibt. Verwenden Sie **keine Rekursion**, sondern implementieren Sie DFS mit Hilfe eines Stacks! Beschreiben Sie, wie Ihr Algorithmus arbeitet.

c) Analysieren Sie die asymptotische worst-case-Laufzeit[1] Ihres Algorithmus und geben Sie sie in $\mathcal{O}$-Notation an. Begründen Sie Ihre Antwort!

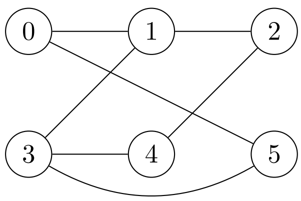
Aufgabe 3 (2 + 6 + 2 Punkte)

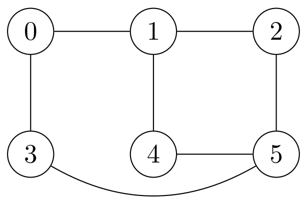
Es sei $G = (V, E)$ ein Graph. Wir nennen G *zweiteilbar*, wenn wir V in zwei Mengen A und B aufteilen können, sodass $V = A \cup B$, $A \cap B = \emptyset$ und

$$\forall (u, v) \in E : u \in A \text{ und } v \in B \text{ oder } u \in B \text{ und } v \in A$$

gilt. Jeder Knoten ist also entweder in A oder B und es gib also keine Kante zwischen den Knoten in A, bzw. B.

a) Bestimmen Sie die *Zweiteilbarkeit* der folgenden zwei Graphen. Begründen Sie ihre Antwort!





b) Implementieren Sie einen Algorithmus in Python, der einen Graphen in Form einer Adjazenzliste als Eingabe bekommt und A und B in Form zweier Listen zurückgibt (oder `None`, falls der Graph nicht *zweiteilbar* ist). Beschreiben Sie, wie Ihr Algorithmus arbeitet!

c) Analysieren Sie die asymptotische worst-case-Laufzeit[1] Ihres Algorithmus und geben Sie sie in $\mathcal{O}$-Notation an. Begründen Sie Ihre Antwort!

[1] *Hinweis: Verwenden Sie die aus der Vorlesung bekannten Laufzeiten für Stack, Queue und List (nehmen sie an, dass* `append` *konstante Laufzeit hat).*