Übungsblatt 4

Algorithmen und Datenstrukturen (WS 2013, Ulrike von Luxburg)

Präsenzaufgabe 1 (Binary Search) Nutzen Sie Binary Search (Szenario 1) auf den folgenden Arrays $A[0 \dots N-1]$, um die Zahl 5 zu suchen. Geben Sie die Werte für *low*, *mid* und *high* nach jedem Aufruf von $mid = \lfloor (high + low)/2 \rfloor$ an.

(a) 1, 2, 3, 4, 5, 5, 9, 17, 23, 36, 52, 53, 54, 62, 65, 68

(b) 1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6

(c) Finden Sie eine möglichst kurze unsortierte Eingabesequenz, welche die 5 enthält, aber in welcher die Suche nach 5 fehlschlägt.

Präsenzaufgabe 2 (Graphen-Repräsentation) Sei $G = (V, E)$ ein ungewichteter Graph mit $n = |V|$ Knoten und $m = |E|$ Kanten einmal gegeben als Adjazenzmatrix $A : V \times V \to \mathbb{R}$, sowie einmal als Adjanzenzlisten L_i für $i \in V$. Beschreiben Sie formal, woran man an der Adjazenzmatrix bzw. den Adjanzenzlisten erkennt, dass G die folgenden Eigenschaften hat, und bestimmen sie die Laufzeit in Abhängigkeit von n und m zur Ermittlung dieser Eigenschaft.

(a) G enthält eine Kante von i nach j

(b) G ist ungerichtet, oder gerichtet mit $(i, j) \in E \Leftrightarrow (j, i) \in E$

(c) G ist schleifenfrei ("Schleife" = Kante eines Knotens zu sich selbst)

(d) G hat maximalen (Ausgangs-)Grad $\leq c$ für ein gegebenes $c \in \mathbb{N}$

Präsenzaufgabe 3 (Graphen) G sei schleifenfrei, ungerichtet und ungewichtet. Beweisen oder widerlegen Sie:

(a) Jeder kürzeste Pfad in G ist immer einfach (d.h., kein Knoten wird mehrfach besucht).

(b) Der Durchmesser $diam(G)$ ist höchstens $n - 1$ (der Durchmesser von G ist die maximale Länge irgendeines kürzesten Pfades in G).

(c) Ist G ein Baum, so ist jeder Pfad einfach.

(d) Wenn G keine Zyklen enthält, so ist G ein Baum.

(e) Die Summe aller Knotengrade in G ist gleich m.

(f) Wenn G ein vollständiger k-närer Baum ist ($k \geq 2$), so gilt $diam(G) \leq 2 \lceil \log_k n \rceil$.

(g) G hat höchstens $n(n-1)/2$ Kanten.

(h) Es existiert kein solcher Graph G, in dem jeder Knoten zu jedem anderen mit einem Pfad der Länge 2 verbunden werden kann.

Präsenzaufgabe 4 (DFS/BFS)

(a) DFS und BFS nutzen die Adjazenzlistendarstellung des Graphen. Wie würde es sich auf die Laufzeit auswirken, wenn stattdessen die Adjazenzmatrixdarstellung genutzt würde?

(b) Sei G ein vollständiger binärer Baum mit n Knoten. Wie groß ist der Speicherplatzbedarf für eine bestmöglich modifizierte DFS gestartet an der Wurzel? Wie groß für BFS?

Hausaufgaben zum 4. Dezember, 9:00 (Vorlesungsbeginn).

Aufgabe 1 (Binary Search ($\frac{1}{2}$+1+$\frac{1}{2}$+1)) Diese Aufgabe beschäftigt sich mit Binary Search für Szenario 1 (Folien ≈ 17ff).

(a) Modifizieren Sie Zeile 4 des Pseudocodes zu `while(low < high)` und lassen Sie den Rest unverändert. Funktioniert der Algorithmus weiterhin? Beweisen oder widerlegen Sie.

(b) Modifizieren Sie den Code so, dass er für *absteigend* sortierte Eingabedaten funktioniert.

(c) Beweisen Sie formal die Terminierung ihrer Modifikation in (b).

(d) Beweisen Sie formal die Korrektheit ihrer Modifikation in (b).

Für die Lösungen genügt es, wenn Sie lediglich die Abweichungen von den Folien beschreiben.

Aufgabe 2 (Färbung ($\frac{1}{2}$+$\frac{1}{2}$+$\frac{1}{2}$+1+3+2+1+1+$\frac{1}{2}$+1)) Sei $G = (V, E)$ ein ungerichteter Graph und $c_k : V \to \{1, \ldots, k\}$ eine Abbildung derart, dass $i \sim j \Rightarrow c_k(i) \neq c_k(j)$. Jeder Graph für den eine solche Abbildung existiert heißt k-färbbar, und c_k eine k-Färbung.

(a) Beweisen Sie:

(i) Ein Graph ist 1-färbbar genau dann wenn er keine Kanten enthält.

(ii) Wenn ein Graph k-färbbar ist, so ist er auch $(k+1)$-färbbar.

(iii) Jeder schleifenfreie Graph ist n-färbbar.

(b) Im Folgenden sei G bipartit (aber möglicherweise unzusammenhängend):

(i) Beweisen Sie, dass ein Graph genau dann bipartit ist wenn er 2-färbbar ist.

(ii) Geben Sie den Pseudocode eines Algorithmus an, der eine 2-Färbung von G findet.

(iii) Wieviele verschiedene 2-Färbungen für G gibt es?

(c) Betrachten Sie eine beliebige Landkarte, in welcher jedes Land eine zusammenhängende Teilfläche einnimmt. Stellen Sie eine Vermutung auf, wieviele Farben ℓ minimal notwendig sind (unabhängig von der Anzahl Länder), um eine beliebige solche Landkarte so einzufärben, dass direkt benachbarte Länder stets verschiedene Farben haben. Dann suchen Sie im Internet nach einer Lösung hierfür.

(i) Erstellen Sie einen Graphen H, der für jeden Hamburger Bezirk (**A**ltona, **B**ergedorf, **E**imsbüttel, **H**arburg, **M**itte, **N**ord, **W**andsbek) einen Knoten enthält, und diese mit einer Kante verbindet, wenn die Bezirke aneinandergrenzen.

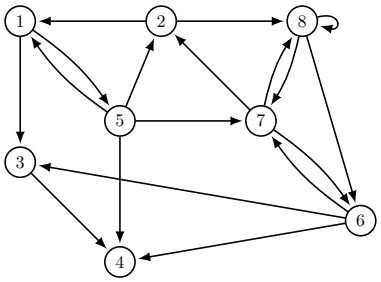
(ii) Geben Sie für H eine gültige 3-Färbung an.

(iii) Warum widerspricht (ii) nicht der Tatsache, dass "ℓ Farben minimal sind"?

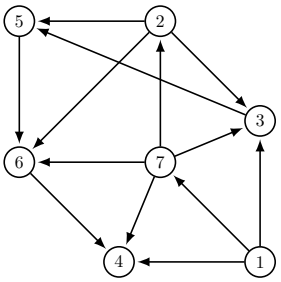
(iv) Konstruieren Sie eine Landkarte, die nicht 3-färbbar ist.

Aufgabe 3 (DFS/BFS (1+1+1+1+1+1)) Gegeben sind folgende zwei Graphen:

$G_1 :=$



$G_2 :=$



Nehmen Sie im Folgenden stets an, dass die Adjazenzlisten aufsteigend anhand der Knoten-Ids sortiert sind. Geben Sie für beide Graphen ihre Knoten-Ids in der Reihenfolge an, ...

(a) ..., in der sie bei einer Tiefensuche gestartet an 1 grau gefärbt werden.

(b) ..., in der sie bei einer Tiefensuche gestartet an 1 schwarz gefärbt werden.

(c) ..., in der sie bei einer Breitensuche gestartet an 1 erstmals besucht werden.

(d) Entscheiden Sie für jeden Graphen, ob eine mit ihm konsistente topologische Sortierung existiert. Geben Sie eine solche an, oder beweisen Sie warum keine existiert.

(e) Sind die möglicherweise in Aufgabe (d) gefundenen topologischen Sortierungen eindeutig? Beweisen oder widerlegen Sie.

(f) Geben Sie die starken Zusammenhangskomponenten der Graphen an.

Aufgabe 4 (Tron vs. MCP (3+1+1)) Stellen Sie sich vor Sie seien *Tron* und bekämpfen das bösartige "Master Control Program" (MCP). Dabei handelt es sich um ein aus vielen Modulen bestehendes Hardware-/Software-Geflecht mit komplizierten Abhängigkeiten der Module untereinander. Dieses Netzwerk können wir als gerichteten Graphen darstellen, indem wir eine gerichtete Kante von Modul i nach Modul j definieren, wenn j durch i direkt beeinflusst wird. Somit ist ein Modul k direkt *oder indirekt* von i abhängig, falls ein gerichteter Pfad von i nach k existiert.
Um sich aus der virtuellen Realität zu befreien, müssen Sie das gesamte Netzwerk eliminieren. Dazu können Sie beliebige einzelne Module infiltrieren. Sobald ein Modul infiltriert ist, können Sie dieses sowie alle direkt oder indirekt von ihm abhängigen Module auf einen Schlag eliminieren.
Damit Ihr Vorhaben nicht erkannt wird, müssen Sie allerdings eine *minimale* Anzahl von Modulen ermitteln und infiltrieren, um dann auf einen Schlag das *gesamte* MCP auszuschalten.

(a) Entwerfen Sie einen Algorithmus, der das gesamte MCP mit einer minimalen Anzahl infiltrierter Module eliminiert. Geben Sie ihn in "High-Level"-Pseudocode an, wobei Sie auf alle aus der Vorlesung bekannten Algorithmen zurückgreifen dürfen.

(b) Begründen Sie, warum das *gesamte* MCP eliminiert wird.

(c) Begründen Sie, warum die ermittelte Anzahl Module *minimal* ist.

Freiwillige Zusatzaufgabe (99 Bonuspunkte) Wir haben bereits bei den "leichten" Modifikationen von Merge-Sort gesehen, dass Rekursionen schnell eine ungeahnte Komplexität entfalten können. Die folgende Rekursion ist sogar noch viel erstaunlicher. Zerbrechen Sie sich daher (solange Sie möchten) den Kopf darüber, ob der folgende "simple" Algorithmus für jede Eingabe $n \in \mathbb{N}_{>0}$ terminiert, oder nicht. Ein korrekter Beweis für dieses Problem wird mit 99 Bonuspunkten belohnt.

```
function Coll(n)
    if n = 1 then
        return
    else
        if n ist gerade then
            Coll(n/2)
        else
            Coll(3n + 1)
        end if
    end if
end function
```