Übungsblatt 5

Abgabe am 19.11.2018

Aufgabe 1: Tron vs. MCP (3+1+1+6 Punkte)

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, werden auch alle direkt oder indirekt von ihm abhängigen Module auf einen Schlag infiltriert. Sobald alle Module infiltriert sind, ist das Netzwerk eliminiert.
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 der Module *minimal* ist.

d) Implementieren Sie Ihren Algorithmus! Im Moodle sind zwei Dateien bereit gestellt. Zum einen die Klasse `Graph`, mit der das MCP Netzwerk verwaltet wird. Es enthält eine `LinkedList<Vertex>` aller Knoten des Graphen. Jeder Knoten der Klasse `Vertex` wiederum verwaltet seine Nachbarn bzw. die gerichteten Kanten auf diese Nachbarn ebenfalls als `LinkedList<Vertex>`. Machen Sie sich zunächst mit diesen zwei Klassen und den wichtigsten Methoden vertraut, auch wenn Sie unterm Strich nur die `Getter`-Funktionen benötigen werden. Mehr dazu in den Kommentaren in der Datei `Graph.java`.

   Zum anderen ist im Moodle die Datei `Tron.java` hinterlegt. Füllen Sie mit Ihrem Algorithmus die Funktion `prepareAttackMCP(Graph MCP)`, die einen `Vector<Integer> ModulesToInfiltrate` zurück gibt, in dem die Indizes der Knoten stehen, die infiltriert werden sollen. Die Attacke selbst ist bereits implementiert. Wenn Sie das Programm ausführen, bekommen Sie eine Rückmeldung darüber, ob sie das MCP zerstört haben. In dieser Funktion können Sie auf das Graph-Objekt `MCP` zugreifen. Nebst der Funktion `prepareAttackMCP(Graph MCP)` können Sie in der Klasse `Tron` noch weitere Hilfsfunktionen hinzufügen. Lassen Sie ansonsten den Rest der Datei unberührt.

   Machen Sie außer von `LinkedList` und `Vector` nicht von anderen vorgefertigten Klassen Gebrauch. **Kommentieren** Sie ihren Code bitte ausführlich, denn die Punkte werden auf Grundlage der Implementierung und nicht des Ergebnisses vergeben. Je klarer die Tutoren Ihren Code nachvollziehen können, desto besser.

   Reichen Sie bis zum **19.11.2018, 15:45**, ihre bearbeitete Datei `Tron.java` per Mail bei Ihrem Tutor ein.

Aufgabe 2: Kürzester einfacher Pfad (1 + 4 + 2 + 2 Punkte)

Wir betrachten einen gewichteten Graphen G mit allgemeinen Kantengewichten (d.h. die Gewichte können jeden beliebigen Wert aus $\mathbb{R}$ annehmen). Wir interessieren uns für die Länge eines kürzesten einfachen Pfades in G. Erinnerung: "Einfach" bedeutet, dass kein Knoten mehrfach besucht wird.

a) Ist dieses Problem wohldefiniert (d.h. gibt es überhaupt mindestens einen kürzesten einfachen Pfad in G)? Begründen Sie Ihre Antwort.

b) Beschreiben Sie in Worten eine Adaption des Bellman-Ford-Algorithmus, um das Problem zu lösen. Argumentieren Sie die Korrektheit Ihrer Adaption.

c) Ist die worst-case Laufzeit Ihres Verfahrens polynomiell in der Anzahl an Knoten von G? Auf welcher Eingabe wird die worst-case Laufzeit erzielt?

d) Kann man Voraussetzungen an G stellen, sodass das Problem (viel) schneller gelöst werden kann?