Übung zur Vorlesung Algorithmen und Datenstrukturen

Aufgabe T20

Gegeben sei ein gerichteter Graph G und zwei seiner Knoten s und t.

Es soll festgestellt werden, ob es zwischen s und t mindestens zwei kantendisjunkte Pfade gibt. Überlegen Sie, was das genau bedeutet.

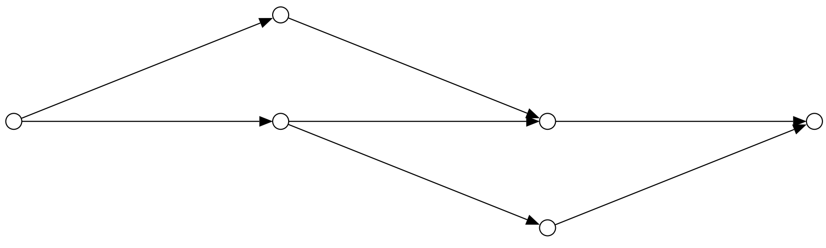
Nun entwirft ein genialer Erfinder folgenden Algorithmus, um dieses Problem zu lösen:

> *Nun, zuerst suchen wir nach irgendeinem Pfad zwischen s und t. Wenn es keinen gibt, dann gibt es natürlich auch keine zwei kantendisjunkte Pfade. Das kann ich mit Tiefensuche in linearer Zeit bewerkstelligen.*
>
> *Falls ich einen solchen Pfad finde, dann lösche ich einfach alle Kanten aus G, die auf dem Pfad liegen. Dadurch kann ich keinesfalls eine Kante aus Versehen zweimal verwenden.*
>
> *Jetzt suche ich einfach wieder nach einem Pfad von s nach t. Finde ich einen, dann gibt es zwei kantendisjunkte Pfade zwischen diesen Knoten. Wenn nicht, dann eben nicht. Die Gesamtlaufzeit ist jedenfalls linear.*

Was sagen Sie dazu?

Lösungsvorschlag:

Leider funktioniert dieses Verfahren nicht, wie folgendes Beispiel zeigt. Wird zunächst der „mittlere" Pfad von links nach rechts gewählt und gelöscht, kann dann kein zweiter Pfad mehr gefunden werden.



Aufgabe T21

	Quicksort	Heapsort	Mergesort	Insertion-Sort	Straight-Radix	Radix-Exchange
in-place?						
stabil?						
Laufzeit (worst-case)						
Laufzeit (Durchschnitt)						
vergleichsbasiert?						

Beantworten Sie die Fragen für alle Sortierverfahren. Gehen Sie davon aus, daß ein Vergleich in konstanter Zeit durchgeführt wird und die Anzahl der zu sortierenden Elemente n beträgt. Für Laufzeiten tragen Sie eine Funktion $f(n)$ in die Tabelle ein, um eine Laufzeit von $O(f(n))$ auszudrücken.

Lösungsvorschlag:

	Quicksort	Heapsort	Mergesort	Insertion-Sort	Straight-Radix	Radix-Exchange
in-place?	??	Ja	Nein	Ja	Nein	??
stabil?	Nein	Nein	Ja	Ja	Ja	Nein
Laufzeit (worst-case)	n^2	$n \log n$	$n \log n$	n^2	nw	nw
Laufzeit (Durchschnitt)	$n \log n$	$n \log n$	$n \log n$	n^2	nw	nw
vergleichsbasiert?	Ja	Ja	Ja	Ja	Nein	Nein

Aufgabe H20 (5 Punkte)

Sie haben Zuweisungen der folgenden Art:

```
y := 5*a      d := c - 3      a := c - d      c := 5      x := y - 2*c
```

Gehen Sie davon aus, daß arithmetische Operationen mit nicht initialisierten Variablen einen Fehler werfen. Wie können Sie eine gültige Folge der Zuweisungen bestimmen, die keinen Fehler wirft? Wann ist dies nicht möglich?

Lösungsvorschlag:

Es herrschen Abhängigkeiten zwischen den Variablen. Wenn in einer Zuweisung für die Variable x der Wert der Variable y auftaucht, gibt es eine Kante zwischen der Zuweisung für y und der für x.

Falls der resultierende Graph ein DAG ist, können wir das Problem mittels topologischen Sortierens lösen.

Eine korrekte Sortierung für das Beispiel ist die folgende:

```
c := 5      d := c - 3      a := c - d      y := 5*a      x := y - 2*c
```

Falls der resultierende Graph Kreise besitzt ist es nicht möglich die Operationen in eine fehlerfreie Reihenfolge zu bringen.

Aufgabe H21 (9 Punkte)

Beweisen Sie die folgenden Aussagen:

a) Ein ungerichteter Baum mit $n \geq 1$ Knoten hat genau $n - 1$ Kanten.

b) Eine Tiefensuche in einem ungerichteten Graphen liefert keine Querkanten.

c) Ergibt eine Tiefensuche in einem ungerichteten Graphen genau eine Rückwärtskante, so liefert jede Tiefensuche in diesem Graphen genau eine Rückwärtskante.

Lösungsvorschlag:

a) Dies kann man sich einfach veranschaulichen, indem man in einem Baum willkürlich eine Wurzel festlegt und alle Kanten von der Wurzel weg orientiert—auf diese Weise zeigt nun genau eine Kante auf jeden Knoten, abgesehen von der Wurzel, auf welche keine Kante zeigt. Alternativ kann man auch Induktion benutzen: Ein Baum mit einem Knoten hat null Kanten. Ein neu eingefügtes Blatt erhöht die Kantenzahl um eins.

b) Angenommen eine Tiefensuche würde eine Querkante zischen zwei Knoten u und v liefern. Das heißt v ist kein Nachfahre von u und u ist ein Nachfahre von v. Sei u der Knoten mit der geringeren discovery Zeit. Während u grau markiert ist, ist v also noch weiß markiert. Eine Tiefensuche würde die Kante von u nach v als Baumkante aufnehmen. Dies ist ein Widerspruch. Somit kann es keine Querkanten geben.

c) Stellen wir zunächst fest, daß wir uns auf zusammenhängende Graphen beschränken können: Sollte der Graph mehrere Komponenten besitzen, so kann natürlich nur in einer eine Rückwärtskante auftauchen.

   Diese Komponente besteht also aus den Baumkanten, einer Rückwärtskante und laut b) keiner Querkante. Angenommen die Komponente hat n Knoten. Laut a) gibt besteht der Tiefensuchbaum aus $n-1$ Kanten. Insgesamt gibt es also n Kanten.

   Jeder Tiefensuchbaum muss, da wir ihn als Spannbaum unseres Graphen auffassen können, $n-1$ Baumkanten besitzen. Da der Graph n Kanten besitzt, muß die verbleibende Kante eine Rückwärtskante sein.