Lehrstuhl für Informatik I
Algorithmen und Komplexität
Universität Würzburg

Würzburg, den 15. Januar 2026

Prof. Dr. Alexander Wolff
Samuel Wolf, M. Sc.

9. Übungsblatt zur Vorlesung Algorithmen und Datenstrukturen (Winter 2025/26)

Aufgabe 1 – Tiefensuche

Geben Sie für jedes der geforderten Beispiele den Graphen und das Ergebnis einer Tiefensuche in Form der resultierenden Bäume sowie der Entdeckungs- und Abschlusszeiten an. In dieser Aufgabe sind keine Selbst- oder Mehrfachkanten erlaubt.

a) Geben Sie ein Beispiel an, das folgende Behauptung widerlegt:

   Sei G ein gerichteter Graph, der einen Pfad von u nach v enthält, und sei $u.d < v.d$ das Resultat einer Tiefensuche in G. Dann folgt, dass v im Tiefensuchbaum ein Nachkomme von u ist (d.h. es gibt in diesem Baum einen u-v-Pfad). **2 Punkte**

b) Geben Sie ein Beispiel an, das folgende Behauptung widerlegt:

   Sei G ein gerichteter Graph, der einen Pfad von u nach v enthält. Für die Entdeckungs- und Abschlusszeiten jeder Tiefensuche in G gilt dann $v.d < u.f$. **1 Punkt**

c) Geben Sie ein Beispiel für eine Tiefensuche in einem gerichteten Graphen G an, in der ein Baum mit einem einzelnen Knoten u gebildet wird, obwohl u sowohl eingehende als auch ausgehende Kanten hat. **2 Punkte**

Aufgabe 2 – Wandern im Gebirge

Sie planen eine gefährliche Wanderung in einem gebirgigen Terrain, bei der Sie immer wieder über tiefe Schluchten springen müssen. Ihnen steht eine Wanderkarte zur Verfügung, auf der Berggipfel und Wegabschnitte eingezeichnet sind. Ein Wegabschnitt verbindet immer zwei Berggipfel und enthält genau eine Schlucht, die man auf dem Wegabschnitt überwinden muss. Zu jedem Wegabschnitt ist zusätzlich die Breite der zu überwindenden Schlucht in Zentimetern angegeben. Einige Wegabschnitte kann man nur in eine Richtung gehen, weil man z.B. hinunterspringen muss.

Sie starten Ihre Wanderung auf einem Berggipfel s und möchten herausfinden, wie weit man springen können muss, um jeden Berggipfel von s aus zu erreichen. (Ein Berggipfel t ist für Sie von s aus erreichbar, wenn es einen Weg von s nach t gibt, bei der Sie jede Schlucht überspringen können.)

a) Noch aus dem Sportunterricht in der Schule wissen Sie, dass Sie lediglich ℓ Zentimeter weit springen können. Sie können also über Schluchten, die breiter als ℓ sind, nicht springen.

Sie möchten herausfinden, welche Berggipfel Sie von s aus erreichen können. Modellieren Sie zunächst das Problem als ein Graphenproblem. **3 Punkte**

b) Sie möchten vor Ihrer Wanderung trainieren und Ihre Sprungweite verbessern. Hierzu möchten Sie für jeden Berggipfel t herausfinden, wie weit Sie springen können müssen, um t von s aus zu erreichen.

Geben Sie in Worten einen Algorithmus an, der dieses Problem möglichst effizient löst. Geben Sie seine Worst-Case-Laufzeit in Abhängigkeit von der Anzahl der Berggipfel und der Anzahl der Wegabschnitte an. **4 Punkte**

Aufgabe 3 – Pfade in kreisfreien Graphen

Gegeben sei ein gerichteter kreisfreier Graph $G = (V, E)$, zwei Knoten $s, t \in V$ und eine Menge $W \subset V$ von k Knoten, wobei $s, t \notin W$. Wir bezeichnen einen s-t-Pfad P als *zulässig*, wenn er durch alle Knoten in W geht. Dabei ist es egal, in welcher Reihenfolge die Knoten in W durchlaufen werden.

a) Seien P_1 und P_2 zwei zulässige s-t-Pfade. Zeigen Sie, dass P_1 und P_2 die Knoten in W in derselben Reihenfolge durchlaufen. **1 Punkt**

b) Ergänzen Sie den folgenden Algorithmus, so dass er einen zulässigen s-t-Pfad als Liste von Knoten zurückgibt, falls es solch einen Pfad gibt. Falls es keinen zulässigen s-t-Pfad gibt, soll der Algorithmus `nil` zurückgeben. Begründen Sie, warum der Algorithmus korrekt ist. **3 Punkte**

```
getPath(DirectedAcyclicGraph G, Vertex s, Vertex t, Array of Vertices W)
P = new List()
A = getVertexOrder(G, s, t, W) // Liefert ein Feld mit s, mit den Knoten in W in
    der richtigen Reihenfolge, wenn es eine solche gibt, und mit t.
for i = A.length downto 2 do
    // Ergänze Pfad P um Knoten zwischen A[i − 1] und A[i]
    // (ohne A[i − 1] aber einschließlich A[i]).
    // Gib nil zurück, wenn es von A[i − 1] nach A[i] keinen Pfad gibt.
P.Insert(A[1])
return P
```

c) Hat Ihr Algorithmus aus Aufgabenteil b) die Laufzeit $O(|V| + |E|)$? Falls nicht, beschreiben Sie in Worten, wie man diese Laufzeit durch Verbesserung Ihres Algorithmus erreichen kann. **2 Punkte**

d) Gegeben sei ein gerichteter Graph $G = (V, E)$, der nicht notwendigerweise kreisfrei ist, zwei Knoten $s, t \in V$ und eine Folge von Knoten $W = (w_1, w_2, \ldots, w_n)$. Zeigen Sie, dass im schlechtesten Fall jeder s-t-Pfad, der die Knoten in W durchläuft und dabei die gegebene Reihenfolge einhält, Länge $\Omega(|V|^2)$ hat. Beachten Sie, dass ein Pfad einen Knoten mehrfach durchlaufen kann. **2 Punkte**

Bitte geben Sie Ihre Lösungen bis **Donnerstag, 22. Januar 2026, 14:00 Uhr** einmal pro Gruppe über Wuecampus als pdf-Datei ab. Vermerken Sie dabei stets die Namen und Übungsgruppen aller BearbeiterInnen auf der Abgabe.

Grundsätzlich sind stets alle Ihrer Aussagen zu begründen und Ihr Pseudocode ist stets zu kommentieren.

Die Lösungen zu den mit PABS gekennzeichneten Aufgaben, geben Sie bitte nur über das PABS-System ab. Vermerken Sie auf Ihrem Übungsblatt, in welchem Repository (*sXXXXXX*-Nummer) die Abgabe zu finden ist. Geben Sie Ihre Namen hier als Kommentare in den Quelltextdateien an.