Karlsruher Institut für Technologie
Institut für Theoretische Informatik
Prof. Dr. Jörn Müller-Quade
Björn Kaidel, Sebastian Schlag, Sascha Witt

10. Tutorenblatt zu Algorithmen I im SoSe 2017

http://crypto.iti.kit.edu/index.php?id=799
{bjoern.kaidel,sascha.witt}@kit.edu

In Vorlesung und Übung werden neben Graphtraversierung hauptsächlich Algorithmen zum Finden kürzester Wege vorgestellt worden sein, wie der Algorithmus von Dijkstra und der Bellman-Ford-Algorithmus. Beide sollten gut verstanden sein. Bitte rechnet mit den Studenten jeweils ein Beispiel.

Aufgabe 1 (*Tiefensuche iterativ*)

Entwerfe eine nicht rekursive Tiefensuche ausgehend von einem Knoten s und gib **Pseudocode** an. Die Laufzeit $\mathcal{O}(m+n)$ darf nicht überschritten werden.

Lösung:
Die folgende nicht rekursive Implementierung der Tiefensuche beruht auf der Verwendung eines Stacks. Die Idee ist es, jeweils die Nachbarn des derzeit bearbeiteten Knotens auf einen Stack zulegen. Man initialisiert den Stack S mit dem Startknoten s. Solange der Stack S nicht leer ist, wird das vorderste Element vom Stack heruntergenommen, als gesichtet markiert und anschließend alle noch nicht gesichteten Nachbarn vorne auf den Stack gelegt. So ist sichergestellt, dass man zuerst in die Tiefe geht.

```
procedure DFS(NodeId s, Graph G)
    Stack S := ⟨s⟩
    visited := ⟨false, ..., false⟩: Array of Boolean
    while S ≠ ∅ do
        NodeId u := S.pop()
        if !visited[u] then
            visited[u] := true
            forall (u, v) ∈ E do
                if !visited[v] then
                    S.push(v)
```

Aufgabe 2 (*Kürzeste Pfade*)

Gegeben sei ein (gerichteter oder ungerichteter) zusammenhängender Graph $G = (V, E)$ mit nichtnegativen Kantengewichten $\omega : E \to \mathbb{R}^+$. Beschreibe einen effizienten Algorithmus, der für einen Startknoten s und alle Zielknoten $t \in V$ den Pfad mit den wenigsten Kanten unter allen kürzesten Pfaden zwischen s und t berechnet.

Lösung:
Eine Variante von Dijkstras Algorithmus löst das Problem. Dijkstras Algorithmus verwendet ursprünglich skalare Werte für die Kosten einer Kante. Er kann jedoch leicht modifiziert werden, sodass die Kosten einer Kante als Tupel angegeben werden können, sofern auf diesen Tupeln die Operationen Addition und Vergleich definiert sind:
Für eine Kante $\{u, v\}$ definieren wir die Kosten als Tupel $(\omega(u, v), 1)$. Addition zweier Tupel wird als komponentenweise Addition definiert, d.h. $(a, b) + (c, d) = (a + c, b + d)$. Dadurch erhalten wir sowohl die Länge als auch die Anzahl der Kanten in einem Pfad, wenn wir die Kosten über einen Pfad summieren. Der Vergleich zweier Tupel wird lexikographisch definiert, d.h. $(a, b) < (c, d)$ wenn

- $a < c$ oder
- $a = c$ und $b < d$.

Nach diesen Modifikationen können wir Dijkstras Algorithmus anwenden, um den kürzesten Pfad mit der geringsten Anzahl an Kanten zu finden.

Aufgabe 3 (*Modellieren mit Graphen*)

Mithilfe von Graphentheorie lassen sich viele Probleme aus der realen Welt modellieren und lösen. Stelle folgende Aufgabenstellungen als Graphen dar. Beschreibe die Transformation, indem du die jeweiligen Bedeutungen von Knoten und Kanten angibst. Löse das Problem mithilfe des Graphens zeichnerisch. Was muss man in dem Graphen finden, um das Ausgangsproblem zu lösen?

a) Ein Fährmann soll einen Wolf, eine Ziege und einen Kohlkopf von der linken auf die rechte Seite eines Flusses befördern. Sein kleines Boot hat aber nur Platz für ihn und ein weiteres Objekt. Außerdem frisst der Wolf die Ziege und die Ziege den Kohlkopf, wenn der Fährmann nicht dabei ist. Zum Glück mag der Wolf kein Gemüse. Wie kann der Fährmann den Wolf, die Ziege und den Kohlkopf unbeschadet übersetzen?

Betrachten Sie nun folgende Dominosteine:

1	4	1	2	1	5	1	3	4	6
4	2	4	5	6	2	2	5	5	3

b) Es dürfen nur zwei Steine mit gleicher Punktzahl aneinandergelegt werden. Den Winkel und Abstand aneinandergelegter Steine könnt ihr vernachlässigen. Ist es möglich, *alle* Dominosteine in einen geschlossenen Ring zu legen? Ansonsten begründe, warum dies unmöglich ist.

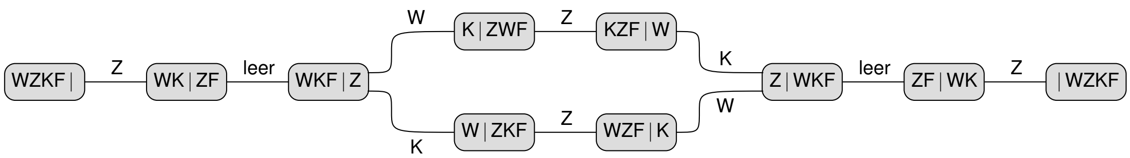
c) Ersetze im ersten Stein die 4 durch eine 6 und wiederhole den Aufgabenteil b).

d) Sollte in einem der vorherigen beiden Aufgabenteile keine Lösung möglich sein, untersuche, ob und wie durch Entfernen *eines* Steins das Problem wieder lösbar wird.

e) Welche Eigenschaft muss eine beliebige Menge von Dominosteinen besitzen, damit der Aufgabenteil b) gelöst werden kann?

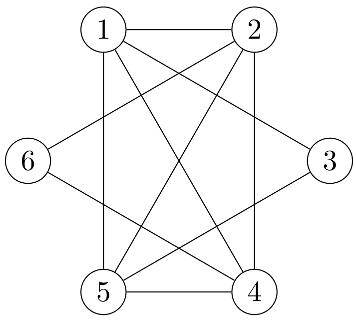
Lösung:

a) Wir betrachten alle möglichen Zustände des Problems und stellen jeden Zustand als einen Knoten dar. Kanten existieren zwischen dieses Zuständen, wenn ein Zustand durch eine Fahrt des Fährmanns in einen anderen überführt werden kann. Alle Zustandsüberführungen sind symmetrisch, daher ist der Graph ungerichtet.
Wir stellen die Zustände in der Form [L|R] dar, wobei L die linke Seite des Flusses, R die rechte Seite des Flusses beschreibt. Ungültige Zustände (bei denen die Ziege oder der Kohlkopf gefressen wird), werden nicht dargestellt. Es gibt die folgenden gültigen Zustände:
[FWZK|], [W|FZK], [FWK|Z], [FWZ|K], [FZ|WK], [WK|FZ], [K|FWZ], [Z|FWK], [W|FZK], [|FWZK]
Der Graph:



Eine Lösung kann gefunden werden, indem im Graph ein Weg von [FWZK|] nach [|FWZK] gefunden wird. Es gibt zwei Möglichkeiten, beide haben die Länge 7.

b) Die Zahlen sind die Knoten, und Dominosteine sind Kanten. Da es keine doppelten Dominosteine gibt, benötigen wir keine parallele Kantens, ein einfacher Graph reicht also. Dies ist der zugehörige Graph:

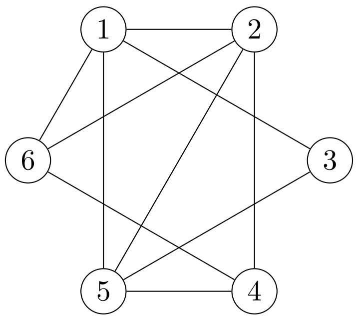


Eine mögliche Lösung ist durch den Zyklus

$$1 \to 4 \to 6 \to 2 \to 5 \to 3 \to 1 \to 2 \to 4 \to 5 \to 1$$

gegeben, wobei jeder Pfeil mit Anfang und Ende einem Dominostein entspricht.

c) Nach Umändern des ersten Steins:



Es gibt keinen solchen Kreis, da es zwei Knoten mit Grad 3 gibt, von denen man nicht mehr wegkommt und man so keinen Eulerschen Kreis bilden kann.

d) Entfernt man die Kante zwischen diesen beiden Knoten (d.h. den Stein [4|6]), so ist ein Zyklus möglich mit

$$1 \to 6 \to 2 \to 4 \to 5 \to 3 \to 1 \to 2 \to 5 \to 1.$$

e) Allgemein muss der durch die Dominosteine modellierte Graph (eventuell mit parallelen Kanten) einen Eulerschen Kreis enthalten, was genau dann der Fall ist, wenn der Graph zusammenhängend ist und alle Knoten geraden Grad haben.