Algorithmen und Datenstrukturen 1

ALGO1 · SoSe-2021 · tcs.uni-frankfurt.de/algo1

Wochenplan: Darstellung von Graphen, Breitensuche, Tiefensuche

Revision 494692b (2021-05-10)

Vorbereitung

Lies CLRS Einleitung Teil VI, Kapitel 22.1–22.4, sowie Appendix B.4–B.5 und schau das Video der Woche.

Dienstag

Aufgabe 1 (Darstellung, Eigenschaften und Algorithmen). Betrachte die Graphen in Abbildung 1. Löse die folgenden Teilaufgaben.

a) (einfach) Gib die Adjazenzlisten und Adjazenzmatrizen für die Graphen 1 und 2 an.

b) (einfach) Tiefensuche wird auf Graph 1, beginnend von Knoten 0, ausgeführt. Die Adjazenzlisten sind hierbei aufsteigend sortiert. Gib den Tiefensuchbaum, sowie die Entdeckungszeit und Endzeit an.

c) (einfach) Breitensuche wird auf Graph 1, beginnend von Knoten 0, ausgeführt. Die Adjazenzlisten sind hierbei aufsteigend sortiert. Gib den Breitensuchbaum und die Distanz zum Startknoten für alle Knoten an.

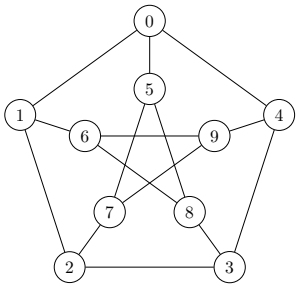
d) Gib die Zusammenhangskomponenten der 3 Graphen an.

e) Welche der 3 Graphen sind bipartit?

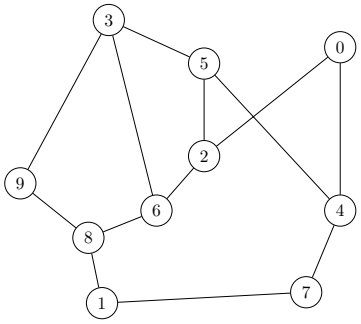
Aufgabe 2 (Buchstabenlabyrinth). Algolina und ihre kleine Schwester spielen *Buchstabenlabyrinth*. In diesem Spiel ist eine $N \times N$ Matrix gegeben, wo jeder Eintrag entweder A oder B ist. Zum Beispiel:

A	A	**A**	**B**	**A**
B	B	**B**	B	**B**
A	**B**	**A**	A	**A**
A	B	B	B	**B**
A	A	A	A	**A**

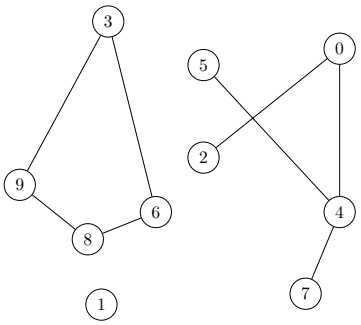
Die algorithmische Aufgabe ist es nun, einen kürzesten Pfad von oben links nach unten rechts zu finden. Die Knoten auf dem Pfad müssen dabei allerdings zwischen A und B alternieren, sprich die Knoten eines Pfades buchstabieren ABABABAB…. Der Pfad darf in jedem Schritt immer nur horizontal oder vertikal gehen, diagonale Bewegungen sind also nicht erlaubt. Im Beispiel sind die Buchstaben des kürzesten Pfads fett geschrieben.



Graph 1



Graph 2



Graph 3

Abbildung 1: Graphen für Aufgabe 1. Graph 1 wird auch als *Petersen-Graph* bezeichnet.

Da die Schwestern sich nicht sicher sind, ob sie auch tatsächlich den kürzesten Weg gefunden haben, wollen sie ein Programm schreiben, das es für sie beantwortet. Entwirf einen Algorithmus, der für eine gegebene AB-Matrix die Länge eines kürzesten Weges findet. Implementiere den Algorithmus in einer Programmiersprache deiner Wahl.

Aufgabe 3 (Tiefensuche mittels eines Stapels). Erkläre, wie Tiefensuche ohne Rekursion mit einem Stapel implementiert werden kann.

Aufgabe 4 (Wer nix weiß, sucht einen Kreis). Entwirf einen Algorithmus, der feststellt, ob ein gegebener Graph einen Kreis enthält. Wie schnell ist dein Algorithmus?

Aufgabe 5 (Anzahl kürzester Wege). Entwirf einen Algorithmus, der für einen Graphen G und zwei Knoten s, t die Anzahl der kürzesten Pfade zwischen s und t ausgibt.

Freitag

Aufgabe 6 (Labyrinthe und Gittergraphen). Ein $k \times k$ Gittergraph ist ein Graph, in dem die Knoten, wie in einem Netz, in k Zeilen mit jeweils k Knoten angeordnet sind. Kanten dürfen sich hierbei nur zwischen Knoten befinden, die in horizontaler und vertikaler Richtung adjazent sind. Siehe Abbildung 2 (a). Löse die folgenden Teilaufgaben.

a) Seien n und m die Anzahl der Knoten und Kanten in einem $k \times k$ Gittergraph. Drücke obere Schranken für n und m in asymptotischer Notation als Funktion von k aus.

Ein $k \times k$ Labyrinth ist eine quadratische Struktur, die aus k Zeilen mit jeweils k Zellen besteht. Jede Zelle wird durch vier Seiten begrenzt, und jede Seite ist entweder frei oder eine Wand. Ein Pfad im Labyrinth ist eine Sequenz F von Zellen $f_1, \ldots, f_\ell$, sodass aufeinanderfolgende Zellen f_i, f_{i+1} mit $1 \leq i < \ell$ horizontal oder vertikal adjazent sind und sich keine Wand zwischen ihnen befindet. Eine spezielle Zelle ist als Start gekennzeichnet und eine weitere als Ziel.
Der Verein „Daten- und Gartenbau“ bewertet ein Labyrinth als *schön*, falls die folgenden Voraussetzungen eingehalten werden:

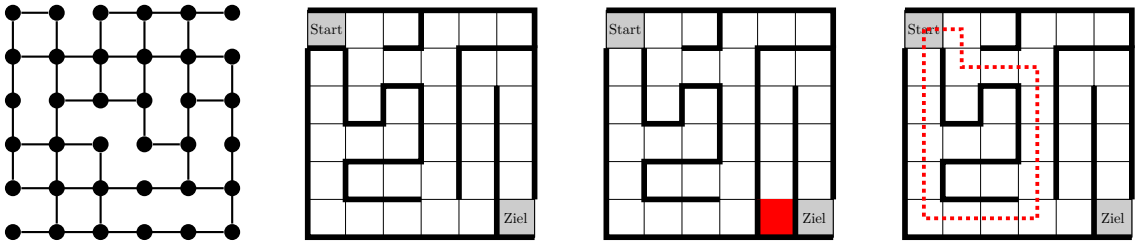
- Es gibt genau einen Weg vom Start zum Ziel.
- Es gibt einen Weg vom Start zu jeder anderen Zelle des Labyrinths.
- Es gibt keinen Weg, der im Kreis führt.

Ein Labyrinth wird als *unschön* bewertet, wenn mindestens einer dieser Punkte verletzt wird. Siehe Abbildung 2 (b)–(d).

b) Beschreibe wie man ein $k \times k$ Labyrinth als $k \times k$ Gittergraph modelliert.

c) Zeichne Abbildung 2 (b) als Gittergraph.

d) Mit dem Aufschwung der Gärtnerei im letzten Jahr wurden nun so viele Labyrinthe eingereicht, dass der Verein es nun nicht mehr stemmen kann, jedes Labyrinth von Hand zu bewerten. Entwirf einen Algorithmus, der als Eingabe einen $k \times k$ Gittergraph erhält, der ein $k \times k$ Labyrinth modelliert, und prüft, ob das Labyrinth schön ist. Zeige die Korrektheit des Algorithmus[1] und gib eine Laufzeitanalyse an, in der die asymptotische Laufzeit als Funktion von k beschreiben wird.



(a) (b) (c) (d)

Abbildung 2: (a) ein 6×6 Gittergraph. (b), (c) und (d) sind 6×6 Labyrinthe. (b) ist schön, (c) und (d) sind unschön. In (c) ist das Ende nicht erreichbar und (d) enthält einen Weg der im Kreis führt.

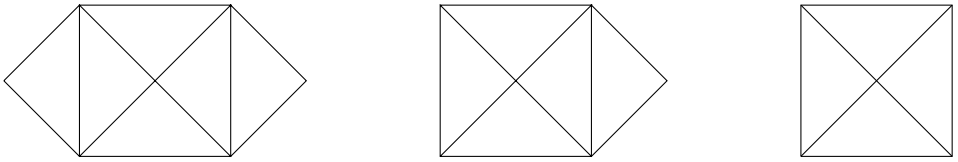
[1] Beweis, dass die Ausgabe deines Algorithmus auf allen möglichen Eingaben richtig ist.

Aufgabe 7 (Eulerkreis und Eulerpfad). Sei G ein zusammenhängender Graph mit n Knoten und m Kanten. Ein Eulerpfad in G ist ein Pfad, der alle Kanten genau ein mal enthält. Ein Eulerkreis ist ein Eulerpfad, der in demselben Knoten beginnt und endet. Löse die folgenden Teilaufgaben.

a) (schwer) Zeige, dass G einen Eulerkreis hat genau dann, wenn alle Knoten einen geraden Knotengrad haben.

b) (schwer) Zeige, dass G einen Eulerpfad hat genau dann, wenn 0 oder 2 Knoten einen ungeraden Knotengrad haben.

c) Welche dieser Zeichnungen können gezeichnet werden ohne den Stift abzusetzen? Kannst du den Stift am selben Punkt auf- und absetzen?



d) Entwirf einen Algorithmus, der in Zeit $O(n+m)$ ermittelt, ob G einen Eulerkreis hat.

e) (schwer) Entwirf einen Algorithmus, der in Zeit $O(n+m)$ einen Eulerkreis findet, falls G einen enthält.

Aufgabe 8 (Durchmesser von Bäumen). Sei T ein binärer Baum mit n Knoten. Der *Durchmesser von* T ist die Länge des längsten kürzesten Weges zwischen Knotenpaaren aus T[2].

a) Entwirf einen Algorithmus, der den Durchmesser von T in Zeit $O(n^2)$ ermittelt.

b) (sehr schwer) Entwirf einen Algorithmus, der den Durchmesser von T in Zeit $O(n)$ ermittelt.

[2] Sei $dist_G(u,v)$ die Länge des kürzesten Weges von Knoten u nach Knoten v im Graph G, dann ist der Durchmesser $\max_{u,v \in G}(dist_G(u,v))$.