Aufgabensammlung ADS-Repetitorium WS 24/25

Graphen – Graphenalgorithmen – Greedy-Algorithmen

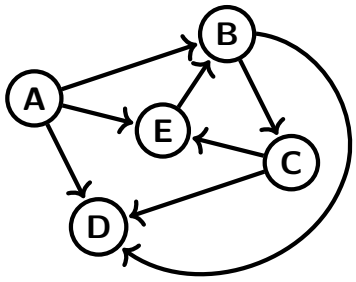


Abbildung 1: Ein gerichteter Graph für Aufgabe 1

Aufgabe 1: Repräsentation von Graphen

Gegeben sei der Graph in Abbildung 1.

(a) Enthält der Graph Kreise? Wenn ja, geben Sie einen Kreis an.

(b) Repräsentieren Sie diesen Graphen mit einer Adjazenzliste.

(c) Repräsentieren Sie diesen Graphen mit einer Adjazenzmatrix.

(d) Sei nun ein beliebiger simpler Graph G gegeben, dessen Maximalgrad $\Delta(G) = \lfloor\sqrt{|V|}\rfloor$ ist. In welcher Zeit kann man den Grad eines Knotens bestimmen oder testen, ob zwei Knoten benachbart sind, wenn G mit einer Adjazenzliste oder eine Adjazenzmatrix dargestellt ist?

(e) Beweisen oder widerlegen Sie folgende Aussage: Für einen beliebigen zusammenhängenden, einfachen Graphen $G = (V, E)$ mit $|V| \geq 2$ gilt: $\log(|E|) \in \Theta(\log|V|)$.

Aufgabe 2: Fehlende Kanten

Auf folgendem Graph in Abbildung 2 wurde eine Breitensuche ausgeführt. Dabei steht die Zahl in den Knoten für den Zeitpunkt, zu dem sie entdeckt wurden. Einige Kanten in dem Graph sind hier nicht dargestellt. Zeichnen Sie diese ein. Dabei darf jeder Knoten maximal den Grad 3 haben.

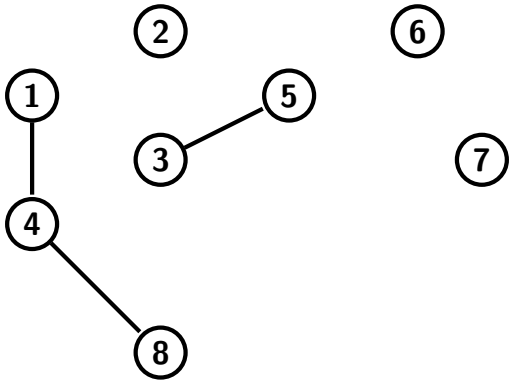


Abbildung 2: Abbildung Graphens für Aufgabe 2

Aufgabe 3: Korrektheit von Breitensuchen erkennen

In folgenden Graphen sind die π-Zeiger der Knoten nach einer Breitensuche in blau eingezeichnet. Geben

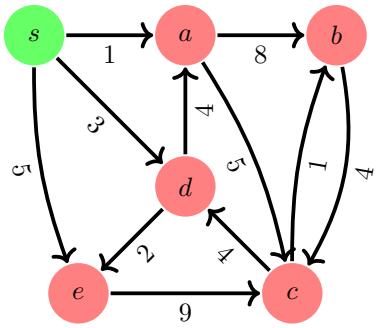
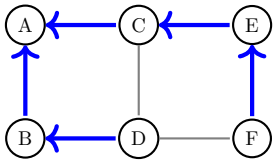


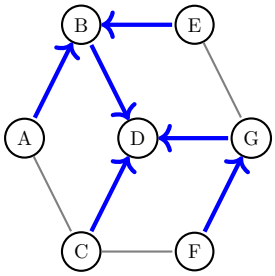
Abbildung 3: Beispiel-Graph für Dijkstras-Algorithmus.

Sie an, ob die folgenden Graphen durch eine Breitensuche entstanden sein können. Begründen Sie Ihre Antworten.

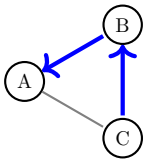
(a)



(b)



(c)



Aufgabe 4: Dijkstra's Algorithmus

Gegeben sei der Graph in Abbildung 3. Bearbeiten Sie auf diesem Graph die folgenden Aufgaben.

(a) Führen Sie Dijkstra's Algorithmus mit Startknoten s aus. Erstellen Sie dazu eine Tabelle mit drei Spalten: *Iteration*, *Schwarze Knoten*, *Graue Knoten*, zu der sie nach jeder Iteration der While-Schleife eine Zeile hinzufügen. Schreiben Sie hinter jeden Knoten in Klammern die aktuelle Distanz und den Vorgänger-Knoten.

(b) Zeichnen Sie den entstandenen Kürzeste-Wege-Baum.

(c) Seien nun negative Kantengewichte erlaubt. Verändern Sie den gegeben Graphen, sodass Dijkstra nach Beendigung kein richtiges Ergebnis liefert, obwohl der kürzeste Weg definiert ist. Warum kann dies nicht behoben werden, indem wir das minimale Kantengewicht $g < 0$ identifizieren und alle Gewichte um $|g|$ erhöhen? Dann wären alle Kantengewichte wieder positiv und wir könnten Dijkstra verwenden. Begründen Sie allgemein, warum diese Vorgehensweise nicht korrekt ist.
Vorschau: Morgen werden wir uns im Rahmen der Dynamischen Programmierung mit einem Algorithmus beschäftigen, der mit negativen Kanten zurecht kommt.

Aufgabe 5: Tiefensuche iterativ

Sie haben in der Vorlesung zwei Durchlaufstrategien für Graphen kennengelernt: die Breitensuche und die Tiefensuche. Während die Breitensuche „iterativ" mit einer Schlange funktioniert, benutzt die Tiefensuche Rekursion, um den Graphen zu explorieren. Da rekursive Algorithmen manchmal langsamer sind und außerdem durch die Größe der maximalen Rekursionstiefe limitiert ist, möchte man gelegentlich die Tiefensuche ebenfalls „iterativ" implementieren.
Hinweis: Sie müssen die Farben nicht setzen und können eine einzige flag $u.visited$ für einen Knoten u benutzen.

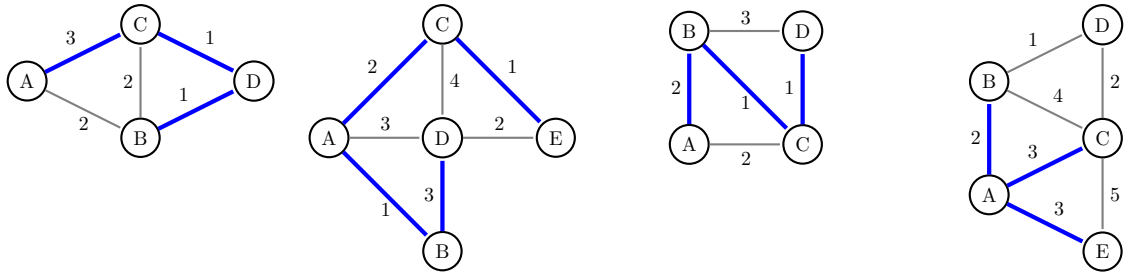
(a) Implementieren Sie die Tiefensuche mit Hilfe eines Stapels in Pseudocode. Ignorieren Sie hierfür die *time stamps* $u.d$ und $u.f$ eines Knotens u. Die Entdeckungsreihenfolge muss nicht identisch mit der rekursiven Version der Tiefensuche sein!

(b) Können Sie Ihren Pseudocode so zu modifizieren, dass Sie auch die *time stamps* korrekt setzen?

Aufgabe 6: Erkennen von minimalen Spannbäumen

Entscheiden Sie für die folgenden blau markierten Teilgraphen, ob diese minimale Spannbäume sind. Geben Sie für jeden Graphen, dessen blau markierter Teilgraph kein minimaler Spannbaum ist, einen minimalen Spannbaum an und begründen Sie Ihre Antworten.

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



Aufgabe 7: Kürzeste-Wege-Bäume und minimale Spannbäume

Die Algorithmen von Dijkstra und Jarnik-Prim gehen ähnlich vor. Beide berechnen, ausgehend von einem Startknoten s, einen Baum. Allerdings brechnet der Algorithmus von Dijkstra einen Kürzesten-Wege-Baum, während der Algorithmus von Jarnik-Prim einen minimalen Spannbaum berechnet.

(a) Geben Sie einen ungerichteten gewichteten Graphen $G = (V, E)$ mit höchstens 5 Knoten und einen Startknoten $s \in V$ an, sodass Dijkstra und Jarnik-Prim ausgehend von s verschiedene Bäume in G liefern. Geben Sie beide Bäume an.

(b) Geben Sie eine Familie von Graphen an, auf denen der Algorithmus von Jarnik-Prim asymptotisch schneller als der Algorithmus von Kruskal ist.

(c) Angenommen Sie haben einen ungerichteten, gewichteten Graphen $G = (V, E)$ gegeben und alle Gewichte sind natürliche Zahlen zwischen 1 und $|V|$. Wie könnten Sie die Laufzeit von Kruskal's Algorithmus beschleunigen?

Aufgabe 8: Modellierung als Graphen

Viele algorithmische Fragestellungen können als Graphen modelliert werden. Bearbeiten Sie folgende Punkte für jede Fragestellung:

1. Geben Sie einen Algorithmus in Worten an, der die Fragestellung als Graph modelliert. Achten Sie auf eine präzise Definition der Knoten- und Kantenmenge.
2. Mit welchem Graph-Algorithmus kann die Fragestellung auf dem Graph gelöst werden? Wie müssen die Algorithmen gegebenfalls modifiziert werden?
3. Von welchen Parametern hängt die Laufzeit ab? Können Sie die Laufzeit genau angeben?

Einige der Probleme lassen sich eventuell auch einfacher ohne Graph lösen. In dieser Aufgabe sollen sie aber explizit den Umgang mit Graphen üben.

(a) Sie arbeiten im Marketing einer Firma, die Kameras herstellt. Auf einer Veranstaltung, die von n Menschen besucht wird, bieten Sie folgende Werbekampagne an:
Sie verteilen m Einwegkameras, mit denen man je genau ein Bild schießen kann. Die Kameras sollen dazu benutzt werden, Twofies (= Bilder, auf denen genau zwei Personen abgebildet sind) zu schießen. Die Person, die am Ende der Veranstaltung mit den meisten anderen Menschen auf Twofies abgebildet ist, hat gewonnen.
Wie können Sie die Person ermitteln, die gewinnt?

(b) Das Kommunalunternehmen eines Landkreises mit n Gemeinden möchte jede Gemeinde an ein Radwegnetz anbinden. Um Kosten zu sparen, sollen die Radwege nur an den *breitesten* Straßen im Landkreis angelegt werden und Rundfahrten zwischen den Gemeinden sollen nicht möglich sein.

(c) Ein Software-Projekt besteht aus n Modulen. Jedes Modul kann von anderen Modulen abhängig sein. Ein Modul kann nur gestartet werden, falls alle Module, von denen das Modul abhängt, bereits gestartet wurden. Gesucht ist also die Reihenfolge, in der die Module gestartet werden können.

Aufgabe 9: Terminale verbinden

Sie sind Betreiber eines Routernetzwerkes, wobei R die Menge Ihrer Router darstellt. Die Router sind untereinander verbunden, sodass Ihr gesamtes Netzwerk zusammenhängend ist. Dabei muss nicht notwendigerweise jeder Router mit jedem anderen Router verbunden sein. Damit die Verbindungen funktionieren, muss jede Verbindung mit Strom versorgt werden.

Im Falle eines Stromausfalls kann jede Verbindung mit je einem teurem Notstromaggregat betrieben werden. Die Kosten für dieses Notstromaggregat sind je Verbindung verschieden.

In Ihrem Netzwerk befinden sich einige besonders wichtige Knotenrouter K. Falls der Strom ausfällt, sollen weiterhin alle Router in K miteinander verbunden sein. Alle anderen Router $R \setminus K$ dürfen, aber müssen nicht unbedingt, ans Netzwerk angeschlossen sein. Sie sind nun an einer Auswahl an Verbindungen interessiert, die möglichst günstig mit Notstromaggregaten betrieben werden können und alle Knotenrouter K verbindet.

(a) Modellieren Sie das Problem als Graph-Problem. *Tipp:* Machen Sie sich mit einer Skizze die Aufgabenstellung klar.

(b) Angenommen $|K| = 2$. Geben Sie einen effizienten Algorithmus an, der das Problem löst.

(c) Angenommen $K = R$, mit anderen Worten: Alle Router sind wichtig. Geben Sie einen effizienten Algorithmus an, der das Problem löst.

(d) Das Software-Unternehmen PISNP bietet einen Algorithmus an, der das Problem für alle K löst. Dieser Algorithmus berechnet einen Graph T, der angeblich die oben beschrieben Anforderungen erfüllt. Geben Sie einen effizienten Algorithmus an, der überprüft, ob T tatsächlich gültig ist. Ihr Algorithmus erhält als Eingabe ihr Routernetzwerk, wie in a) modelliert, sowie K und T.

Aufgabe 10: Greedy-Algorithmen

Geben Sie für jedes der folgenden Probleme einen Greedy-Algorithmus an. Denken Sie daran, die folgenden Punkte zu beweisen:

- Beweisen Sie, dass die Lösung des Greedy-Algorithmus zulässig ist.
- Beweisen Sie, dass die Lösung des Greedy-Algorithmus optimal ist, oder geben Sie ein Beispiel an, in dem der Greedy-Algorithmus nicht die optimale Lösung findet.
- Geben Sie die Laufzeit an.

(a) Sei $G = (V, E)$ ein ungerichteter Graph. Gesucht ist eine möglichst große Teilmenge U der Knoten V, sodass für keine zwei Knoten $u, v \in U$ die Kante uv in E ist.

(b) Gegeben ein Baum $T = (V, E)$, in dem jeder Knoten v einen positiven Wert $v.w$ hat. Gesucht ist ein Pfad von der Wurzel zu einem Blatt, sodass die Summe der auf dem Weg liegenden Knoten möglichst groß ist.

(c) Gegeben sind n positive Zahlen in einem Feld. Selektieren Sie $n/2$ Zahlen so, dass dass deren Summe möglichst klein ist.

(d) Gegeben eine Liste von Jobs $j_1, \ldots, j_n$ sowie die Zeiten t_i, die zum Abarbeiten des i. Jobs benötigt wird. Es stehen zwei Maschinen zur Verfügung. Verteile die Jobs so auf beide Maschinen, dass alle Jobs möglichst schnell bearbeitet wurden.

(e) Sie wollen eine Wüste durchqueren. In der Wüste sind $n + 1$ Oasen, Sie starten in Oase 0 und Ihr Ziel ist Oase n. Dazwischen sind $n - 1$ Oasen, in denen Sie Ihre Wasserflasche auffüllen können. Ihre Wasserflasche reicht für r Kilometer. Oase i und Oase $i + 1$ sind d_i Kilometer voneinander entfernt. Finden Sie eine Folge von Oasen, sodass die möglichst selten nachfüllen müssen. Sie können davon ausgehen, dass alle Distanzen zwischen den Oasen kleiner als r sind.

Aufgabe 11: Anzahl der einfachen Pfade im Graph
Verwenden Sie den Tiefensuche-Algorithmus, um die *Anzahl* der einfachen Pfade zwischen zwei Knoten in einem azyklischen, gerichteten und ungewichteten Graphen in $\mathcal{O}(|V| + |E|)$ Zeit zu berechnen. Verwendet Ihr Algorithmus das Prinzip dynamischer Programmierung oder ist er ein Greedy-Algorithmus?

Aufgabe 12: Matchings in Graphen
Sei $G = (V, E)$ ein ungerichteter Graph. Eine Teilmenge $M \subseteq E$ der Kantenmenge heißt *Matching*, wenn keine zwei Kanten in M einen Knoten gemeinsam haben. Ein Matching heißt *nicht erweiterbar*, wenn es keine Kante e in $E \setminus M$ gibt, sodass $e \cup M$ ein Matching ist.

(a) Überlegen Sie sich ein Szenario, das man als Graph modellieren und mit einem Algorithmus, der ein maximales Matching findet, lösen kann.

(b) Schreiben Sie einen Algorithmus in Pseudocode, der für einen gegebenen Graphen $G = (V, E)$ und eine Teilmenge $M \subseteq E$ bestimmt, ob M ein Matching ist.

(c) Entwickeln Sie einen Algorithmus in Pseudocode, der für einen gegebenen Graphen $G = (V, E)$ ein gültiges, nicht erweiterbares Matching berechnet.

Aufgabe 13: Graphen-Cliquen
Sei $G = (V, E)$ ein ungerichteter Graph. Eine Teilmenge $C \subseteq V$ heißt *Clique*, wenn jedes Knotenpaar $e, d \in E$ durch eine Kante verbunden ist.
Schreiben Sie einen Algorithmus, der für einen Graphen $G = (V, E)$ und eine Teilmenge $C \subseteq E$ prüft, ob C eine Clique in G ist.