



Kürzeste Pfade in Graphen

Algorithmus von Dijkstra

Benjamin Schmid

Ein Einsatzgebiet von Graphen ist die Darstellung von Karten. Oft möchte man auf diesen den kürzesten Pfad zwischen zwei Punkten bestimmen, um z.B. eine Wegbeschreibung anzeigen zu können. In diesem Skript schauen wir uns einen Algorithmus an, der auf einem gewichteten Graphen kürzeste Pfade effizient berechnen kann.

Inhaltsverzeichnis

1 Eine Analogie	2
2 Schnellere Simulation	3
3 Der Algorithmus	4
4 Dijkstra von Hand	5
5 Zusammenfassung	7
6 Lösungen	8

Kapitel 1: Eine Analogie

Wir möchten gerne auf einem gewichteten Graphen den kürzesten Pfad zwischen zwei Knoten bestimmen. Wir haben also einen Startknoten und einen Zielknoten. Es gibt meist viele verschieden mögliche Pfade, um von einem Knoten zu einem anderen Knoten zu gelangen. Von all diesen möglichen Pfaden möchten wir also denjenigen bestimmen, der am kürzesten ist. Es kann mehrere solche Pfade geben, dann möchten wir einfach einen davon finden.

Wir beginnen zunächst mit einer Analogie. Wir stellen uns vor, dass sich am Startknoten ein Ameisennest befindet. Die Ameisen möchten nun ihre Umgebung erkunden. Hierzu starten sie genau um Mitternacht und laufen entlang der Kanten im Graph umher. Alle Ameisen laufen genau gleich schnell. Jedesmal, wenn sie einen Knoten erreichen, schreiben die Ameisen zum Knoten die aktuelle Uhrzeit und von welchem Knoten sie kamen. Anschliessend teilen sie sich auf alle noch nicht begangenen ausgehenden Kanten auf¹ und laufen weiter. Sollten sich mal zwei Gruppen von Ameisen auf einer Kante begegnen, wissen sie, dass es dort nichts mehr zu erkunden gibt (die andere Gruppe hat das ja bereits erledigt) und sie gehen nach Hause.

Auf <https://smbe.ch/dijkstra> ist eine Visualisierung verlinkt, welche genau diesen Prozess abbildet.

Übung 1

Starten Sie die Visualisierung, indem Sie entweder auf „Play“ oben rechts klicken, oder mittels „Step back“ und „Step“ in Ihrem eigenen Tempo die Visualisierung durchgehen. Sie haben oben links die Möglichkeit, zwischen einem kleinen und einem grossen Graphen auszuwählen. Zu Beginn reicht der kleine Graph aus.

Beschreiben Sie, welche Elemente der Visualisierung welchen Elementen der obigen Analogie entsprechen.

Übung 2

Warum scheint es so, als ob sich die Ameisen nicht auf allen Kanten gleich schnell bewegen?

¹Dank den Spurpheromonen können die Ameisen riechen, ob eine Kante bereits von anderen Ameisen begangen wurde.

Übung 3

Die Uhrzeit über einem Knoten hat eine wichtige Eigenschaft. Erkennen Sie, welche das ist?

Begründen Sie, warum diese Eigenschaft gilt.

Übung 4

Wir sind eigentlich nur am kürzesten Pfad zwischen zwei bestimmten Knoten interessiert, doch mit den Uhrzeiten können wir sogar noch mehr bestimmen. Beschreiben Sie, zwischen welchen Knoten wir alles die Länge des kürzesten Pfads kennen.

Kapitel 2: Schnellere Simulation

Wir haben nun also ein Verfahren, welches uns erlaubt, den kürzesten Pfad von einem Startknoten zu allen anderen Knoten zu bestimmen. Um dieses aber in einem Algorithmus umzusetzen, haben wir jedoch noch ein paar Probleme. Zum einen passieren viele Dinge gleichzeitig und zum anderen simulieren wir viele Details, die gar nicht relevant sind. Aber wir haben Glück und können die Simulation der Ameisen stark vereinfachen.

Übung 5

Überlegen Sie sich, wie Sie die Simulation vereinfachen könnten, um diese Probleme zu umgehen.

Übung 6

In der Visualisierung können Sie oben rechts von „Ants“ auf „Events“ umstellen. Dies stellt die Visualisierung um, sodass nicht mehr alle Details simuliert werden und stattdessen nur die wichtigen Ereignisse.

Vergleichen Sie die Visualisierung in den Modi „Ants“ und „Events“. Welche Unterschiede gibt es? Welche Gemeinsamkeiten?

Beschreiben Sie die neuen bzw. geänderten Elemente der Visualisierung.

Übung 7

Die Visualisierung springt im „Events“-Modus jeweils zum nächsten interessanten Zeitpunkt, wenn Ameisen einen Knoten erreichen. Können Sie anhand der Zahlen über den Knoten voraussagen, welcher Knoten im nächsten Schritt erreicht wird? Können Sie voraussagen, wie sich die Zahlen verändert werden, wenn der Knoten erreicht wird?

Kapitel 3: Der Algorithmus

Wir konnten die Simulation vereinfachen und auf einige wichtige Zeitpunkte reduzieren. Es fehlt nur noch ein kleines Element, damit der Computer dies effizient ausführen kann: wir müssen jeweils wissen, bei welchem Knoten als nächstes Ameisen ankommen werden, sodass wir diesen dann verarbeiten können und die Ameisen wieder weiterschicken können.

Übung 8

Stellen Sie die Visualisierung oben rechts auf „Queue“ um.

Vergleichen Sie die Visualisierung in den Modi „Events“ und „Queue“. Welche Unterschiede gibt es? Welche Gemeinsamkeiten?

Beschreiben Sie die neuen bzw. geänderten Elemente der Visualisierung.

Um die sortierte Liste von den kommenden Knoten zu verwalten, gibt es spezielle Datenstrukturen², welche dies effizient machen können. Für den Moment können Sie einfach annehmen, dass diese existieren; die genaue Funktionsweise ist nicht wichtig. In eine solche Datenstruktur können Sie Elemente einfügen und das jeweils kleinste Element abrufen. Sie können auch bestehende Elemente aktualisieren, um so z.B. die Ankunftszeit bei einem Knoten anzupassen.

Dieser letzte Modus in der Visualisierung entspricht nun ziemlich genau dem Algorithmus, den ein Computer nutzt, um den kürzesten Pfad von einem Knoten zu allen anderen Knoten zu bestimmen. Und wenn wir den kürzesten Pfad zu allen anderen Knoten kennen, kennen wir natürlich auch den kürzesten Pfad zu unserem Zielknoten.

Den Algorithmus nennt man den „Algorithmus von Dijkstra“, nach dessen Erfinder Edsger W. Dijkstra.

Übung 9

Die Länge des kürzesten Pfades zu einem Knoten können wir direkt ablesen. Aber wie können Sie bestimmen, welche Kanten Sie nehmen müssen für den kürzesten Pfad zwischen zwei Knoten?

Übung 10

Beschreiben Sie den Algorithmus in eigenen Worten.

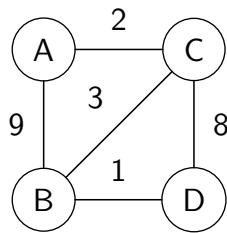
Übung 11

Stellen Sie den Algorithmus in einem Flussdiagramm dar.

²Diese nennt man eine Priority Queue.

Kapitel 4: Dijkstra von Hand

Zum Schluss möchten wir den Algorithmus noch von Hand durchspielen. Wir schauen uns dies zunächst an einem sehr kleinen Graphen an.



Wir können die folgende Tabelle nutzen, um den aktuellen Zustand des Algorithmus abzubilden.

Knoten	Distanz	Vorgänger	Fertig
A			
B			
C			
D			

Wir starten den Algorithmus von Knoten A und wir setzen die Distanz zum Startknoten auf 0.

Knoten	Distanz	Vorgänger	Fertig
A	0		
B			
C			
D			

Dann finden wir den Knoten mit der kleinsten Distanz, was der Startknoten ist, und bearbeiten alle Nachbarn.

Knoten	Distanz	Vorgänger	Fertig
A	0		✓
B	9	A	
C	2	A	
D			

Der nächste Knoten, den wir bearbeiten, ist der Knoten C. Wenn wir die Nachbarn von C bearbeiten, finden wir einen kürzeren Pfad zu Knoten B und aktualisieren die Tabelle entsprechend.

Knoten	Distanz	Vorgänger	Fertig
A	0		✓
B	9 5	A C	
C	2	A	✓
D	10	C	

Nun bearbeiten wir den Knoten C und finden einen kürzeren Pfad zu Knoten D.

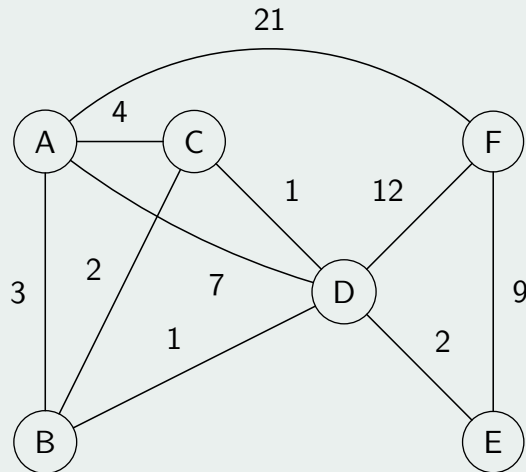
Knoten	Distanz	Vorgänger	Fertig
A	0		✓
B	9 5	A C	✓
C	2	A	✓
D	10 6	C B	

Als letzten Knoten bearbeiten wir D. Hier gibt es nicht mehr viel zu tun, da wir alle Nachbarn bereits bearbeitet haben.

Knoten	Distanz	Vorgänger	Fertig
A	0		✓
B	9 5	A C	✓
C	2	A	✓
D	10 6	C B	✓

Übung 12

Bestimmen Sie die kürzesten Pfade ausgehend von Knoten A im folgenden Graph. Führen Sie hierzu den Algorithmus von Hand aus.



Knoten	Distanz	Vorgänger	Fertig
A			
B			
C			
D			
E			
F			

Kapitel 5: Zusammenfassung

Der Algorithmus von Dijkstra kann verwendet werden, um in einem gewichteten Graphen die kürzesten Pfade von einem bestimmten Knoten zu allen anderen Knoten zu bestimmen. Dabei geht der Algorithmus ähnlich vor wie sich aufteilende Ameisengruppen, die auf dem Graphen umherlaufen. Sobald eine Ameisengruppe einen Knoten das erste Mal erreicht, ist der von diesen Ameisen genommene Pfad der kürzeste vom Startknoten zu diesem Knoten.

Dieses Vorgehen kann ein Computer effizient simulieren, indem nur die wichtigen Zeitpunkte, wenn Ameisen ankommen, betrachtet werden. Dank einer guten Datenstruktur kann effizient bestimmt werden, welcher Knoten als nächstes bearbeitet werden muss. Wenn ein Knoten bearbeitet wird, werden für alle Nachbarn berechnet, wann die Ameisen dort ankommen werden.

Der Algorithmus kann nicht nur die Distanz zu einem Zielknoten bestimmen, sondern auch den konkreten Pfad.

Kapitel 6: Lösungen

6.1. Kapitel 1: Eine Analogie

Übung 1: Der orange Knoten ist der Startknoten, oder eben das Ameisennest.

Der Text in der Box oben links vom Graphen ist die aktuelle „Uhrzeit“.

Die rot markierten Teile der Kanten sind Ameisen, die den Graphen erkunden.

Die Zahl über einem Knoten ist die Uhrzeit, zu der ein Knoten besucht wurde, oder ∞ , falls er noch nicht besucht wurde.

In der Tabelle rechts ist sowohl diese Uhrzeit als auch der Knoten, von dem die Ameisen ankamen („Previous“), für jeden Knoten aufgeführt.

Wenn zwei Ameisengruppen sich auf einer Kante treffen, wird dies farblich angezeigt.

Übung 2: Dies sieht so aus, weil die Kanten verschiedene Gewichte (= Länge) haben, aber unter Umständen eine ähnliche Länge in der Darstellung haben. Wenn z.B. eine Kante mit Gewicht 6 und eine Kante mit Gewicht 2 beide 2 cm lang sind in der Darstellung, dann scheint es so, als ob sich die Ameisen auf der zweiten Kante dreimal so schnell fortbewegen, weil sie die 2 cm in nur 2 Zeiteinheiten zurücklegen, während die Ameisen auf der ersten Kante die 2 cm in 6 Zeiteinheiten zurücklegen.

Übung 3: Die Uhrzeit über einem Knoten entspricht genau der Länge des kürzesten Pfads vom Startknoten zu diesem Knoten.

Da alle Ameisen gleich schnell laufen und die Ameisen alle Kanten parallel begehen, müssen die ersten Ameisen, die an einem Knoten ankommen, den kürzesten Pfad genommen haben. Da die Ameisen eine bereits begangene Kante nicht mehr begehen und sich nicht auf Kanten kreuzen, kommen später auch keine weiteren Ameisen mehr zu diesem Knoten.

Übung 4: Wir kennen den kürzesten Pfad zwischen dem Startknoten und allen anderen Knoten.

6.2. Kapitel 2: Schnellere Simulation

Übung 5: Wir müssen nicht alle Details simulieren, es reicht, wenn wir die Zeitpunkte simulieren, an denen etwas Interessantes passiert. Die interessanten Zeitpunkte sind immer diejenigen, an denen eine Ameise an einem Knoten ankommt. Wenn eine Ameise von einem Knoten losläuft, wissen wir bereits, wann sie ankommen wird. D.h. immer wenn eine Ameise an einem Knoten ankommt, können wir berechnen, wann sie bei den Nachbarknoten ankommen wird und diese Zeitpunkte zu den interessanten Zeitpunkten hinzufügen.

Die parallelen Schritte können wir einfach nacheinander simulieren. Wir müssen dabei aber aufpassen, dass die interessanten Ereignisse in der richtigen Reihenfolge simuliert werden.

Übung 6: Neu „springt“ die Visualisierung immer zum nächsten interessanten Zeitpunkt, wo Ameisen einen neuen Knoten erreichen.

Über den Knoten wird neu bereits eine Zeit angezeigt, sobald die Ameisen loslaufen, und nicht erst, wenn Sie angekommen sind. Diese Zeit wird mit einem Fragezeichen markiert, da sie nicht unbedingt die schlussendliche Zeit ist. Wenn Ameisen loslaufen, die einen kürzeren Weg über andere Knoten gefunden haben, kann sich so eine erwartete Ankunftszeit auch verändern und kleiner werden. Sobald die ersten Ameisen einen Knoten erreichen, wird das Fragezeichen entfernt und die Zeit ist nun final.

In der Übersichtstabelle wird nun, wie bei der Zeit, auch für den Ausgangsknoten („Previous“) bereits beim Loslaufen etwas eingetragen. Dies kann sich aber auch mit der Zeit noch ändern, falls die Ameisen einen schnelleren Weg entdecken.

Übung 7: Der Knoten, der im nächsten Schritt erreicht wird, ist der Knoten mit der kleinsten Zahl, die noch mit einem Fragezeichen versehen ist. Die Zahlen mit Fragezeichen sind die berechneten Ankunftszeiten der Ameisen, die bereits unterwegs sind. Entsprechend muss auch der Knoten mit der frühesten Ankunftszeit zuerst behandelt werden.

Alle Nachbarn des erreichten Knoten könnten sich potentiell ändern. Sie ändern sich jedoch nur, falls durch den erreichten Knoten ein neuer Knoten erreicht werden kann oder wenn ein kürzerer Pfad zustande kommt.

6.3. Kapitel 3: Der Algorithmus

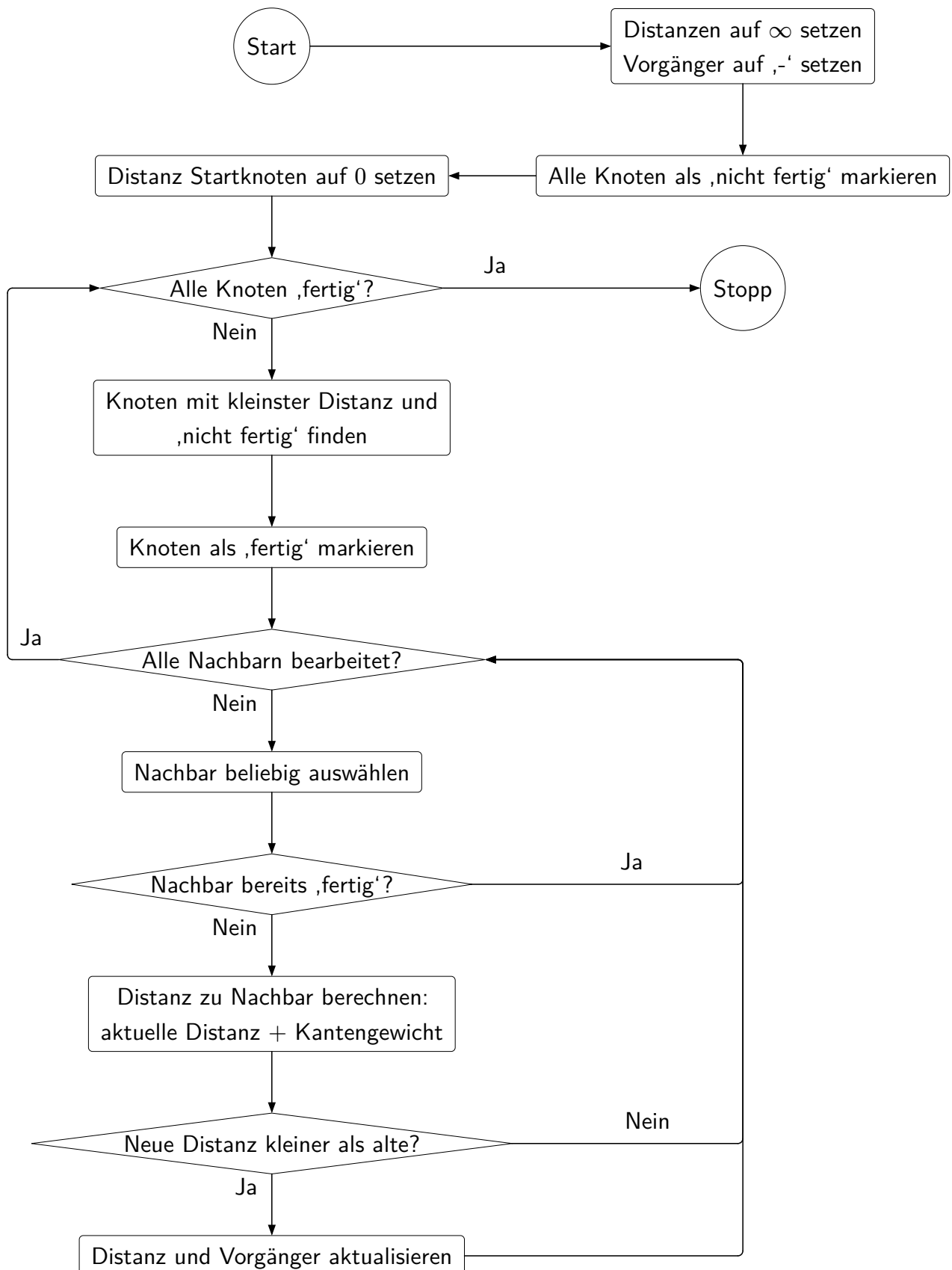
Übung 8: Unten rechts kommt ein Bereich „Priority Queue“ hinzu. Dieser zeigt alle Knoten, auf die momentan Ameisen zulaufen und die noch nicht erreicht wurden. Diese sind aufsteigend nach der erwarteten Ankunftszeit („Distance“ genannt; die Gesamtdistanz vom Startknoten) sortiert. Jeweils der oberste Eintrag (bzw. mehrere, wenn diese die gleiche Ankunftszeit haben) ist markiert. Dieser Knoten wird jeweils als nächstes verarbeitet. Im Graphen ist dieser Knoten ebenfalls markiert mit einem Rahmen um die Ankunftszeit.

Übung 9: Wir speichern neben der Länge des Pfades zu jedem Knoten auch, von wo die Ameisen gekommen sind („Previous“ in der Visualisierung). Wir können daher vom Zielknoten aus diesen Vorgängerknoten nachlaufen, um den Weg zu bestimmen, welche die schnellsten Ameisen genommen hatten.

Übung 10:

1. Distanzen für alle Knoten auf ∞ setzen.
2. Vorgänger für alle Knoten auf ‚-‘ setzen.
3. Alle Knoten als ‚nicht fertig‘ markieren.
4. Distanz des Startknoten auf 0 setzen.
5. Wiederhole, solange nicht alle Knoten als ‚fertig‘ markiert sind:
 1. Finde den Knoten mit der kleinsten Distanz, der als ‚nicht fertig‘ markiert ist.
 2. Markiere diesen Knoten als ‚fertig‘.
 3. Für jeden Nachbarn, der als ‚nicht fertig‘ markiert ist:
 1. Berechne die Distanz zu diesem Nachbarn über den aktuellen Knoten als Distanz zum aktuellen Knoten plus der Länge der Kante.

2. Falls diese Distanz kleiner ist als die bisherige Distanz zu diesem Nachbarn, aktualisiere die Distanz und setze den Vorgänger auf den aktuellen Knoten.

Übung 11:

6.4. Kapitel 4: Dijkstra von Hand**Übung 12:**

Knoten	Distanz	Vorgänger	Fertig
A	0		✓
B	3	A	✓
C	4	A	✓
D	7 4	A B	✓
E	6	D	✓
F	21 16 15	A D E	✓