BA Leipzig

Theoretische Informatik - Datenstrukturen und Algorithmen

Prof. Dr. Holger Perlt

**Dijkstra Algorithmus**

Hausarbeit

in der Studienrichtung Informatik

Eingereicht von: Fritz Schubert

08340 Schwarzenberg, Hauptstraße 4a

5CS21-2

5002083

Schwarzenberg, den 28.02.2023

Inhaltsverzeichnis

1 Einleitung 3

1.1 Zustand 3

1.2 Motivation 3

2 Aufgabenstellung 4

3 Aufgabenlösung 4

3.1 Beschreibung des Algorithmus 4

3.2 Eigenschaften und Voraussetzungen 5

3.3 Programmcode und Erläuterung 5

3.4 Aufgabe 5 5

4 Fazit 6

5 Verzeichnisse 7

5.1 Quellverzeichnis 7

5.2 Abbildungsverzeichnis 9

5.3 Abkürzungsverzeichnis 9

6 Anhänge 10

# Aufgabenstellung

* Beschreiben Sie den Algorithmus Dijkstra
* Diskutieren Sie wesentliche Eigenschaften und Voraussetzungen zur Anwendung
* Schreiben Sie einen Programmcode, welcher den Algorithmus implementiert
* Als Graphen verwenden Sie die Dateien Graph Dijk2 1.txt und Graph Dijk2 2.txt.  
  Jede Zeile ist gespeichert in der Form: Knoten1 Knoten2 Gewicht12
* Berechnen Sie die kürzesten Wege vom Startknoten 0 aus; zeigen Sie, dass der Dijkstra Algorithmus mit negativen Gewichten i. allg. keine korrekten Ergebnisse liefert.

# Aufgabenlösung

## Beschreibung des Algorithmus

Der Dijkstra-Algorithmus ist ein Algorithmus zur Bestimmung des kürzesten Weges in einem gewichteten Graphen. Er wurde 1956 von dem niederländischen Informatiker Edsger W. Dijkstra entwickelt und ist einer der bekanntesten Algorithmen in der Graphentheorie. [1,2]

Der Algorithmus arbeitet iterativ und berechnet für jeden Knoten des Graphen den kürzesten Weg zum Startknoten. Dabei nutzt er eine Prioritätswarteschlange, um die Knoten nach ihrem geschätzten Abstand zum Startknoten zu sortieren und in der Reihenfolge ihres Abstands zu besuchen. [2]

Zu Beginn wird der Startknoten in die Warteschlange eingefügt und sein Abstand zu sich selbst auf 0 gesetzt. Dann wird so lange der Knoten mit dem kleinsten Abstand aus der Warteschlange genommen, bis alle Knoten besucht wurden. Für jeden besuchten Knoten werden die Abstände seiner unbesuchten Nachbarn überprüft und aktualisiert, wenn ein kürzerer Weg gefunden wurde. []

Der Algorithmus endet, wenn alle Knoten besucht wurden oder der Zielknoten erreicht wurde. Wenn der Zielknoten erreicht wurde, kann der kürzeste Weg durch Rückverfolgung der Vorgänger-Knoten vom Zielknoten bis zum Startknoten bestimmt werden. []

## Eigenschaften und Voraussetzungen

Der Dijkstra-Algorithmus funktioniert nur für gewichtete Graphen mit nicht-negativen Kantengewichten. Für Graphen mit negativen Kantengewichten kann er keine korrekten Ergebnisse liefern und es kann zu Endlosschleifen führen.

Eine weitere Voraussetzung ist, dass der Graph zusammenhängend sein muss, da sonst keine kürzesten Wege zwischen allen Knoten existieren. []

Der Dijkstra-Algorithmus ist ein optimaler Algorithmus, d.h. er findet immer den kürzesten Weg zwischen zwei Knoten, wenn er korrekt implementiert und für den gegebenen Graphen anwendbar ist. []

## Programmcode und Erläuterung

* Siehe Datei „main.py“, Kommentare und folgende Erläuterung

Erläuterung:

Dieser Code implementiert den Dijkstra-Algorithmus, einen bekannten Algorithmus zur Lösung des Single-Source-Shortest-Path Problems.

Zu Beginn wird eine Klasse "Graph" definiert, die ein Dictionary "graph" enthält, welches die Kanten des Graphen repräsentiert. Der Graph wird als Adjazenzliste gespeichert, wobei die Schlüssel des äußeren dictionaries die Knoten sind und die Werte die Nachbarn des Knotens in Form eines inneren dictionaries sind.

Die Klasse enthält auch Methoden zur Rückgabe der Knoten, der Nachbarn eines Knotens und des Gewichts einer Kante zwischen zwei Knoten.

Die Funktion "dijkstra" ist der Kern des Algorithmus. Der Algorithmus berechnet den kürzesten Pfad von einem Startknoten zu allen anderen Knoten im Graphen. Es wird eine Liste unbesuchter Knoten initialisiert, die Gewichtung aller Knoten auf einen sehr großen Wert gesetzt und die Gewichtung des Startknotens auf 0 gesetzt.

In einer Schleife wird dann solange das Minimum der Knoten aus der Liste unbesuchter Knoten gewählt, bis alle Knoten besucht wurden. Für den ausgewählten Knoten werden die Gewichtungen seiner unbesuchten Nachbarn aktualisiert und der beste Pfad zu jedem Knoten wird gespeichert. Der aktuell besuchte Knoten wird aus der Liste unbesuchter Knoten entfernt.

Am Ende wird der kürzeste Pfad von jedem Knoten zum Startknoten zurückgegeben. Die Funktion "print\_ergebnis" gibt den kürzesten Pfad von einem Startknoten zu einem Zielknoten auf der Konsole aus.

## Aufgabe 5

Für die Datei "Graph\_Dijk2\_1.txt" liefert der Dijkstra-Algorithmus die korrekten Ergebnisse für die kürzesten Wege vom Startknoten 0 aus. Die kürzesten Wege sind wie folgt:

Der kürzeste Pfad von 0 zu 0 hat die Länge/Gewichtung 0 und sieht wie folgt aus.

0

Der kürzeste Pfad von 0 zu 1 hat die Länge/Gewichtung 2 und sieht wie folgt aus.

0 -> 1

Der kürzeste Pfad von 0 zu 2 hat die Länge/Gewichtung 5 und sieht wie folgt aus.

0 -> 1 -> 2

Der kürzeste Pfad von 0 zu 3 hat die Länge/Gewichtung 7 und sieht wie folgt aus.

0 -> 1 -> 2 -> 3

Der kürzeste Pfad von 0 zu 4 hat die Länge/Gewichtung 6 und sieht wie folgt aus.

0 -> 1 -> 4

Der kürzeste Pfad von 0 zu 5 hat die Länge/Gewichtung 7 und sieht wie folgt aus.

0 -> 1 -> 2 -> 5

Der kürzeste Pfad von 0 zu 6 hat die Länge/Gewichtung 9 und sieht wie folgt aus.

0 -> 1 -> 2 -> 3 -> 6

Der kürzeste Pfad von 0 zu 7 hat die Länge/Gewichtung 12 und sieht wie folgt aus.

0 -> 1 -> 2 -> 3 -> 6 -> 7

Der kürzeste Pfad von 0 zu 8 hat die Länge/Gewichtung 7 und sieht wie folgt aus.

0 -> 1 -> 4 -> 8

Der kürzeste Pfad von 0 zu 9 hat die Länge/Gewichtung 11 und sieht wie folgt aus.

0 -> 1 -> 2 -> 3 -> 6 -> 9

Für die Datei "Graph\_Dijk1\_2.txt" liefert der Dijkstra-Algorithmus aufgrund der negativen Gewichte keine korrekten Ergebnisse. Der Dijkstra-Algorithmus ist für solche Graphen ungeeignet, da er nur für Graphen mit nicht-negativen Gewichten geeignet ist. In diesem Fall gibt es einen negativen Kreis, was bedeutet, dass ein Kreis im Graphen existiert, dessen Gewichte negativ sind. In einem solchen Fall ist es nicht möglich, den kürzesten Pfad zu finden, da es theoretisch möglich ist, den Pfad durch diesen Kreis unendlich oft zu durchlaufen und die Distanz immer weiter zu reduzieren. Der Dijkstra-Algorithmus kann dieses Problem nicht bewältigen.

# Verzeichnisse

## Quellverzeichnis (Autor, Titel, Wo, erscheinungsdatum, link, letzter abruf)

[1] ingenieurkurse.de, Operations Research 1 DIJKSTRA-ALGORITHMUS, URL: https://www.ingenieurkurse.de/unternehmensforschung/greedy-algorithmus/dijkstra-algorithmus.html#:~:text=Der%20Dijkstra%2DAlgorithmus%20wurde%20im,der%20Zeitschrift%20Numerische%20Mathematik%20ver%C3%B6ffentlicht, Letzter Abruf: 06.03.2023

[2] Edsger Wybe Dijkstra: A note on two problems in connexion with graphs, Numerische Mathematik, Erste Auflage 1959

# Anhänge

* Main.py
* Txt dateien für die Graphen

Selbstständigkeitserklärung:

Ich versichere, dass ich die vorliegende Arbeit ohne fremde Hilfe

selbstständig verfasst und nur die angegebenen Quellen und

Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen

Werken entnommene Stellen sind unter Angabe der Quellen kenntlich

gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form

weder veröffentlicht noch einer anderen Prüfungsbehörde vorgelegt.

Ort, Datum, Unterschrift