

Graphentheoretische Konzepte und Algorithmen

Thema 02

Suchstrategien und optimale Wege

Julia Padberg



Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Kantenlänge

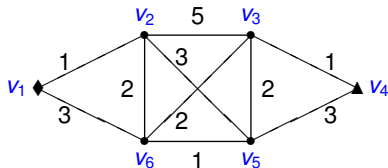
Definition

Jeder Kante eine reelle Zahl zuordnen, die wir als **Länge** dieser Kante bezeichnen. Man spricht dann von einem Graphen mit bewerteten Kanten.

BSP:

Sechs Städte werden durch das folgende Eisenbahnnetz miteinander verbunden. Die Länge einer Kante gibt dabei die durchschnittlich zu erwartende Fahrzeit in Stunden an.

Für einen Reisenden, der von \blacklozenge -Stadt nach \blacktriangle -City unterwegs ist, stellt sich die Frage, auf welchem Weg er am schnellsten zum Ziel kommt.



Der Algorithmus von Dijkstra

- ▶ dient der Bestimmung kürzester Wege
- ▶ von einem fest vorgegebenen Knoten zu allen anderen Knoten
- ▶ in einem schlichten zusammenhängenden gerichteten Graphen
- ▶ mit endlicher Knotenmenge und nichtnegativ bewerteten Kanten
- ▶ und liefert einen Weg mit einer minimalen Gesamtlänge
- ▶ Bei der Anwendung dieses Verfahrens auf ungerichtete Graphen muß jede Kante durch ein Paar entgegengesetzt gerichteter Kanten ersetzt werden.

Der Algorithmus von Dijkstra

Vorbereitungsphase

Algorithmus

Es bezeichne l_{ij} die Länge der Kante $v_i v_j$. Falls es keine Kante $v_i v_j$ gibt, sei $l_{ij} := \infty$. Für jeden Knoten $v_i \in V$ des zu untersuchenden Graphen werden drei Variable angelegt:

1. $Entf_i$ gibt die bisher festgestellte kürzeste Entfernung von v_1 nach v_i an. Der Startwert ist 0 für $i = 1$ und ∞ sonst.
2. $Vorg_i$ gibt den Vorgänger von v_i auf dem bisher kürzesten Weg von v_1 nach v_i an. Der Startwert ist v_1 für $i = 1$ und undefiniert sonst.
3. OK_i gibt an, ob die kürzeste Entfernung von v_1 nach v_i schon bekannt ist. Der Startwert für alle Werte von i ist *false*.

Der Algorithmus von Dijkstra

Iterationsphase

Algorithmus

Wiederhole

- ▶ Suche unter den Knoten v_i mit $OK_i = false$ einen Knoten v_h mit dem kleinsten Wert von $Entf_i$.
- ▶ Setze $OK_h := true$. (Wieso?)
- ▶ Für alle Knoten v_j mit $OK_j = false$, für die die Kante $v_h v_j$ existiert:
 - ▶ Falls gilt $Entf_j > Entf_h + l_{hj}$ dann
 - ▶ Setze $Entf_j := Entf_h + l_{hj}$
 - ▶ Setze $Vorg_j := v_h$

solange es noch Knoten v_i mit $OK_i = false$ gibt.

Beispiel – siehe Folie 122

Nach der Vorbereitungsphase haben *Entf*, *Vorg* und *OK* die Werte:

<i>i</i>	1	2	3	4	5	6
<i>Entf</i>	0	∞	∞	∞	∞	∞
<i>Vorg</i>	1					
<i>OK</i>	<i>f</i>	<i>f</i>	<i>f</i>	<i>f</i>	<i>f</i>	<i>f</i>

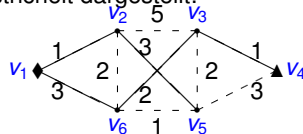
Der Knoten v_4 hat also von v_1 den Abstand 6, und auf dem Weg von v_1 nach v_4 hat den vorletzten Knoten den Index $Vorg_4 = 3$ sowie den drittletzten Knoten den Index $Vorg_3 = 6$ und wegen $Vorg_6 = 1$ lautet dann der komplette Weg: v_1, v_6, v_3, v_4

In den Iterationsschritten ist v_h nacheinander der Knoten $v_1, v_2, v_6, v_5, v_3, v_4$, und am Ende haben *Entf*, *Vorg* und *OK*

die Werte:

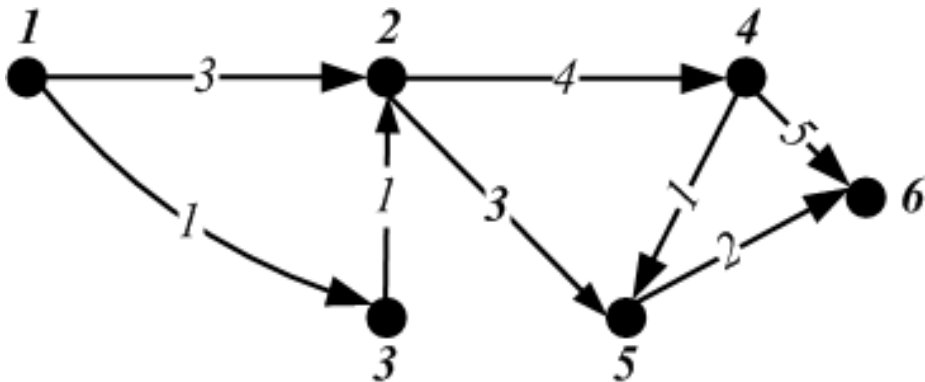
<i>i</i>	1	2	3	4	5	6
<i>Entf</i>	0	1	5	6	4	3
<i>Vorg</i>	1	1	6	3	2	1
<i>OK</i>	<i>t</i>	<i>t</i>	<i>t</i>	<i>t</i>	<i>t</i>	<i>t</i>

In der folgenden Darstellung sind die nicht zu den kürzesten Wegen gehörenden Kanten gestrichelt dargestellt:

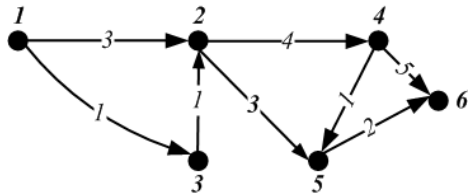


Aufgabe 1:

Führen Sie bitte für diesen Graphen den Dijkstra-Algorithmus durch:



Lösung von Aufgabe 1



	1	2	3	4	5	6
<i>Entf</i>	0	3	1	6	5	7
<i>Vorg</i>	1	1	1	2	2	5
<i>OK</i>	<i>t</i>	<i>t</i>	<i>t</i>	<i>t</i>	<i>t</i>	<i>t</i>

A*-Algorithmus

- ▶ ein informierter Suchalgorithmus
eine Schätzfunktion (Heuristik) hilft zielgerichtet zu suchen und damit die Laufzeit zu verringern
- ▶ zur Berechnung eines kürzesten Pfades zwischen zwei Knoten in einem Graphen mit positiven Kantengewichten
- ▶ 1968 von Peter Hart, Nils J. Nilsson und Bertram Raphael beschrieben
- ▶ ist optimal
es wird immer die optimale Lösung gefunden, falls eine existiert.

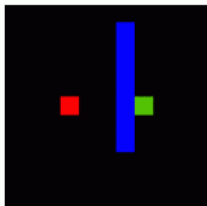
Grundidee

Die Knoten werden während der Suche in drei verschiedene Klassen eingeteilt:

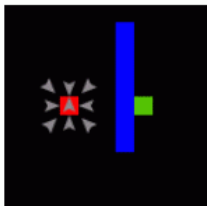
- ▶ unbekannte Knoten:
noch nicht gefunden, noch kein Weg bekannt.
Jeder Knoten (außer dem Startknoten) ist zu Beginn des Algorithmus unbekannt.
- ▶ bekannte Knoten:
bekannte (suboptimale) Wege mit ihrem f-Wert in der Open List
Wahl des vielversprechendsten Knoten
anfangs nur der Startknoten
- ▶ abschließend untersuchte Knoten:
kürzeste Wege in der Closed List
anfangs leer.

Beispiel

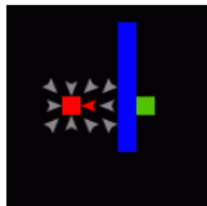
http://www.geosimulation.de/methoden/a_stern_algorithmus.htm



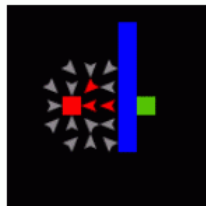
a



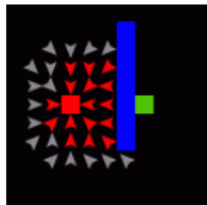
b



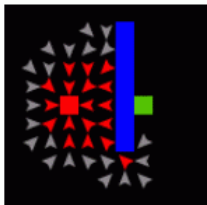
c



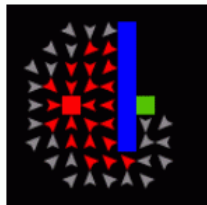
d



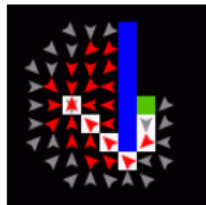
e



f



g



h

A*-Algorithmus: Initialisierung

Algorithmus

Knoten $V = \{v_1, \dots, v_n\}$

offenen Liste: OL nur der Startknoten v_1 mit $f_1 = h_1$, $g_1 = 0$ und $p_1 = v_1$

geschlossene Liste: CL leer

heuristische Knotenwerte : $h_i \in \mathbb{R}^+$

Kantenlängen zwischen v_i und v_j : $l_{ij} \in \mathbb{R}^+$

Vorgänger: $p_i \in V$

aktueller Schätzwert $f_i = \infty$ für $i > 1$

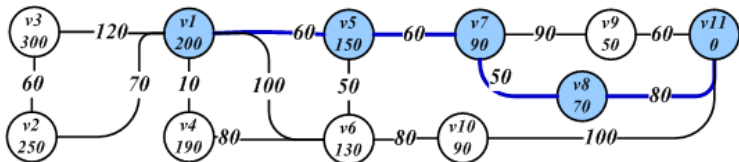
zurückgelegter Weg $g_i = \infty$ für $i > 1$

A*-Algorithmus

Algorithmus

1. Knoten v_k mit niedrigsten f_k in OL suchen
2. Knoten v_k in die CL schieben.
3. adjazente Knoten v_j , die nicht in der CL sind, in die OL schreiben und prüfen, ob $g_j > g_k + l_{k,j}$. Falls
 JA, wird der aktuelle Knoten v_k Vorgängerknoten: $p_j := v_k$
 und neuer g- und der f-Wert: $g_j := g_k + l_{k,j}$ und $f_j := g_j + h_j$
4. Falls der Zielknoten in der geschlossenen Liste; gehe zu 5.
 Falls kein Zielknoten gefunden und offene Liste leer; gehe zu 6.
 Sonst; gehe zu 1.
5. Der Pfad läßt sich vom Zielknoten aus mittels der Vorgänger bis zum Startknoten zurück verfolgen.
6. Es gibt keinen Pfad.

Beispiel A*



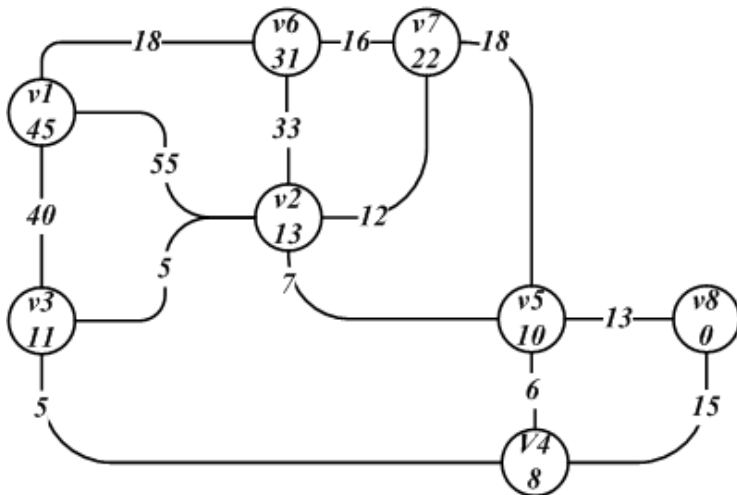
	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}
<i>Vorg</i>	v_1	v_1	v_1	v_1	v_1	v_1 v_4	v_5	v_7	v_7	v_6	v_8
<i>h</i>	200	250	300	190	150	130	90	70	50	90	0
<i>g</i>	0	70	120	10	60	100 90	120	170	210	170	250
<i>f</i>	200	320	420	200	210	230 220	210	240	260	260	250
<i>CL</i>	<i>t</i>			<i>t</i>	<i>t</i>	<i>t</i>	<i>t</i>	<i>t</i>			<i>t</i>

Länge von v_1 nach v_{11} : 250

Weg $v_1 - v_5 - v_7 - v_8 - v_{11}$

Aufgabe 2:

Bitte berechnen Sie mit Hilfe des A*-Algorithmus den kürzesten Weg von v_1 nach v_8 .



Lösung von Aufgabe 2

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
<i>Vorg</i>	v_1	v_1 v_6 v_3	v_1	v_3	v_4	v_1	v_6	v_4
<i>h</i>	45	13	11	8	10	31	22	0
<i>g</i>	0	55 51 45	40	45	51	18	34	60
<i>f</i>	45	68 64 58	51	53	61	49	56	60
<i>CL</i>	t	t	t	t		t	t	t

Länge von v_1 nach v_8 : 60

Weg $v_1 - v_3 - v_4 - v_8$

Diskussion

1. Was muss für die Heuristik gelten, damit A* klappt?

die Heuristik muss monoton sein

(sie muss diese Bedingungen erfüllen):

- ▶ Die Kosten dürfen nie überschätzt werden.
- ▶ Für jeden Knoten k und jeden Nachfolgeknoten j von k muss gelten $h(k) \leq l_{k,j} + h(j)$.

Ist eine der beiden Bedingungen verletzt ergibt sich eine nicht optimale Lösung.

2. Ist der A* immer besser als Dijkstra?

- ▶ nicht, wenn die Heuristik sehr komplex ist.
- ▶ nicht, wenn der Zielknoten auch noch gefunden werden muss