Ludwig-Maximilians-Universität München
Institut für Informatik
Prof. Dr. Thomas Seidl
Anna Beer, Florian Richter

München, 28.06.2018

Algorithmen und Datenstrukturen

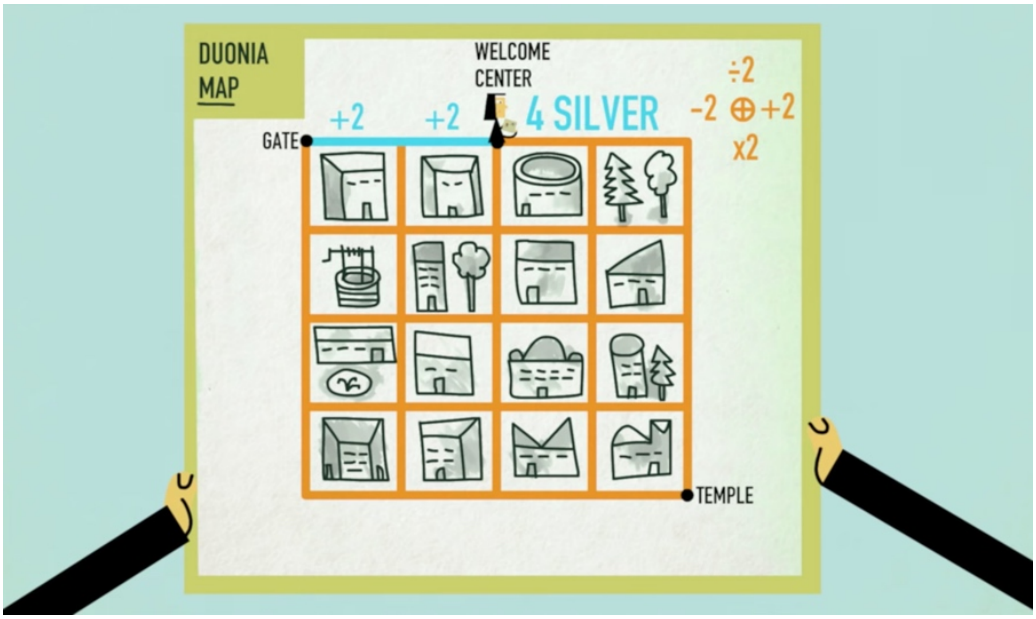
SS 2018

Übungsblatt Global 8: Graphen

Aufgabe Global 8-1 *Knobelei: Pilger*

Ein Pilger kommt am Tor der unten abgebildeten Stadt Duonia an[1]. Er geht zwei Straßen ostwärts zum Willkommenscenter, wo er das Steuersystem der Stadt erklärt bekommt: für jede Straße die er in Richtung Osten geht, zahlt er 2 Taler. Für jede Straße nach Westen bekommt er zwei Taler. Für jede nach Süden wird sein Steuerkonto verdoppelt, und für jede nach Norden halbiert. Zahlen muss man die Steuern, sobald man seinen Zielort in der Stadt erreicht. Durch die zwei Straßen, die er bereits gegangen ist, sind seine aktuellen Schulden bei der Stadt vier Taler. Allerdings hat der Pilger natürlich kein Geld, und möchte den Tempel im Südosten der Stadt mit einem ausgeglichenen Konto erreichen. Dabei muss er sich an die alte Pilgerregel halten, keinen Weg zweimal zu gehen, bereits gegangene Wege darf er aber zumindest kreuzen.

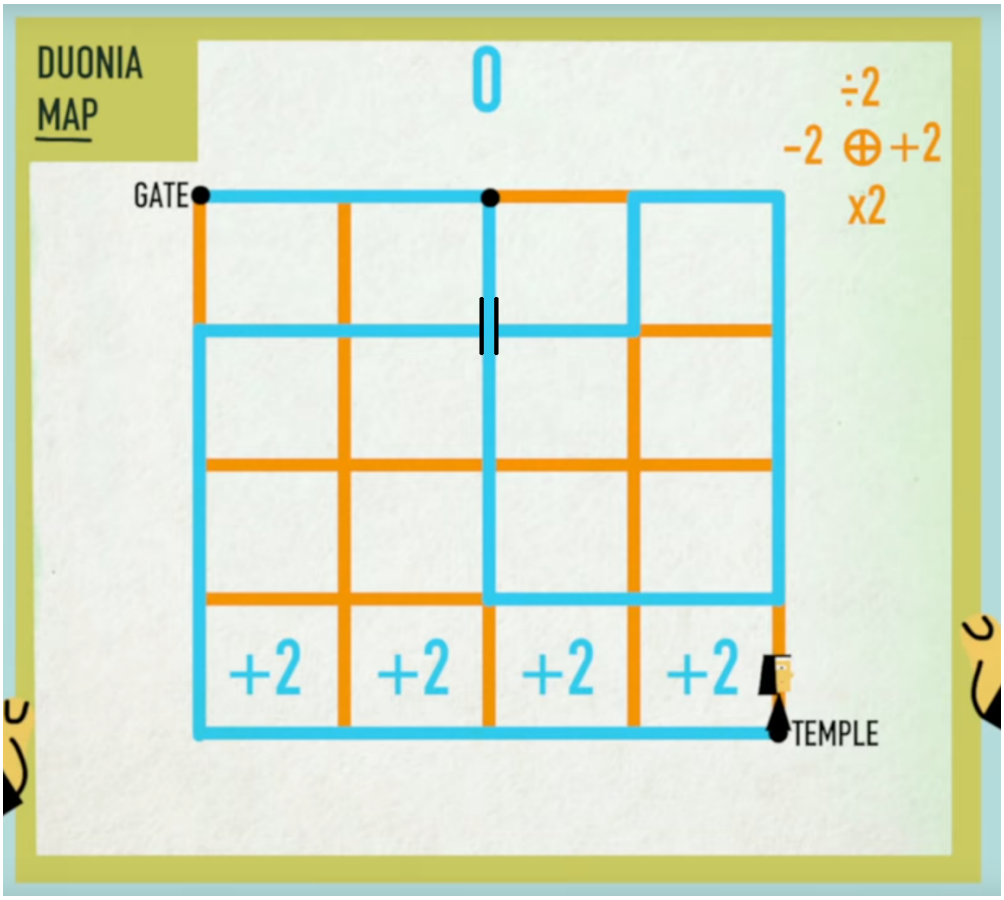
Wie muss der Pilger gehen, damit er bei Ankunft am Tempel keine Steuern zahlen muss?



[1] Source: http://thekidshouldseethis.com/post/can-you-solve-the-penniless-pilgrim-riddle

Lösungsvorschlag:

https://www.youtube.com/watch?v=6sBB-gRhfjE



Aufgabe Global 8-2 *Komplexität von Graphalgorithmen*

(a) Bestimmen Sie die Zeit- und Platzkomplexität des Dijkstra-Algorithmus für einen zusammenhängenden Graphen.

Lösungsvorschlag:

Zeitkomplexität: $O(|V|^2)$
Platzkomplexität: $O(|V|)$

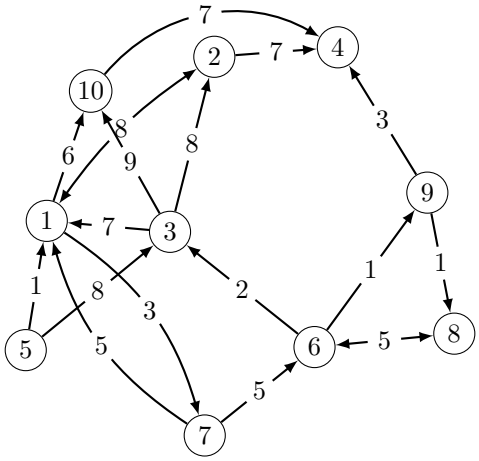
(b) Bestimmen Sie die Zeit- und Platzkomplexität des Floyd-Algorithmus

Lösungsvorschlag:

Zeitkomplexität: $O(|V|^3)$
Platzkomplexität: $O(|V|^2)$

Aufgabe Global 8-3 *Graphalgorithmen*

Gegeben ist folgender Graph:



1. Erstellen Sie die Adjazenzmatrix und Adjazenzlisten des Graphen.
   Auf der Diagonalen und statt ∞ dürfen Sie auch $-$ schreiben.

2. Führen Sie, ausgehend vom Knoten **2** einen Breitendurchlauf und einen Tiefendurchlauf durch den Graphen durch. Zeichnen Sie jeweils den Ergebnisspannbaum, der sich aus Breiten- und Tiefendurchlauf ergibt. Erstellen Sie den Spannbaum so, dass bei Mehrdeutigkeiten zuerst der Knoten mit der kleinsten Knoten-ID besucht wird.

3. Wenden Sie den Algorithmus von Floyd auf den Graphen an. Füllen Sie entsprechend dem Algorithmus die Kostenmatrix A. Die Einträge von A sollen dabei nicht nur die Kosten, sondern zusätzlich (in Klammern) den Knoten angeben, über den der kostenminimale Pfad läuft. Die Kostenmatrix und die *pathCost*-Matrix werden somit in *einer* Tabelle dargestellt. (Hinweis: der in Klammern angegebene Knoten ist nicht notwendigerweise der nächste Schritt, sondern i.d.R. der letzte Knoten, eine Verbesserung des Weges verursacht hat. Es ist in dem Sinne nur ein Wegpunkt auf einem kürzesten Weg.)

4. Berechnen Sie die kürzesten Wege ausgehend von Knoten **5** zu allen anderen Knoten. Verwenden Sie dazu den Algorithmus von Dijkstra. Zeichnen Sie für jeden Schritt einen Baum, der nur die kürzesten Wege enthält. Kennzeichnen Sie besuchte Knoten, um sie von "offenen" Knoten zu unterscheiden.

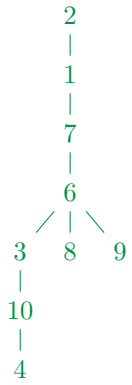
Lösungsvorschlag:

1. Adjazenzmatrix und -listen

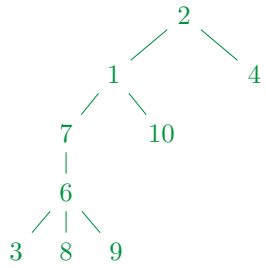
	1	2	3	4	5	6	7	8	9	10
1	–	8	–	–	–	–	3	–	–	6
2	8	–	–	7	–	–	–	–	–	–
3	7	8	–	–	–	–	–	–	–	9
4	–	–	–	–	–	–	–	–	–	–
5	1	–	8	–	–	–	–	–	–	–
6	–	–	2	–	–	–	–	5	1	–
7	5	–	–	–	–	5	–	–	–	–
8	–	–	–	–	–	5	–	–	–	–
9	–	–	–	3	–	–	–	1	–	–
10	–	–	–	7	–	–	–	–	–	–

	Gerichtet	Ungerichtet (Min)
1	2 (8), 7 (3), 10 (6)	2 (8), 3 (7), 5 (1), 7 (3), 10 (6)
2	1 (8), 4 (7)	1 (8), 3 (8), 4 (7)
3	1 (7), 2 (8), 10 (9)	1 (7), 2 (8), 5 (8), 6 (2), 10 (9)
4		2 (7), 9 (3), 10 (7)
5	1 (1), 3 (8)	1 (1), 3 (8)
6	3 (2), 8 (5), 9 (1)	3 (2), 7 (5), 8 (5), 9 (1)
7	1 (5), 6 (5)	1 (3), 6 (5)
8	6 (5)	6 (5), 9 (1)
9	4 (3), 8 (1)	4 (3), 6 (1), 8 (1)
10	4 (7)	1 (6), 3 (9), 4 (7)

2. Tiefensuche:



Breitensuche:



Aber beim Vorführen auch die “abgeschnittenen” Zweige diskutieren!

Lösungsvorschlag:

3. Algorithmus von Floyd:

Knoten	Neue Kanten	Geänderte Kanten
1	(2,7,11) (2,10,14) (3,7,10) (5,2,9) (5,7,4) (5,10,7) (7,2,13) (7,10,11)	
2	(1,4,15) (3,4,15) (5,4,16) (7,4,20)	
3	(6,1,9) (6,2,10) (6,4,17) (6,7,12) (6,10,11)	
4		
5		
6	(7,3,7) (7,8,10) (7,9,6) (8,1,14) (8,2,15) (8,3,7) (8,4,22) (8,7,17) (8,9,6) (8,10,16)	
7	(1,3,10) (1,6,8) (1,8,13) (1,9,9) (2,3,18) (2,6,16) (2,8,21) (2,9,17) (3,6,15) (3,8,20) (3,9,16) (5,6,9) (5,8,14) (5,9,10)	
8	(9,1,15) (9,2,16) (9,3,8) (9,6,6) (9,7,18) (9,10,17)	
9		(1,4,12) (1,8,10) (2,8,18) (3,8,17) (5,4,13) (5,8,11) (6,4,4) (6,8,2) (7,4,9) (7,8,7) (8,4,9)
10		

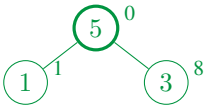
Resultierende Adjazenzmatrix (mit nächstem Knoten):

	1	2	3	4	5	6	7	8	9	10
1	–	8 (-)	10 (7)	12 (9)	–	8 (7)	3 (-)	10 (9)	9 (7)	6 (-)
2	8 (-)	–	18 (7)	7 (-)	–	16 (7)	11 (1)	18 (9)	17 (7)	14 (1)
3	7 (-)	8 (-)	–	15 (2)	–	15 (7)	10 (1)	17 (9)	16 (7)	9 (-)
4	–	–	–	–	–	–	–	–	–	–
5	1 (-)	9 (1)	8 (-)	13 (9)	–	9 (7)	4 (1)	11 (9)	10 (7)	7 (1)
6	9 (3)	10 (3)	2 (-)	4 (9)	–	–	12 (3)	2 (9)	1 (-)	11 (3)
7	5 (-)	13 (1)	7 (6)	9 (9)	–	5 (-)	–	7 (9)	6 (6)	11 (1)
8	14 (6)	15 (6)	7 (6)	9 (9)	–	5 (-)	17 (6)	–	6 (6)	16 (6)
9	15 (8)	16 (8)	8 (8)	3 (-)	–	6 (8)	18 (8)	1 (-)	–	17 (8)
10	–	–	–	7 (-)	–	–	–	–	–	–

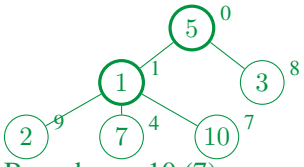
Lösungsvorschlag:

4. Algorithmus von Dijkstra:

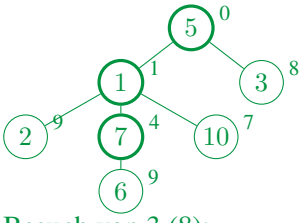
Besuch von 5 (0):



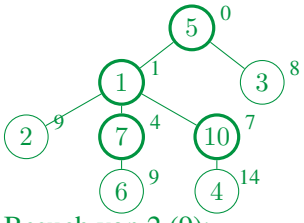
Besuch von 1 (1):



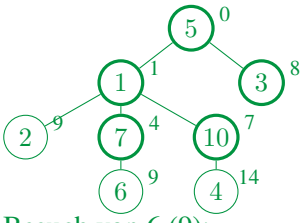
Besuch von 7 (4):



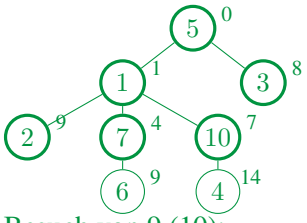
Besuch von 10 (7):



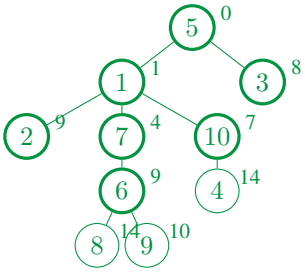
Besuch von 3 (8):



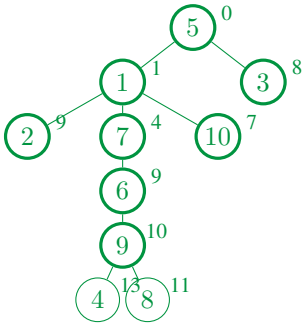
Besuch von 2 (9):



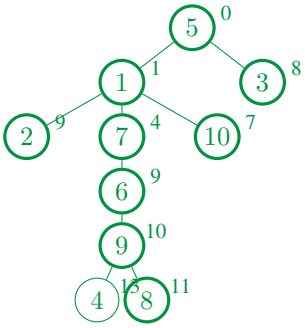
Besuch von 6 (9):



Besuch von 9 (10):



Besuch von 8 (11):



Besuch von 4 (13): Ende.