Technische Universität München
Institut für Informatik I-16
Dr. Tobias Lasser
Jakob Vogel

WiSe 2012/13
Lösungsblatt 10
14. Januar 2013

Algorithmen und Datenstrukturen

Aufgabe 1 Heap-Sort (Beispiellösung)

Die Lösung steht als Quellcode zur Verfügung, den Sie über die Website beziehen können. Beachten Sie bitte auch die Erklärungen auf den Folien.

Aufgabe 2 KMP-Präfix-Tabelle (Beispiellösung)

a)
```
 s t d : : s t r i n g
-1 0 0 0 0 0 1 2 0 0 0
```

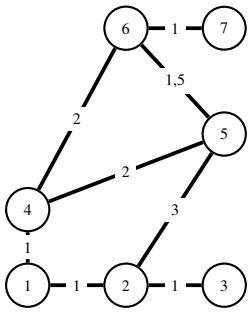
b)
```
 i n t   s o r t ( i n t *   d a t a ,   i n t   n )
-1 0 0 0 0 0 0 0 0 0 1 2 3 0 0 0 0 0 0 0 0 1 2 3 4 0
```

c)
```
 T U   M ü n c h e n
-1 0 0 0 0 0 0 0 0 0
```

Aufgabe 3 Dijkstra (Beispiellösung)

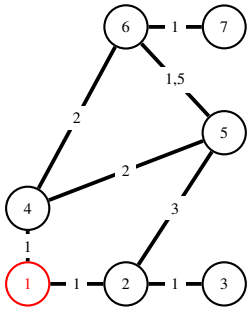
a)
- Initialisiere Startknoten 1.

	1	2	3	4	5	6	7
pred							
dist	0	∞	∞	∞	∞	∞	∞



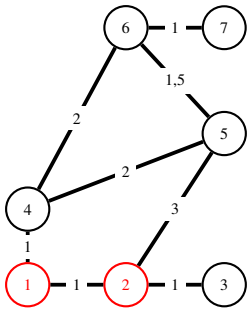
- Besuche Knoten 1. Die Auswahl erfolgt stets wegen minimaler Distanz.

	1	2	3	4	5	6	7
pred		1		1			
dist	0	1	∞	1	∞	∞	∞



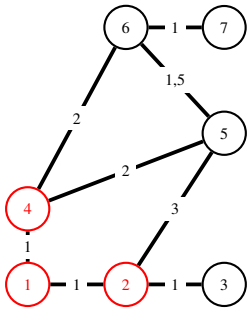
- Besuche Knoten 2.

	1	2	3	4	5	6	7
pred		1	2	1	2		
dist	0	1	2	1	4	∞	∞



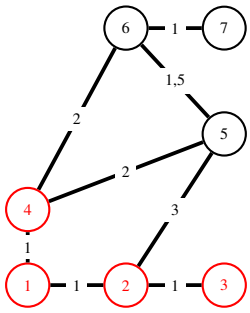
- Besuche Knoten 4. Dabei erfolgt Update für Knoten 5!

	1	2	3	4	5	6	7
pred		1	2	1	4	4	
dist	0	1	2	1	3	3	∞



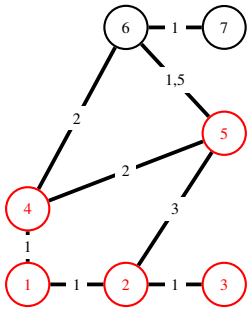
- Besuche Knoten 3. Dieser hat keine unbesuchten Nachbarn, also keine Änderungen!

	1	2	3	4	5	6	7
pred		1	2	1	4	4	
dist	0	1	2	1	3	3	∞



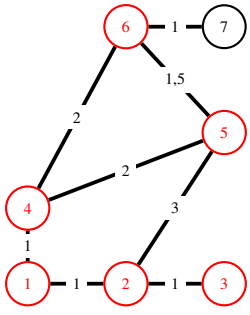
- Besuche Knoten 5. Der Pfad zu Knoten 6 über 5 ist schlechter als der bereits bekannte, also keine Änderungen!

	1	2	3	4	5	6	7
pred		1	2	1	4	4	
dist	0	1	2	1	3	3	∞



- Besuche Knoten 6.

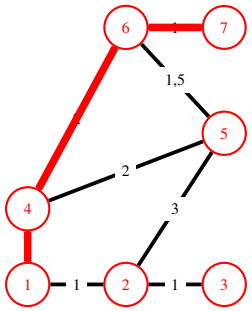
	1	2	3	4	5	6	7
pred		1	2	1	4	4	6
dist	0	1	2	1	3	3	4



- Besuche Knoten 7. Wir sind am Ziel, der kürzeste Pfad ist also $1 \to 4 \to 6 \to 7$ mit

Kosten 4. Dieser Pfad kann aus der Tabelle ausgelesen werden, indem man von Knoten 7 aus die jeweiligen Vorgänger *pred* verfolgt.

	1	2	3	4	5	6	7
pred		1	2	1	4	4	6
dist	0	1	2	1	3	3	4



b) Beim Endknoten 6 wäre der Ablauf grundsätzlich genau gleich, man kann nur früher die Suche abbrechen, da man früher am geplanten Knoten 6 angekommen ist. Bedenken Sie, dass ein solcher Abbruch problemlos möglich ist, da Dijkstra **nicht** „greedy“ ist, sondern stets in monotoner Steigerung immer länger werdende Pfade verfolgt werden. Wird also der Zielknoten besucht[1], gibt es keine kürzeren verfolgbaren Pfade mehr, und man kann abbrechen!

Insofern beginnt man bei der Lösung des *SPSP*-Problems mit einer *SSSP*-Problem-Lösung und bricht allenfalls früher ab.

Aufgabe 4 Dijkstra und A* (Beispiellösung)

Die Lösung steht als Quellcode zur Verfügung, den Sie über die Website beziehen können. Beachten Sie bitte auch die Erklärungen auf den Folien.

c) Die Heuristik wird ausschließlich in der Prioritätsliste verwendet, man würde also die Operation *decreaseKey* statt mit den akkumulierten Kosten *d[v]* mit $d[v] + h[v]$ durchführen. Insbesondere wird das Array der Pfadkosten *d* nicht verändert!

[1] Es ist wichtig, dass der Knoten wirklich besucht wird. Es genügt **nicht**, nur den ersten Vorgänger zu finden!