Gruppe **14**
Datenstrukturen und Algorithmen

Phil Pützstück, 377247
Benedikt Gerlach, 376944
Sebastian Hackenberg, 377550

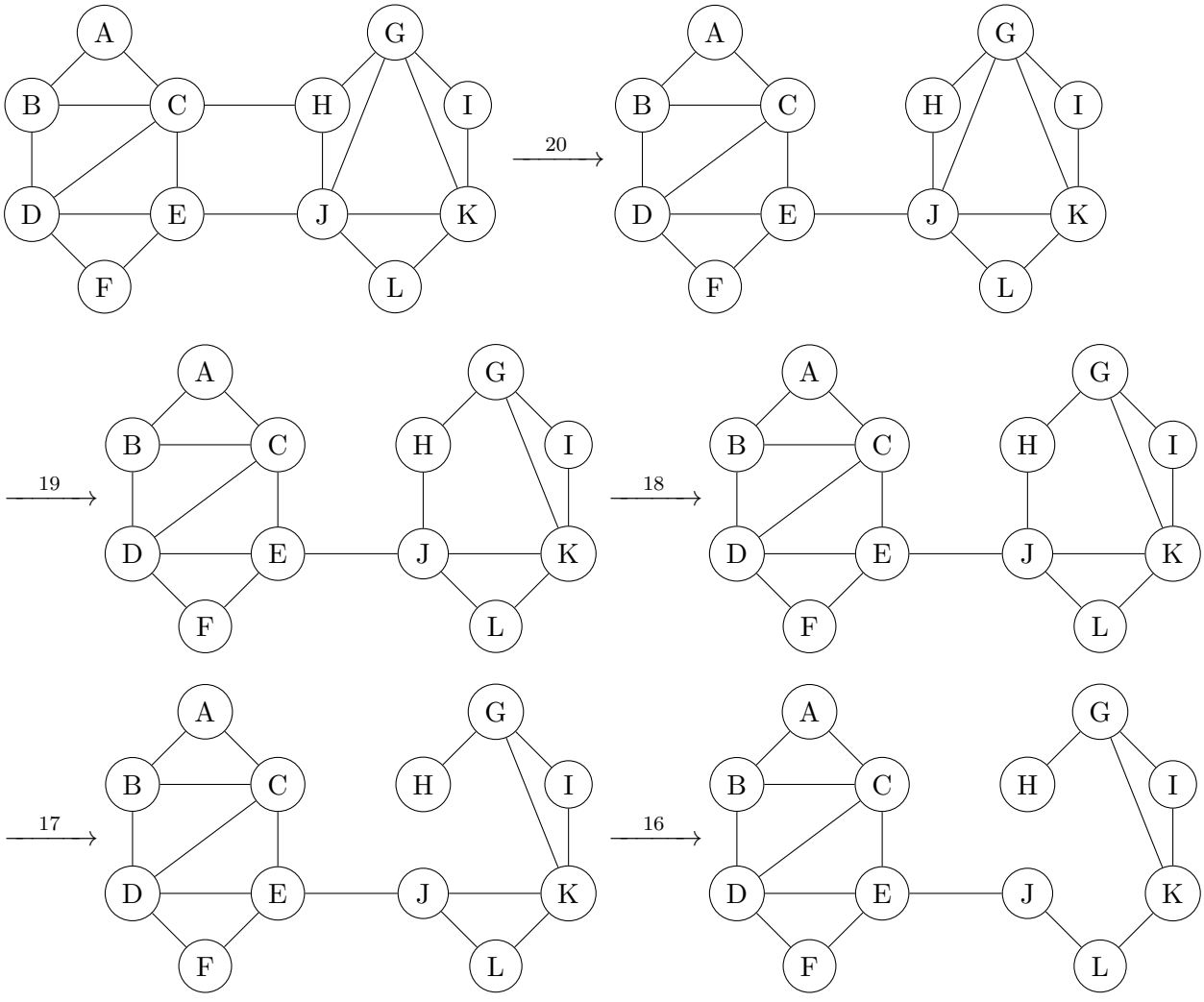
Hausaufgabe 10

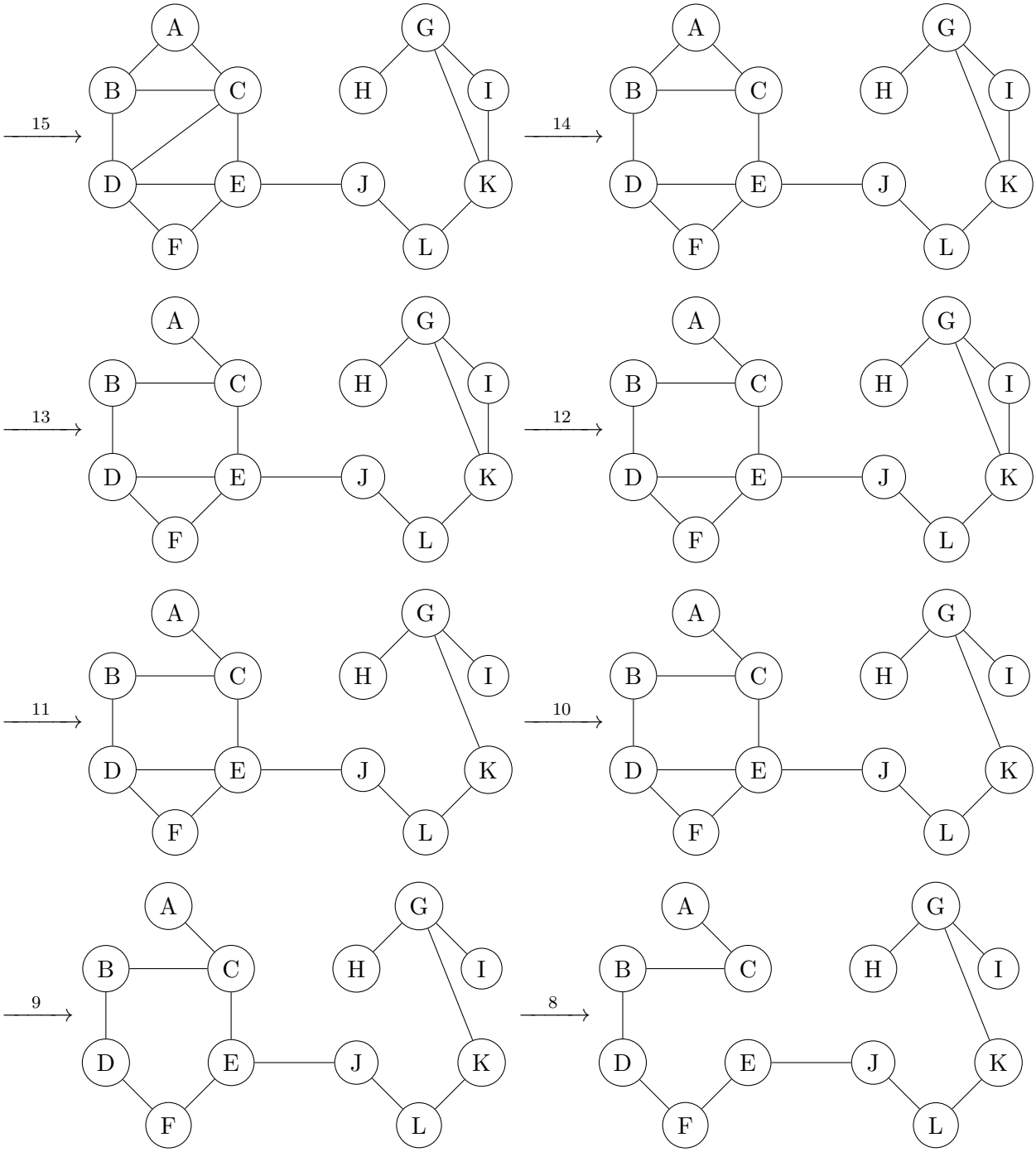
Aufgabe 1

a)

Der Algorithmus berechnet einen minimalen Spannbaum. Die Ausgabe ist ungewichtet.
Wir betrachten zuerst die Kanten mit höchstem Gewicht, absteigend zu denen mit kleinstem. Die betrachteten Kanten werden gelöscht insofern sie nicht eine notwendige Komponente des Spannbaums ist. Haben wir also 2 Kanten um einen Teil des Graphen mit einem anderen zu verbinden, so lösche wir die Kante mit höherem Gewicht. Analog zu Kruskal's Algorithmus.

b)





Von hier an kann keine Kante mehr gelöscht werden, da sonst der Graph nicht mehr zusammenhängend währe. Die restlichen Iterationen sehen dementsprechend genau wie der zuletzt gezeigte Graph aus.

Aufgabe 2

Die Aussage ist wahr:
Angenommen wir haben 2 unterschiedliche minimale Spannbäume S und T von einem Graphen mit paarweise verschiedenen Kantengewichten.

Da S und T verschieden haben wir $D_S := S_E \setminus T_E \neq \varnothing$. Weiter gibt es dann ein $k_S \in D_S$ mit

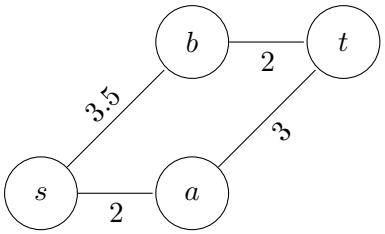
$$\forall k \in D_S \setminus \{k_S\} : W(k_S) < W(k)$$

Da also $k_S \notin T_E$ folgt, dass $T_E \cup \{k_S\}$ einen Zykel enthält. Folglich hat dieser Zykel eine Kante $k_T \in D_T := T_E \setminus S_E \neq \varnothing$ mit minimalem Gewicht in D_T. Sei nun o.B.d.A. $W(k_S) < W(k_T)$, dann können wir k_T durch k_S in T ersetzen und erhalten einen Spannbaum mit kleinerem Gewicht, Widerspruch zur annahme, dass T minimaler Spannbaum ist. (Wenn $W(k_T) < W(k_S)$ so können wir einfach k_S durch k_T ersetzen, da es um den selben Zykel in einem Graphen geht).

Insgesamt erhält man unter der Annahme einen Widerspruch, also muss der minimale Spannbaum eines Graphen mit paarweise verschiedenen Kantengewichten eindeutig sein.

Aufgabe 3

a)



Man betrachte nebenstehenden Graphen. Es gelte
$K(s) = (0,0), K(a) = (2,0), K(b) = (2,2), K(t) = (4,2)$.
Da $2 + h(a) = 6 > 3.5 + h(b) = 5.5$ wird der Algorithmus den Pfad *sat* auswählen, welcher offensichtlich nicht der kürzeste ist.
Damit ist die Aussage widerlegt.

b) Die Aussage ist wahr. Offensichtlich gilt $||(x,y)||_{\max} \leq ||(x,y)||_2$ für $x, y \in \mathbb{R}$ (*).
Wir nehmen an, der Algorithmus findet einen Pfad $p = (p_1, \cdots, p_n) \in V^n$ der Länge $n \in \mathbb{N}$, welcher nicht minimal ist. Sei weiter $q = (q_1, \cdots, q_m)$ ein minimaler Pfad mit Länge $m \in \mathbb{N}$. Wenn bei dem letzten Aufruf von `extractMin` der Knoten q_{m-1} teil des Baumes gewesen wäre, so hätte der Algorithmus anstatt von p_{n-1} zu $p_n = t$ den Pfad q_{m-1} zu $q_m = t$ gewählt, da

$$\texttt{dist[}p\texttt{]} := \texttt{dist[}p_{n-1}\texttt{]} + W(p_{n-1}, p_n) > \texttt{dist[}q_{m-1}\texttt{]} + W(q_{m-1}, q_m) =: \texttt{dist[}q\texttt{]}$$

(weil q minimal, und p nicht). Außerdem gilt ja $h(t) = 0$) Folglich muss $q_{m-1} \notin$ `Q` gegolten haben als p_n ausgewählt wurde, wobei `Q` die Liste aus dem Algoritmus ist. Damit gilt aber zu dem Zeitpunkt bevor $p_n = t$ ausgewählt wurde:

$$\exists i \in [1, m] : \texttt{dist[}q_i\texttt{]} + h(q_i) \geq \texttt{dist[}p\texttt{]}$$

Jedoch:

$$h(q_i) \overset{*}{\leq} ||K(q_i) - K(t)||_2 \leq \sum_{k=i}^{m-1} ||K(q_i) - K(q_{i+1})||_2 \leq \sum_{k=i}^{m-1} W(q_i, q_{i+1})$$

$$\implies \texttt{dist[}q_i\texttt{]} + h(q_i) \leq \texttt{dist[q]}$$

Damit folgt aber:

$$\texttt{dist[}q\texttt{]} \geq \texttt{dist[}q_i\texttt{]} + h(q_i) \geq \texttt{dist[}p\texttt{]}$$

Dies ist Widerspruch zur Annahme $\texttt{dist[}p\texttt{]} > \texttt{dist[}q\texttt{]}$. Folglich kann auch die Annahme, dass ein Pfad, welcher nicht minimal ist, erkannt wird, nicht eintreten. □

Aufgabe 4

Wir können das gegebene Problem als Graph modellieren, in dem wir die Knoten als Zustände des Puzzles betrachten. Eine Kante zwischen zwei Knoten entrspricht eben der Möglichkeit, mit genau einer Aktion von einem der beiden zum anderen zu gelangen. Da man jede Aktion auch wieder rückgängig machen kann, betrachten wie einen ungerichteten Graphen. Weiter ist jede Aktion gleich "teuer", also ist der Graph auch ungewichtet. Der ganze Graph kann hierbei $n!$ Zusände haben, jedoch ist es selten nötig alle zu generieren bzw zu besuchen.

Das Problem lässt sich gut mit einer BFS lösen. Wir starten in dem Knoten der den Startzustand des Puzzles repräsentiert. Dann generieren wir alle möglichen von hier ausgehenden Zustände und besuchen diese mit dem BFS, welcher sich den Pfad und die Länge merkt. Wir führen dies rekursiv dann für die neubesuchten Zustände im typischen Muster einer BFS aus. Sobald wir irgendwo unseren gewünschten Endzustand besuchen, können wir terminieren und den entsprechenden Pfad mit seiner Länge zurückgeben.

Aufgabe 5

Knoten	A	D	E	B	F	G	H
B	6	6	6	6	6	6	6
C	∞	∞	∞	17	17	17	17
D	3	3	3	3	3	3	3
E	3	3	3	3	3	3	3
F	∞	6	6	6	6	6	6
G	∞	12	7	7	7	7	7
H	∞	11	11	11	11	11	11

Aufgabe 6

Wir geben nur die Prioritäten der Knoten **in** der Queue an, für entfernte -

#Iteration	A	B	C	D	E	F	G	H	I	J	K	L	M
1	$\underline{0}$	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	-	4	3	∞	∞	9	∞	∞	∞	$\underline{1}$	∞	∞	∞
3	-	4	$\underline{3}$	∞	∞	9	∞	∞	∞	-	∞	∞	∞
4	-	4	-	4	$\underline{3}$	9	∞	∞	∞	-	∞	∞	∞
5	-	$\underline{4}$	-	4	-	5	∞	∞	∞	-	∞	∞	∞
6	-	-	-	4	-	5	∞	$\underline{3}$	3	-	∞	∞	∞
7	-	-	-	4	-	5	∞	-	$\underline{3}$	-	∞	∞	∞
8	-	-	-	$\underline{4}$	-	5	∞	-	-	-	∞	∞	∞
9	-	-	-	-	-	5	$\underline{3}$	-	-	-	∞	∞	∞
10	-	-	-	-	-	$\underline{5}$	-	-	-	-	∞	∞	∞
11	-	-	-	-	-	-	-	-	-	-	$\underline{4}$	∞	4
12	-	-	-	-	-	-	-	-	-	-	-	4	$\underline{3}$
13	-	-	-	-	-	-	-	-	-	-	-	$\underline{4}$	-

Minimaler Spannbaum (kantengewichte überflüssig, da sie nicht geändert werden)