Prof. Dr. Ir. Joost-Pieter Katoen

Sebastian Junges, Benjamin Kaminski, David Korzeniewski, Tim Quatmann

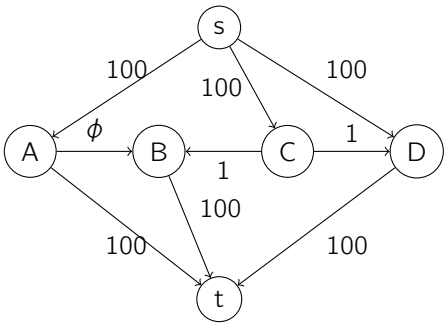
Übung 11

Hinweise:

- Dies ist das letzte reguläre Übungsblatt.
- Die Lösungen müssen bis **Donnerstag, den 12. Juli um 16:00 Uhr** in den entsprechenden Übungskasten eingeworfen werden. Sie finden die Kästen am Eingang Halifaxstr. des Informatikzentrums (Ahornstr. 55).
- Die Übungsblätter **müssen** in Gruppen von je 3 Studierenden aus der gleichen Kleingruppenübung abgegeben werden.
- Drucken Sie ggf. digital angefertigte Lösungen aus. Abgaben z.B. per Email sind nicht zulässig.
- Namen und Matrikelnummer sowie die **Nummer der Übungsgruppe** sind auf jedes Blatt der Abgabe zu schreiben. Abgaben, die aus mehreren Blättern bestehen **müssen geheftet bzw. getackert** werden! Die **Gruppennummer muss sich auf der ersten Seite oben links** befinden.
- **Bei Nichtbeachten der obigen Hinweise müssen Sie mit erheblichen Punktabzügen rechnen!**

Aufgabe 1 (Ford-Fulkerson): (5+5+2=12 Punkte)

Betrachten wir folgenden Graphen, mit $\phi = \frac{\sqrt{5}-1}{2}$.



Wir nutzen folgende augmentierende Pfade.

- $\pi_0 = sCBt$
- $\pi_1 = sABCDt$
- $\pi_2 = sCBAt$
- $\pi_3 = sDCBt$

a) Führen sie den Ford-Fulkerson Algorithmus einige, jedoch mindestens 5 Schritte aus. Notieren Sie dabei den totalen Fluss nach jeder Iteration. Nutzen sie dabei die augmentierende Pfade in folgender Reihenfolge:

$$\pi_0, \pi_1, \pi_2, \pi_1, \pi_3, \pi_1, \pi_2, \pi_1, \pi_3, \pi_1, \ldots$$

b) Konvergiert der totale Fluss vom Ford-Fulkerson Algorithmus gegen den Wert des maximalen Flusses?

Hinweise:

- $1 - \phi = \phi^2$

c) Widerspricht die Aussage aus *b*) der Korrektheit aus der Vorlesung?

Aufgabe 2 (SP-Graphen): (3+12=15 Punkte)

Ein Graph $G = (V, E, c)$ ist ein SP-Graph mit Quelle s_G und Senke t_G, wenn entweder:

- $s \to t$, d.h. $V = \{s, t\}, E = \{(s, t)\}$
- *Parallel*(X, Y) wobei $X = (V_X, E_X)$ und $Y = (V_Y, E_Y)$ SP-Graphen sind und $V_X \cap V_Y = \emptyset$. Insbesondere:
  - $V = \{s, t\} \cup V_X \setminus \{s_X, t_X\} \cup V_Y \setminus \{s_Y, t_Y\}$,
  - $E = \Big(\{(s, x) | (s_X, x) \in E_X\} \cup E_X \cup \{(x, t) | (x, t_X) \in E_X\} \cup \{(s, t) | (s_X, t_X) \in E_X\} \cup$
    $\{(s, x) | (s_Y, x) \in E_Y\} \cup \{(x, t) | (x, t_Y) \in E_Y\} \cup E_Y \cup \{(s, t) | (s_Y, t_Y) \in E_Y\}\Big) \cap V \times V$
- *Serie*(X, Y) wobei $X = (V_X, E_X)$ und $Y = (V_Y, E_Y)$ SP-Graphen sind und $V_X \cap V_Y = \emptyset$. Insbesondere:
  - $V = \{s, t\} \cup V_X \setminus \{s_X\} \cup V_Y \setminus \{s_Y, t_Y\}$,
  - $E = \big(\{(s, x) | (s_X, x) \in E_X\} \cup \{(x, y) | (x, y) \in E_X\} \cup$
    $\{(t_X, x) | (s_Y, x) \in E_Y\} \cup \{(x, t) | (x, t_Y) \in E_Y\} \cup \{(t_X, t_Y) | (s_Y, t_Y) \in E_Y\} \cup \{(x, y) | (x, y) \in E_Y\}\big) \cap V \times V$

Nehmen Sie an, $c(e) = 1$ für alle $e \in E$.

Hinweise:

- Zur Vereinfachung der Aufgabe sollte hier angenommen werden, dass E eine Multi-Menge ist, d.h. es dürfen mehrere Kanten von $u \in V$ nach $v \in V$ existieren.

a) Zeichnen Sie die folgenden Graphen:

a) *Serie*$(s \to t, s \to t)$.

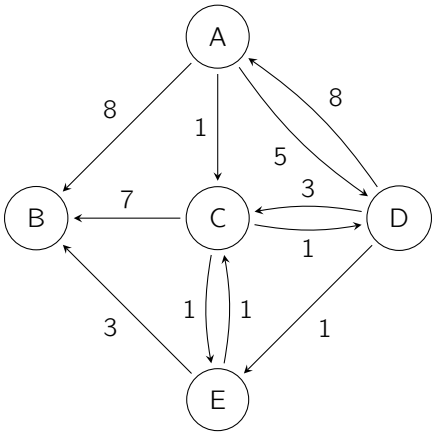
b) *Serie*$(s \to t,$ *Parallel*$(s \to t, s \to t))$

c) *Parallel*(*Serie*$(s \to t,$ *Serie*$(s \to t, s \to t))$, *Parallel*$(s \to t, s \to t))$

b) Beschreiben Sie einen Algorithmus, der eine Beschreibung eines SP-Graphen wie in Aufgabenteil a) als Eingabe nimmt, und den maximalen Fluss bestimmt. Der Algorithmus sollte nicht den gesamten Graphen aufbauen. Erweitern Sie den Algorithmus so, dass er auch mit unterschiedlichen Kantenkapazitäten umgehen kann.

Aufgabe 3 (Floyd-Warshall Algorithmus): (10 Punkte)

Betrachten Sie den folgenden Graphen:



Führen Sie den *Algorithmus von Floyd* auf diesem Graphen aus. Geben Sie dazu nach jedem Durchlauf der äußeren Schleife die aktuellen Entfernungen in einer Tabelle an. Markieren Sie die Tabellenzellen, welche sich im Vergleich zum vorherigen Schritt geändert haben.
Die erste Tabelle enthält bereits die Adjazenzmatrix nach Bildung der reflexiven Hülle. Der Eintrag in der Zeile i und Spalte j ist also ∞, falls es keine Kante vom Knoten der Zeile i zu dem Knoten der Spalte j gibt, und sonst

das Gewicht dieser Kante. Beachten Sie, dass in der reflexiven Hülle jeder Knoten eine Kante mit Gewicht 0 zu sich selbst hat.

①	A	B	C	D	E
A	0	8	1	5	∞
B	∞	0	∞	∞	∞
C	∞	7	0	1	1
D	8	∞	3	0	1
E	∞	3	1	∞	0

②	A	B	C	D	E
A					
B					
C					
D					
E					

③	A	B	C	D	E
A					
B					
C					
D					
E					

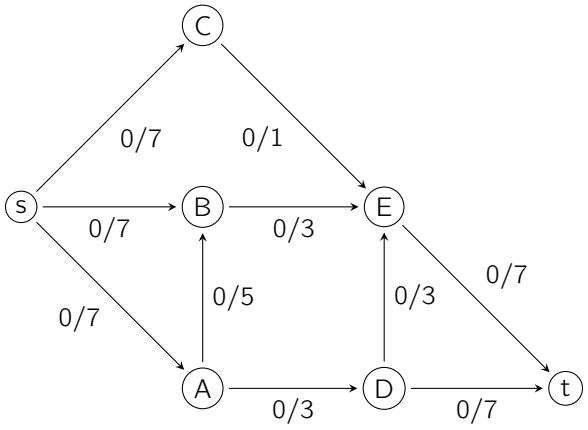
④	A	B	C	D	E
A					
B					
C					
D					
E					

⑤	A	B	C	D	E
A					
B					
C					
D					
E					

⑥	A	B	C	D	E
A					
B					
C					
D					
E					

Aufgabe 4 (Ford-Fulkerson Algorithmus): (8 Punkte)

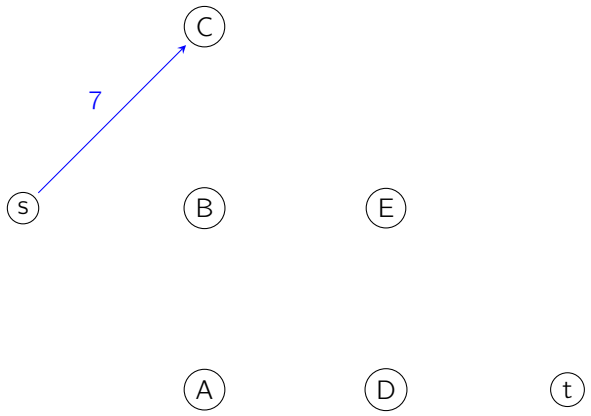
Betrachten Sie das folgende Flussnetzwerk mit Quelle s und Senke t:



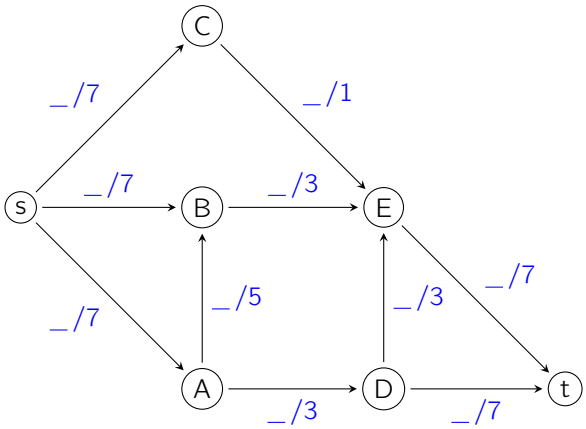
Berechnen Sie den maximalen Fluss in diesem Netzwerk mithilfe der *Ford-Fulkerson Methode*. Geben Sie dazu

jedes Restnetzwerk (auch das initiale) sowie *nach jeder Augmentierung* den aktuellen Zustand des Flussnetzwerks an. **Wählen Sie die augmentierenden Pfade so, dass der Algorithmus nach höchstens 3 Augmentierungsschritten terminiert**. Geben Sie außerdem den *Wert des maximalen Flusses* an.

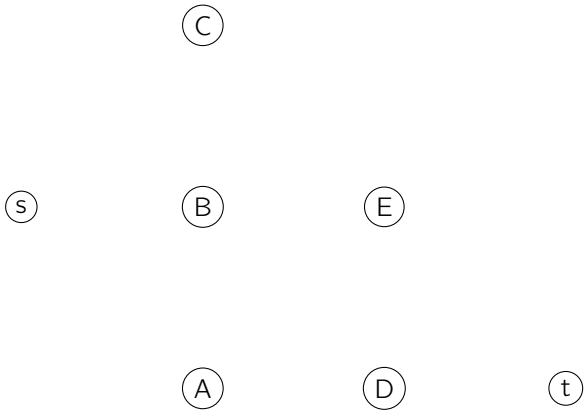
Schritt 1:
Restnetzwerk:



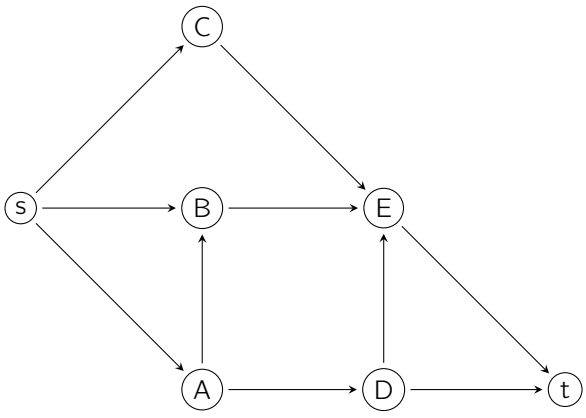
Schritt 2:
Nächstes Flussnetzwerk mit aktuellem Fluss:



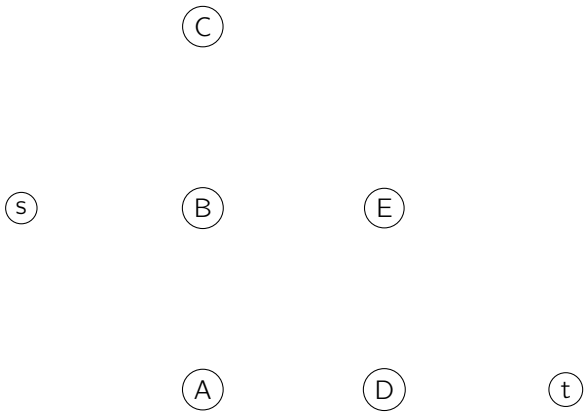
Schritt 3:
Restnetzwerk:



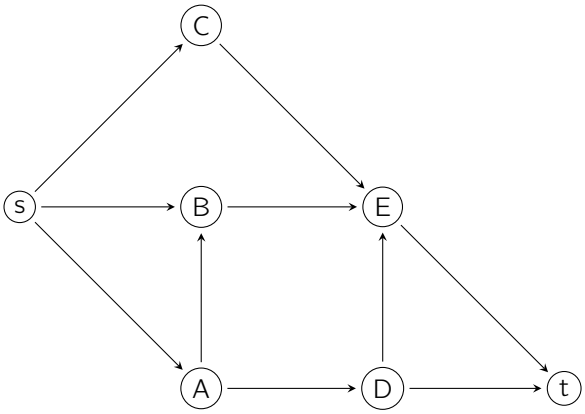
Schritt 4:
Nächstes Flussnetzwerk mit aktuellem Fluss:



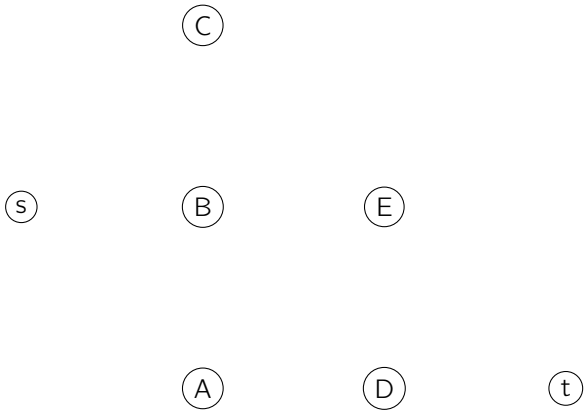
Schritt 5:
Restnetzwerk:



Schritt 6:
Nächstes Flussnetzwerk mit aktuellem Fluss:



Schritt 7:
Restnetzwerk:



Der maximale Fluss hat den Wert:

Aufgabe 5 (Dynamische Programmierung): (5+5=10 Punkte)

a) Gegeben sei ein Rucksack mit *maximaler Tragkraft* 10 sowie 6 *Gegenstände*. Der i-te Gegenstand soll hierbei ein Gewicht von w_i und einen Wert von c_i haben. Bestimmen Sie mit Hilfe des in der Vorlesung vorgestellten Algorithmus zum Lösen des Rucksackproblems mit dynamischer Programmierung den *maximalen Gesamtwert* der Gegenstände, die der Rucksack tragen kann (das Gesamtgewicht der mitgeführten Gegenstände übersteigt nicht die Tragkraft des Rucksacks). Die *Gewichte* seien dabei $w_1 = 2$, $w_2 = 6$, $w_3 = 3$, $w_4 = 1$, $w_5 = 5$ und $w_6 = 3$ und die *Werte* $c_1 = 3$, $c_2 = 9$, $c_3 = 4$, $c_4 = 2$, $c_5 = 7$ und $c_6 = 3$. Geben Sie zudem die vom Algorithmus bestimmte Tabelle `C` und die *mitzunehmenden Gegenstände* an.

	0	1	2	3	4	5	6	7	8	9	10
0											
1											
2											
3											
4											
5											
6											

b) Bestimmen Sie die *längste gemeinsame Teilsequenz* der Sequenzen `DATENSTRUKTUREN` und `ERRATEN`. Benutzen Sie hierfür den in der Vorlesung vorgestellten Algorithmus mit dynamischer Programmierung und füllen Sie die folgende Tabelle aus. Geben Sie außerdem die berechnete Teilsequenz an.

	∅	E	R	R	A	T	E	N
∅		0	0	0	0	0	0	0
D	0							
A	0							
T	0							
E	0							
N	0							
S	0							
T	0							
R	0							
U	0							
K	0							
T	0							
U	0							
R	0							
E	0							
N	0							

Längste gemeinsame Teilsequenz:

Aufgabe 6 (Dynamische Programmierung): (6+6+6=18 Punkte)

Eine Markow Kette ist ein kantengewichteter, gerichteter Graph $G = (V, E, W)$, so dass

- $W\colon E \to (0, 1]$, d.h. alle Kantengewichte sind größer als 0 und höchstens 1 und
- für alle $v \in V$ gilt $\sum_{(v,u)\in E} W((v, u)) = 1$, d.h. die Kantengewichte der ausgehenden Kanten eines Knoten summieren sich immer zu 1.

Bei einer Markow Kette $G = (V, E, W)$ interpretieren wir die Kantengewichte als Wahrscheinlichkeiten: Die Wahrscheinlichkeit eines Pfades $v_0 v_1 \dots v_k$ von G ist

$$\Pr(v_0 v_1 \dots v_k) := \prod_{i=0}^{k-1} W((v_i, v_{i+1})).$$

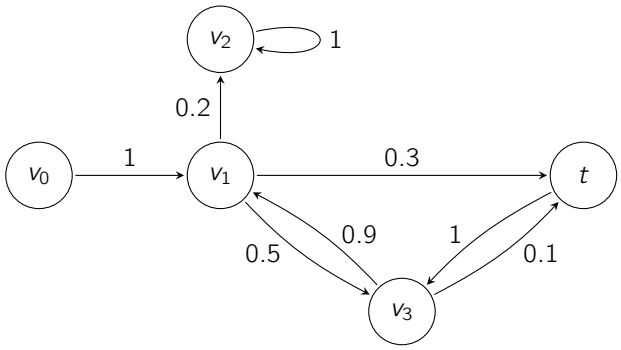
Sei $t \in V$ ein beliebiger, fest gewählter Knoten. Für jedes $v \in V$ und $k \in \mathbb{N}$ betrachten wir die Menge der Pfade

$$\Pi_v^k := \{v_0 v_1 \dots v_m \mid m \leq k, v_0 = v, v_m = t \text{ und } v_i \neq t \text{ für alle } i < m\}.$$

Die Menge Π_v^k enthält also genau die Pfade von v zu t mit Länge höchstens k, in denen t nur einmal vorkommt. Die k-beschränkte Erreichbarkeitswahrscheinlichkeit vom Knoten v zum Knoten t ist nun gegeben durch

$$\Pr_t(\Pi_v^k) := \sum_{\pi\in\Pi_v^k} \Pr(\pi).$$

a) Betrachten Sie für diese Teilaufgabe folgende Markow Kette und den Knoten $t \in V$.



Bestimmen Sie die folgenden Pfadmengen Π_v^k sowie die zugehörigen Wahrscheinlichkeiten $\Pr_t(\Pi_v^k)$.

i) $\Pi_{v_3}^0 = \ldots$ (enthält 0 Pfade) $\quad \Pr_t(\Pi_{v_3}^0) = \ldots$

ii) $\Pi_{v_2}^3 = \ldots$ (enthält 0 Pfade) $\quad \Pr_t(\Pi_{v_2}^3) = \ldots$

iii) $\Pi_t^3 = \ldots$ (enthält 1 Pfad) $\quad \Pr_t(\Pi_t^3) = \ldots$

iv) $\Pi_{v_3}^3 = \ldots$ (enthält 3 Pfade) $\quad \Pr_t(\Pi_{v_3}^3) = \ldots$

v) $\Pi_{v_1}^4 = \ldots$ (enthält 4 Pfade) $\quad \Pr_t(\Pi_{v_1}^4) = \ldots$

vi) $\Pi_{v_0}^5 = \ldots$ (enthält 4 Pfade) $\quad \Pr_t(\Pi_{v_0}^5) = \ldots$

Hinweise:

- Versuchen Sie für Aufgabenteil v) und vi) Ihre Ergebnisse aus vorherigen Aufgabenteilen wiederzuverwenden.

b) Bestimmen Sie eine Rekursionsgleichung für die Pfadmengen Π_v^k mit $k \in \mathbb{N}$ und $v \in V$.

c) Bestimmen Sie eine Rekursionsgleichung für $\Pr_t(\Pi_v^k)$ mit $k \in \mathbb{N}$ und $v \in V$.

Aufgabe 7 (Dynamische Programmierung): (6+9=15 Punkte)

Schreiben Sie Pseudocode, der mittels dynamischer Programmierung folgende Rekursionsgleichungen löst: Schreiben Sie dazu einige Kommentare in den Pseudocode, um ihre Idee zur Erläutern.

a) Für ein beliebiges i:

$$a_i = \begin{cases} i & i \leq 2 \\ \frac{a_{i+1}+a_{i-1}}{2} & i \geq 3 \text{ und } i \bmod 2 = 0 \\ a_{i-2} & \text{sonst.} \end{cases}$$

b) Für ein beliebiges i und j:

$$b_{i,j} = \begin{cases} i & \text{für } j = 1 \\ 1 & \text{für } i = 1 < j \\ b_{j,i} & \text{für } j > i > 1 \\ b_{i+1,j-1} + b_{i-1,j-1} & \text{sonst.} \end{cases}$$

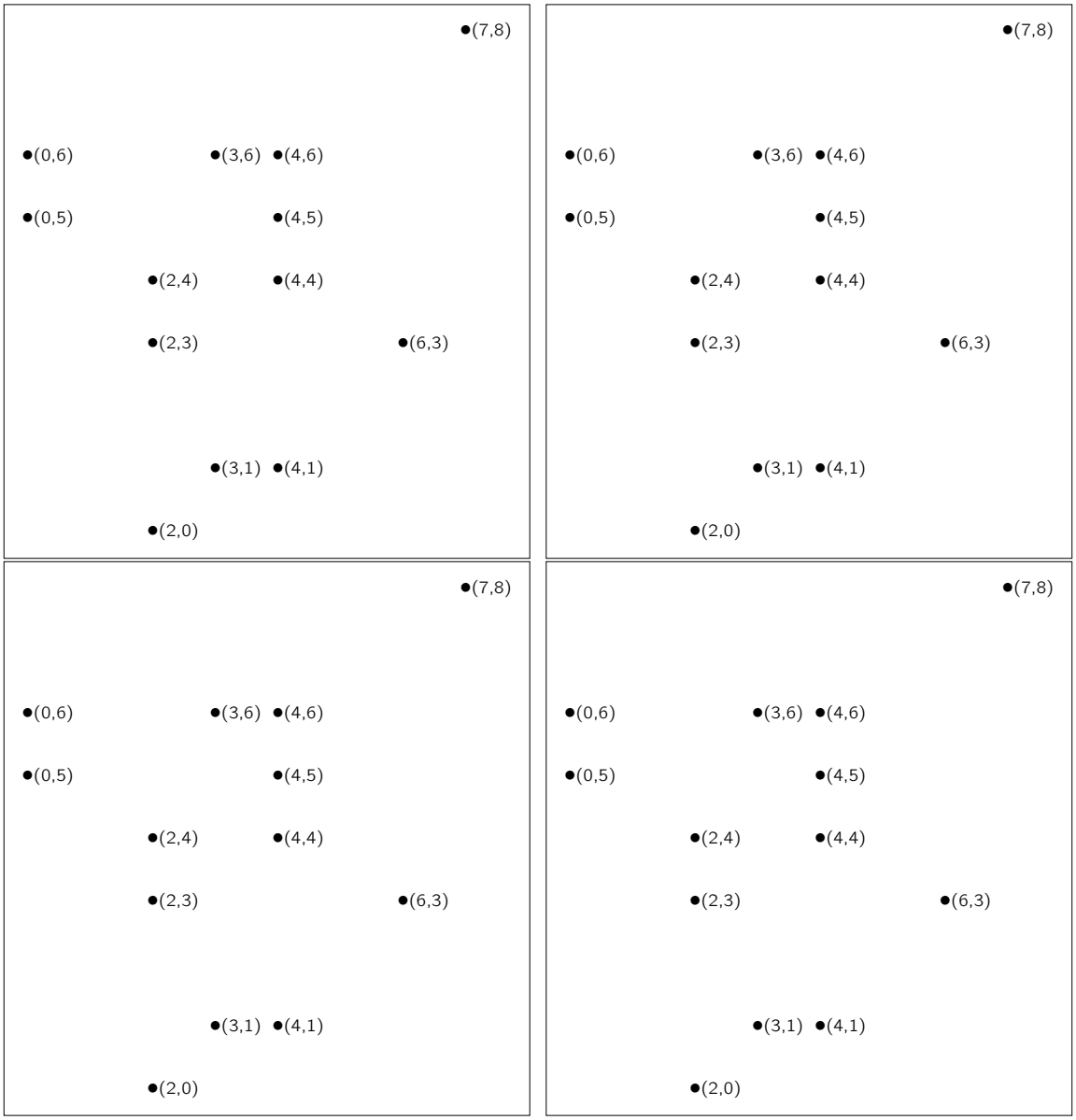
Aufgabe 8 (Graham Scan): (12 Punkte)

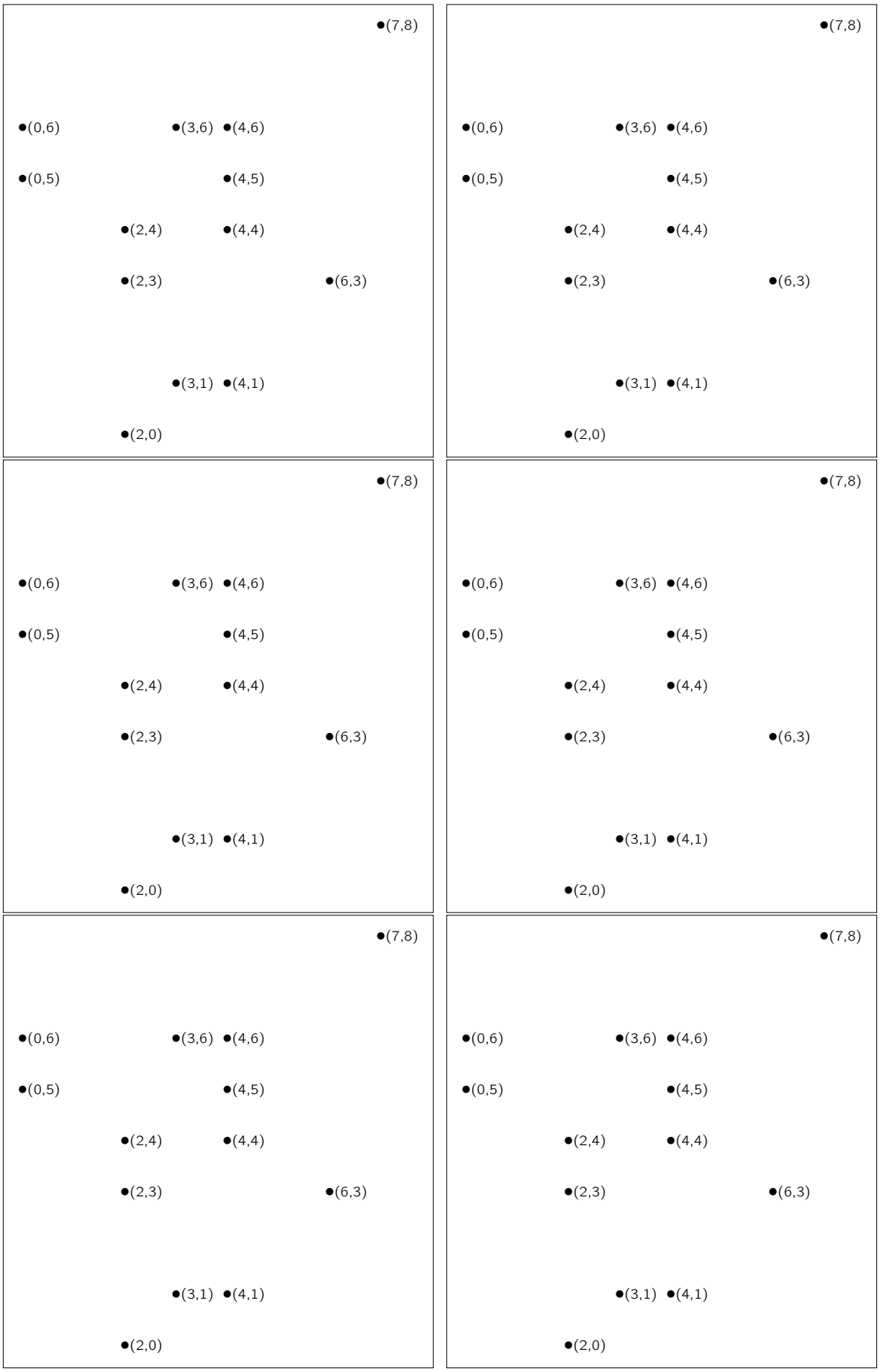
Berechnen Sie die konvexe Hülle der folgenden Punktmenge. Benutzen Sie dafür *Grahams Scan* wie *in der Vorlesung* vorgestellt und geben Sie die Teilschritte *nach jeder Iteration* (also nach jedem neu hinzugefügten Punkt) an.

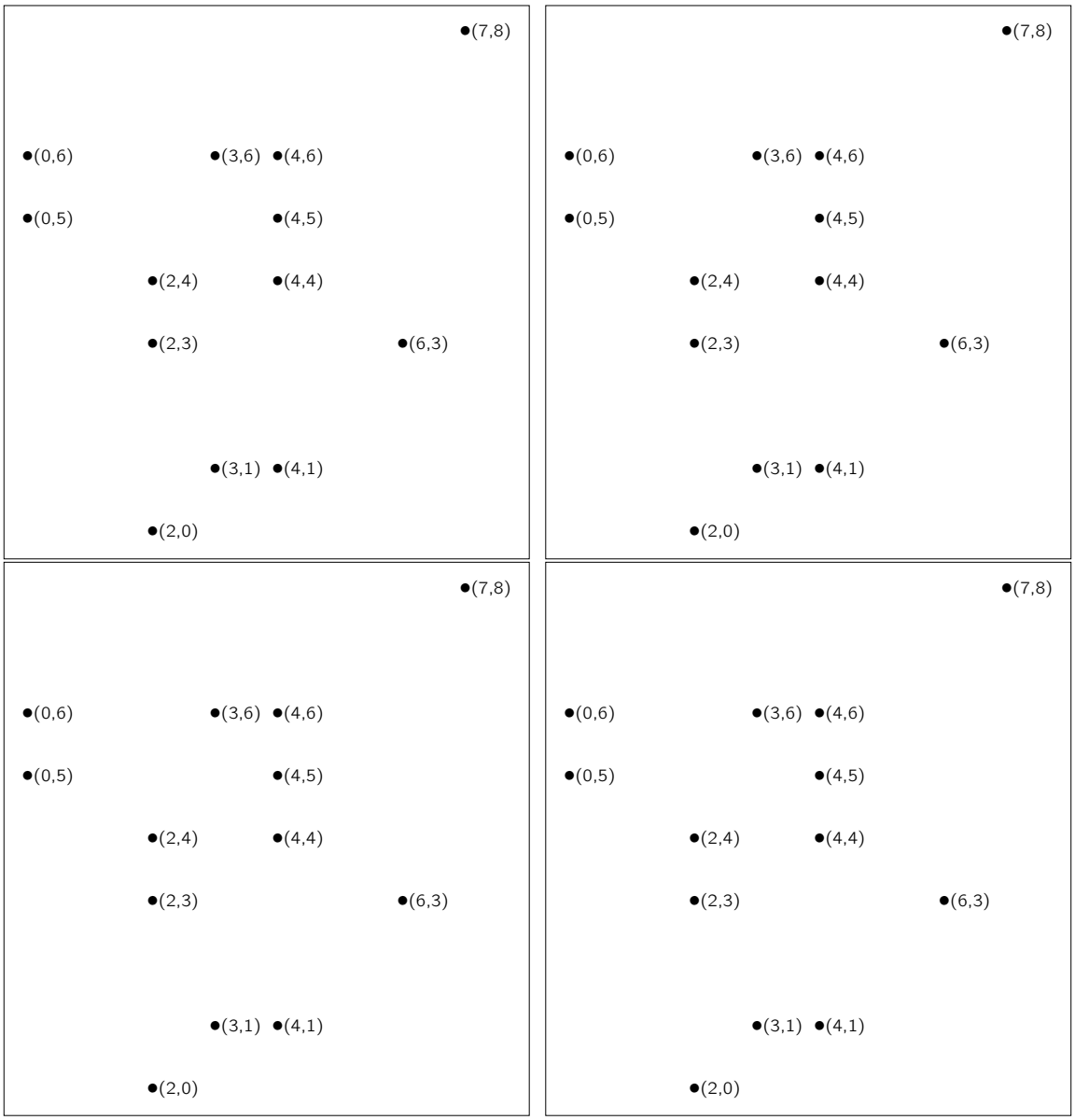
Hinweise:

- Für diese Aufgabe sind die Inhalte der Vorlesung "Algorithmische Geometrie" vom Montag, den 9.7. relevant.

- Nicht alle der unten aufgeführten Koordinatensysteme werden benötigt.




(7,8)
(0,6)
(3,6)
(4,6)
(0,5)
(4,5)
(2,4)
(4,4)
(2,3)
(6,3)
(3,1)
(4,1)
(2,0)


(7,8)
(0,6)
(3,6)
(4,6)
(0,5)
(4,5)
(2,4)
(4,4)
(2,3)
(6,3)
(3,1)
(4,1)
(2,0)
(7,8)
(0,6)
(3,6)
(4,6)
(0,5)
(4,5)
(2,4)
(4,4)
(2,3)
(6,3)
(3,1)
(4,1)
(2,0)
(7,8)
(0,6)
(3,6)
(4,6)
(0,5)
(4,5)
(2,4)
(4,4)
(2,3)
(6,3)
(3,1)
(4,1)
(2,0)
(7,8)
(0,6)
(3,6)
(4,6)
(0,5)
(4,5)
(2,4)
(4,4)
(2,3)
(6,3)
(3,1)
(4,1)
(2,0)