

# Übung 11

## – Musterlösung –

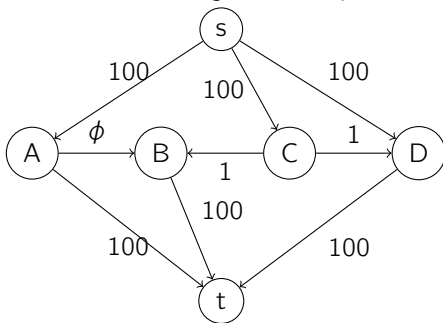
**Hinweise:**

- 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.

- $\pi_0 = sCBt$
- $\pi_1 = sABCDt$
- $\pi_2 = sCBA t$
- $\pi_3 = sDCBt$

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:

$$\pi_0, \pi_1, \pi_2, \pi_1, \pi_3, \pi_1, \pi_2, \pi_1, \pi_3, \pi_1, \dots$$

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?

**Lösung:** \_\_\_\_\_

- a)
- Augmentiere entlang  $\pi_0$  mit Wert 1.
  - Augmentiere entlang  $\pi_1$  mit Wert  $\phi$ . (Insgesamt  $1 + \phi$ ).
  - Augmentiere entlang  $\pi_2$  mit Wert  $\phi$ . (Insgesamt  $1 + 2\phi$ ).
  - Augmentiere entlang  $\pi_1$  mit Wert  $\phi^2$ . (Insgesamt  $1 + 2\phi + \phi^2$ ).
  - Augmentiere entlang  $\pi_3$  mit Wert  $\phi^2$ . (Insgesamt  $1 + 2\phi + 2\phi^2$ ).
  - Augmentiere entlang  $\pi_1$  mit Wert  $\phi^3$ . (Insgesamt  $1 + 2\phi + 2\phi^2 + \phi^3$ ).
  - Augmentiere entlang  $\pi_2$  mit Wert  $\phi^3$ . (Insgesamt  $1 + 2\phi + 2\phi^2 + 2\phi^3$ ).
  - Augmentiere entlang  $\pi_1$  mit Wert  $\phi^4$ . (Insgesamt  $1 + 2\phi + 2\phi^2 + 2\phi^3 + \phi^4$ ).
  - Augmentiere entlang  $\pi_3$  mit Wert  $\phi^4$ . (Insgesamt  $1 + 2\phi + 2\phi^2 + 2\phi^3 + 2\phi^4$ ).
- b) Nein, der Fluss konvergiert gegen  $1 + 2 \sum_{i=1}^{\infty} \phi^i = 1 - 2 + 2 + 2 \sum_{i=1}^{\infty} \phi^i = -1 + 2 \sum_{i=0}^{\infty} \phi^i = -1 + \frac{2}{1-\phi} = -1 + \frac{2}{\phi^2} = 2 + \sqrt{5} < 5$ , der maximale Fluss ist aber 201.
- c) Nein,  $\phi$  ist keine rationale (und insbesondere keine ganze) Zahl.

## 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:

- $s \rightarrow t$ , 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:
  - $V = \{s, t\} \cup V_X \setminus \{s_X, t_X\} \cup V_Y \setminus \{s_Y, t_Y\}$ ,
  - $E = \left( \{(s, x) \mid (s_X, x) \in E_X\} \cup E_X \cup \{(x, t) \mid (x, t_X) \in E_X\} \cup \{(s, t) \mid (s_X, t_X) \in E_X\} \cup \{(s, x) \mid (s_Y, x) \in E_Y\} \cup \{(x, t) \mid (x, t_Y) \in E_Y\} \cup E_Y \cup \{(s, t) \mid (s_Y, t_Y) \in E_Y\} \right) \cap V \times V$
- *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:
  - $V = \{s, t\} \cup V_X \setminus \{s_X\} \cup V_Y \setminus \{s_Y, t_Y\}$ ,
  - $E = \left( \{(s, x) \mid (s_X, x) \in E_X\} \cup E_X \cup \{(t_X, x) \mid (s_Y, x) \in E_Y\} \cup \{(x, t) \mid (x, t_Y) \in E_Y\} \cup \{(t_X, t) \mid (s_Y, t_Y) \in E_Y\} \cup E_Y \right) \cap V \times V$

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

Hinweise:

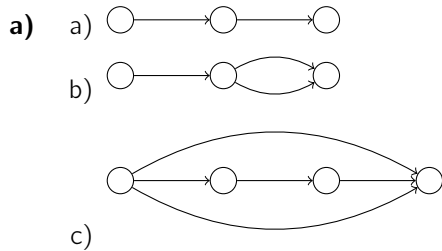
- Zur Vereinfachung der Aufgabe sollte hier angenommen werden, dass  $E$  eine Multi-Menge ist, d.h. es dürfen mehrere Kanten von  $u \in V$  nach  $v \in V$  existieren.

a) Zeichnen Sie die folgenden Graphen:

- a)  $Serie(s \rightarrow t, s \rightarrow t)$ .
- b)  $Serie(s \rightarrow t, Parallel(s \rightarrow t, s \rightarrow t))$
- c)  $Parallel(Serie(s \rightarrow t, Serie(s \rightarrow t, s \rightarrow t)), Parallel(s \rightarrow t, s \rightarrow t))$

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.

**Lösung:** \_\_\_\_\_



b) Beachten wir folgende rekursive Definition:

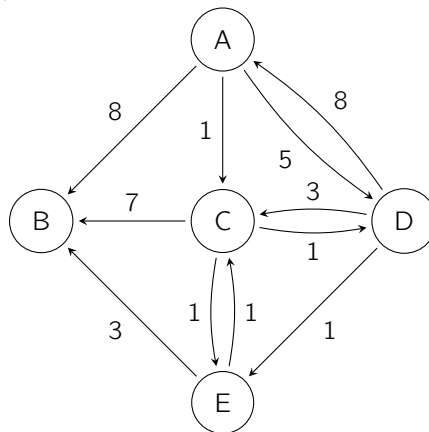
- $MaxFluss(s \rightarrow t) = c(e) = 1$ .
- $Serie(X, Y) = \min\{MaxFluss(X), MaxFluss(Y)\}$ .
- $Parallel(X, Y) = MaxFluss(X) + MaxFluss(Y)$ .

Damit ergibt sich ein Algorithmus, der von den Blättern aufwärts den maximalen Fluss berechnet.

**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  $\infty$ , 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	$\infty$
B	$\infty$	0	$\infty$	$\infty$	$\infty$
C	$\infty$	7	0	1	1
D	8	$\infty$	3	0	1
E	$\infty$	3	1	$\infty$	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					

**Lösung:** \_\_\_\_\_

①	A	B	C	D	E
A	0	8	1	5	$\infty$
B	$\infty$	0	$\infty$	$\infty$	$\infty$
C	$\infty$	7	0	1	1
D	8	$\infty$	3	0	1
E	$\infty$	3	1	$\infty$	0

②	A	B	C	D	E
A	0	8	1	5	$\infty$
B	$\infty$	0	$\infty$	$\infty$	$\infty$
C	$\infty$	7	0	1	1
D	8	16	3	0	1
E	$\infty$	3	1	$\infty$	0

③	A	B	C	D	E
A	0	8	1	5	$\infty$
B	$\infty$	0	$\infty$	$\infty$	$\infty$
C	$\infty$	7	0	1	1
D	8	16	3	0	1
E	$\infty$	3	1	$\infty$	0

④	A	B	C	D	E
A	0	8	1	2	2
B	$\infty$	0	$\infty$	$\infty$	$\infty$
C	$\infty$	7	0	1	1
D	8	10	3	0	1
E	$\infty$	3	1	2	0

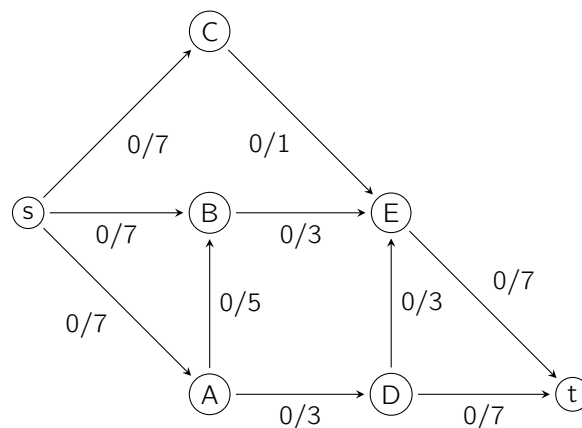
⑤	A	B	C	D	E
A	0	8	1	2	2
B	$\infty$	0	$\infty$	$\infty$	$\infty$
C	9	7	0	1	1
D	8	10	3	0	1
E	10	3	1	2	0

⑥	A	B	C	D	E
A	0	5	1	2	2
B	$\infty$	0	$\infty$	$\infty$	$\infty$
C	9	4	0	1	1
D	8	4	2	0	1
E	10	3	1	2	0

**Aufgabe 4 (Ford-Fulkerson Algorithmus):**

**(8 Punkte)**

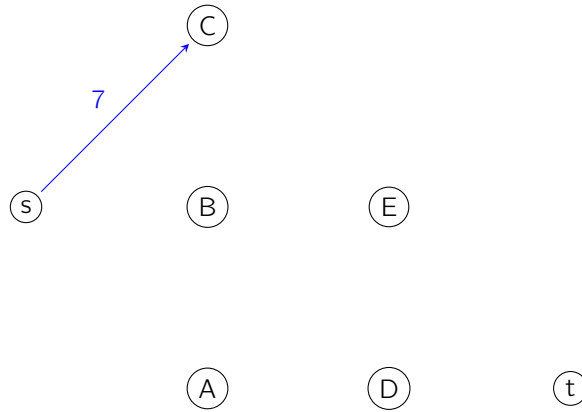
Betrachten Sie das folgende Flussnetzwerk mit Quelle s und Senke t:



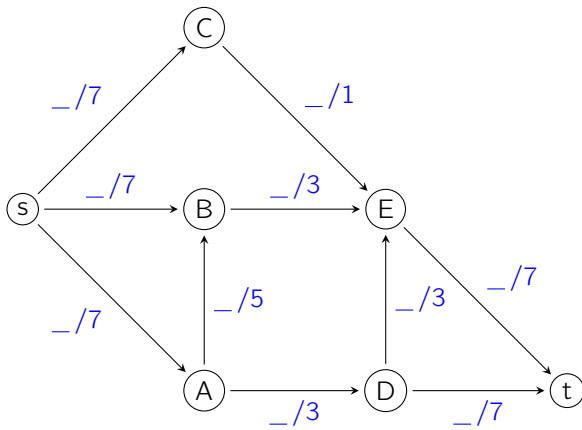
Berechnen Sie den maximalen Fluss in diesem Netzwerk mithilfe der *Ford-Fulkerson Methode*. Geben Sie dazu *jedes Restnetzwerk* (auch das initiale) sowie *nach jeder Augmentierung* den aktuellen Zustand des Flussnetzwerks

an. Wählen Sie die augmentierenden Pfade so, dass der Algorithmus nach höchstens 3 Augmentierungsschritten terminiert. Geben Sie außerdem den Wert des maximalen Flusses an.

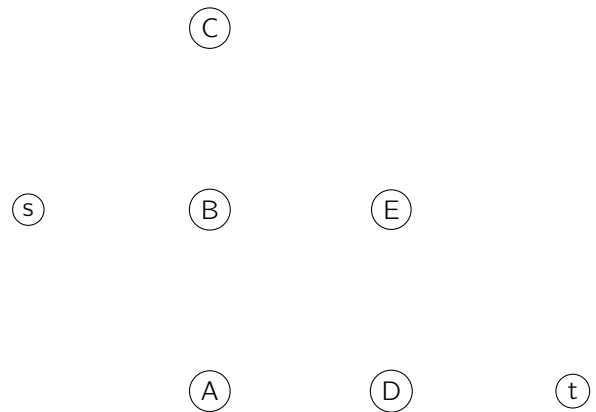
Schritt 1:  
 Restnetzwerk:



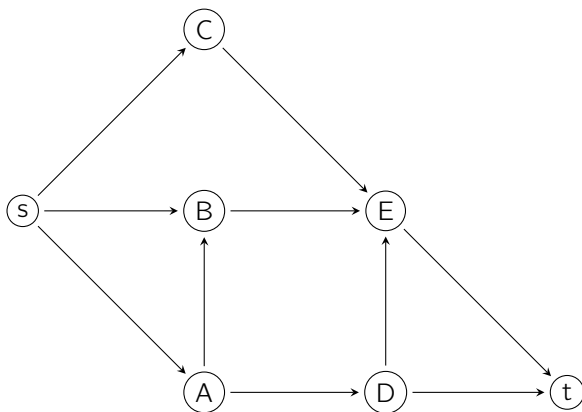
Schritt 2:  
 Nächstes Flussnetzwerk mit aktuellem Fluss:



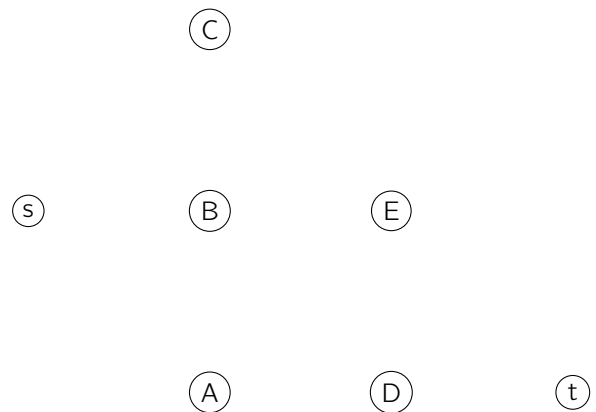
Schritt 3:  
 Restnetzwerk:



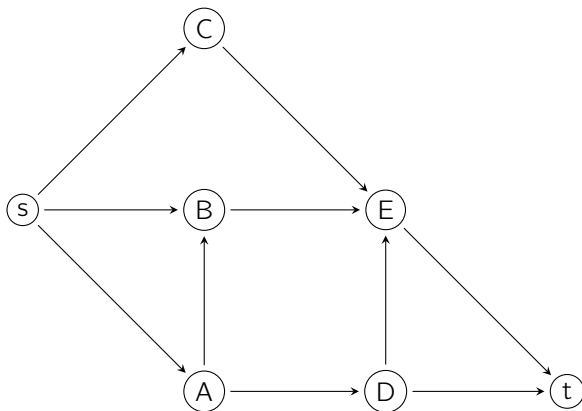
Schritt 4:  
 Nächstes Flussnetzwerk mit aktuellem Fluss:



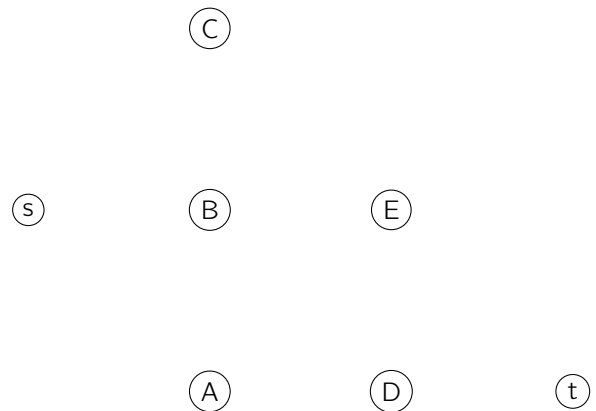
Schritt 5:  
 Restnetzwerk:



Schritt 6:  
 Nächstes Flussnetzwerk mit aktuellem Fluss:



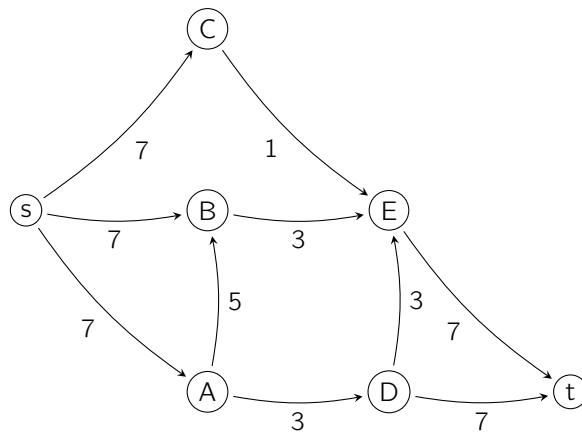
Schritt 7:  
 Restnetzwerk:



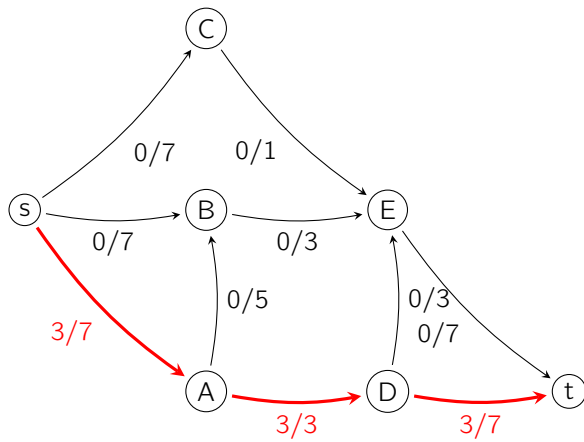
Der maximale Fluss hat den Wert:

**Lösung:** \_\_\_\_\_  
 \_\_\_\_\_

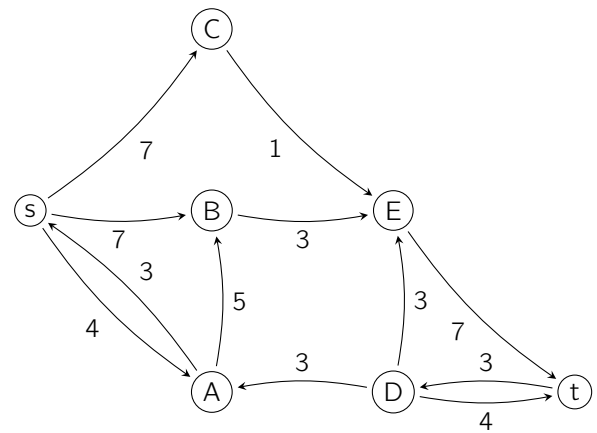
Schritt 1:  
 Restnetzwerk:



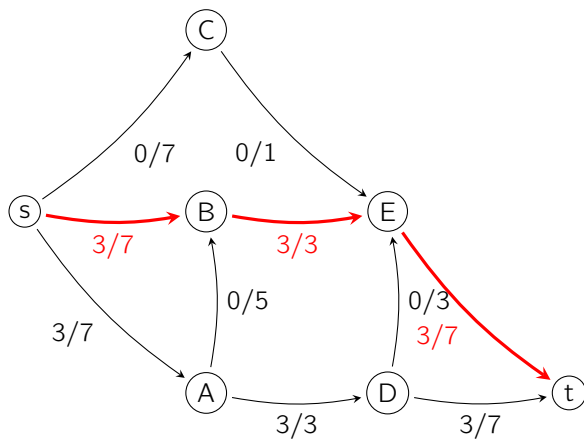
Schritt 2:  
Nächstes Flussnetzwerk mit aktuellem Fluss:



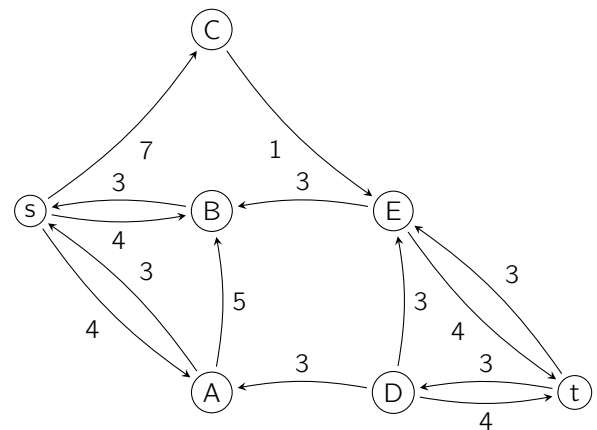
Schritt 3:  
Restnetzwerk:



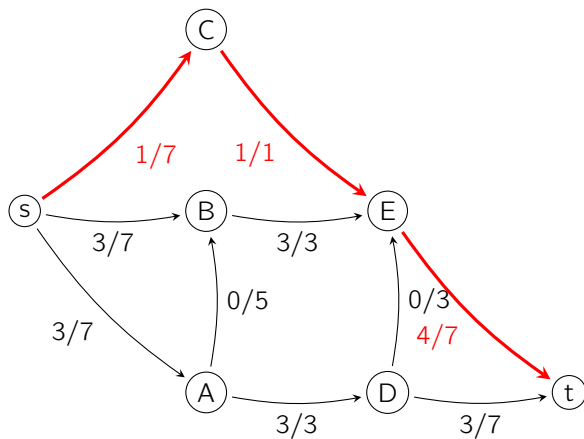
Schritt 4:  
Nächstes Flussnetzwerk mit aktuellem Fluss:



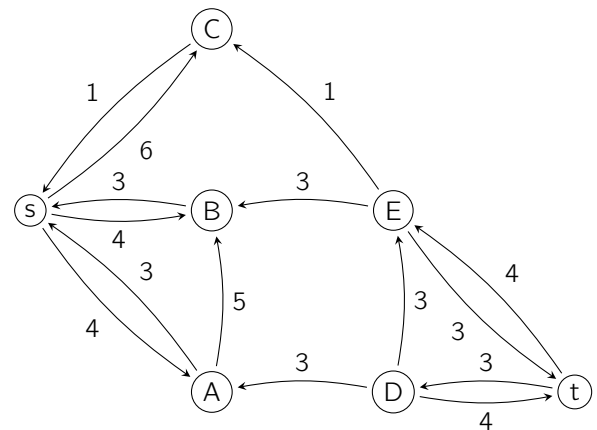
Schritt 5:  
Restnetzwerk:



Schritt 6:  
Nächstes Flussnetzwerk mit aktuellem Fluss:



Schritt 7:  
Restnetzwerk:





Der maximale Fluss hat den Wert: 7

---

**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:

**Lösung:** \_\_\_\_\_

\_\_\_\_\_

a) Die Tabelle C wird vom Algorithmus wie folgt gefüllt:

	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	3	3	3	3	3	3	3	3	3
2	↑ 0	← 0	← 3	← 3	← 3	← 3	← 9	9	12	12	12
3	0	0	3	4	4	7	↑ 9	← 9	← 12	← 13	13
4	0	2	3	5	6	7	9	11	12	↑ 14	← 15
5	0	2	3	5	6	7	9	11	12	14	↑ 15
6	0	2	3	5	6	7	9	11	12	14	↑ 15

Damit ergibt sich der maximale Wert 15 für den Fall, dass der 2., 3. und 4. Gegenstand mitgenommen werden.

b) Die Tabelle wird vom Algorithmus wie folgt gefüllt:

	∅	E	R	R	A	T	E	N
∅	0	0	0	0	0	0	0	0
D	0	0	0	0	0	0	0	0
A	0	0	0	0	1	1	1	1
T	0	0	0	0	1	2	2	2
E	0	↖ 1	← 1	1	1	2	3	3
N	0	1	↑ 1	1	1	2	3	4
S	0	1	↑ 1	1	1	2	3	4
T	0	1	↑ 1	1	1	2	3	4
R	0	1	2	↖ 2	← 2	2	3	4
U	0	1	2	2	↑ 2	2	3	4
K	0	1	2	2	↑ 2	2	3	4
T	0	1	2	2	2	↖ 3	3	4
U	0	1	2	2	2	↑ 3	3	4
R	0	1	2	3	3	↑ 3	3	4
E	0	1	2	3	3	3	↖ 4	4
N	0	1	2	3	3	3	4	↖ 5

Längste gemeinsame Teilsequenz: ERTEN

### Aufgabe 6 (Dynamische Programmierung):

(6+6+6=18 Punkte)

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

- $W: E \rightarrow (0, 1]$ , 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

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

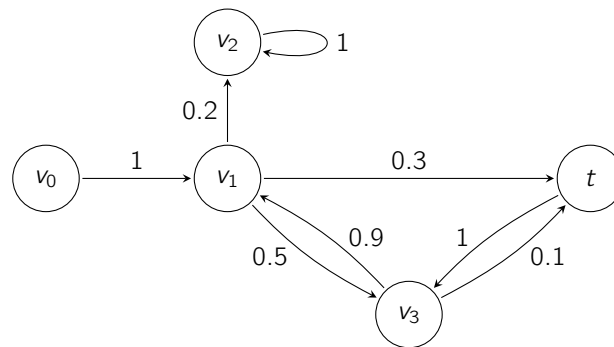
Sei  $t \in V$  ein beliebiger, fest gewählter Knoten. Für jedes  $v \in V$  und  $k \in \mathbb{N}$  betrachten wir die Menge der Pfade

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

Die Menge  $\Pi_v^k$  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  $\Pi_v^k$  sowie die zugehörigen Wahrscheinlichkeiten  $\Pr_t(\Pi_v^k)$ .

- i)  $\Pi_{v_3}^0 = \dots$  (enthält 0 Pfade)       $\Pr_t(\Pi_{v_3}^0) = \dots$
- ii)  $\Pi_{v_2}^3 = \dots$  (enthält 0 Pfade)       $\Pr_t(\Pi_{v_2}^3) = \dots$
- iii)  $\Pi_t^3 = \dots$  (enthält 1 Pfad)       $\Pr_t(\Pi_t^3) = \dots$
- iv)  $\Pi_{v_3}^3 = \dots$  (enthält 3 Pfade)       $\Pr_t(\Pi_{v_3}^3) = \dots$
- v)  $\Pi_{v_1}^4 = \dots$  (enthält 4 Pfade)       $\Pr_t(\Pi_{v_1}^4) = \dots$
- vi)  $\Pi_{v_0}^5 = \dots$  (enthält 4 Pfade)       $\Pr_t(\Pi_{v_0}^5) = \dots$

Hinweise:

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

b) Bestimmen Sie eine Rekursionsgleichung für die Pfadmengen  $\Pi_v^k$  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  $v \in V$ .

**Lösung:** \_\_\_\_\_

- a) i)  $\Pi_{v_3}^0 = \emptyset$        $\Pr(\Pi_{v_3}^0) = 0$   
 ii)  $\Pi_{v_2}^3 = \emptyset$        $\Pr(\Pi_{v_2}^3) = 0$   
 iii)  $\Pi_t^3 = \{t\}$        $\Pr(\Pi_t^3) = 1$   
 iv)  $\Pi_{v_3}^3 = \{v_3 t, v_3 v_1 t, v_3 v_1 v_3 t\}$        $\Pr(\Pi_{v_3}^3) = 0.1 + 0.9 \cdot 0.3 + 0.9 \cdot 0.5 \cdot 0.1 = 0.415$   
 v)  $\Pi_{v_1}^4 = \{v_1 t, v_1 v_3 t, v_1 v_3 v_1 t, v_1 v_3 v_1 v_3 t\}$   
 $\Pr(\Pi_{v_1}^4) = 0.3 + 0.5 \cdot \underbrace{(0.1 + 0.9 \cdot 0.3 + 0.9 \cdot 0.5 \cdot 0.1)}_{=\Pr(\Pi_{v_3}^3)} = 0.5075$   
 vi)  $\Pi_{v_0}^5 = \{v_0 v_1 t, v_0 v_1 v_3 t, v_0 v_1 v_3 v_1 t, v_0 v_1 v_3 v_1 v_3 t\}$        $\Pr(\Pi_{v_0}^5) = 1 \cdot \Pr(\Pi_{v_1}^4) = 0.5075$

b)

$$\Pi_v^k = \begin{cases} \{v\} & , \text{ falls } v = t \\ \emptyset & , \text{ falls } v \neq t, k = 0 \\ \{v\pi \mid (v, u) \in E \text{ und } \pi \in \Pi_u^{k-1}\} & , \text{ falls } v \neq t, k > 0 \end{cases}$$

c) Mit der Rekursionsgleichung aus b) erhalten wir

$$\begin{aligned}
 \Pr(\Pi_v^k) &= \begin{cases} \Pr(\{v\}) & , \text{ falls } v = t \\ \Pr(\emptyset) & , \text{ falls } v \neq t, k = 0 \\ \Pr(\{v\pi \mid (v, u) \in E \text{ und } \pi \in \Pi_u^{k-1}\}) & , \text{ falls } v \neq t, k > 0 \end{cases} \\
 &= \begin{cases} 1 & , \text{ falls } v = t \\ 0 & , \text{ falls } v \neq t, k = 0 \\ \sum_{(v,u) \in E} \sum_{\pi \in \Pi_u^{k-1}} \Pr(v\pi) & , \text{ falls } v \neq t, k > 0 \end{cases} \\
 &= \begin{cases} 1 & , \text{ falls } v = t \\ 0 & , \text{ falls } v \neq t, k = 0 \\ \sum_{(v,u) \in E} \sum_{\pi \in \Pi_u^{k-1}} \Pr(vu) \cdot \Pr(\pi) & , \text{ falls } v \neq t, k > 0 \end{cases} \\
 &= \begin{cases} 1 & , \text{ falls } v = t \\ 0 & , \text{ falls } v \neq t, k = 0 \\ \sum_{(v,u) \in E} \left( \Pr(vu) \cdot \sum_{\pi \in \Pi_u^{k-1}} \Pr(\pi) \right) & , \text{ falls } v \neq t, k > 0 \end{cases} \\
 &= \begin{cases} 1 & , \text{ falls } v = t \\ 0 & , \text{ falls } v \neq t, k = 0 \\ \sum_{(v,u) \in E} \left( W((v, u)) \cdot \Pr(\Pi_u^{k-1}) \right) & , \text{ falls } v \neq t, k > 0 \end{cases}
 \end{aligned}$$


---

### 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 Erläuterung.

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  $j$ :

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

Lösung:

---



---

a) \_\_\_\_\_

```
A(n): Integer -> Integer

C = Array[n+1]
C[1] = 1
C[2] = 2
C[3] = C[1] #= 1
for i = 5, i <= n+1; i += 2:
    C[i] = C[i-2]
    C[i-1] = (C[i] + C[i-2])/2
return C[n]
```

---

```
B(n,m): Integer, Integer -> Integer

# Das Array zum Cachen
# Beachte, dass für (n,m) auch Werte ausserhalb des Rechteckes
# berechnet werden.
C = Array[n+m, m]
# Erster Fall
for i = 1, i < n+m, i += 1:
    C[i,1] = i
# Zweiter Fall
for j=2, j<m, j+= 1:
    C[1,j] = 1

# Den vierten Fall
for j=2, j<m, j +=1:
    for i=j, i<n+m, i += 1:
        C[i,j] = C[i+1,j-1] + C[i-1,j-1]
        # Und eventuell auch den dritten Fall, aber nur, wenn
        # dieser auch in C reinpasst
        if i <= n:
            C[j,i] = C[i,j]
```

---

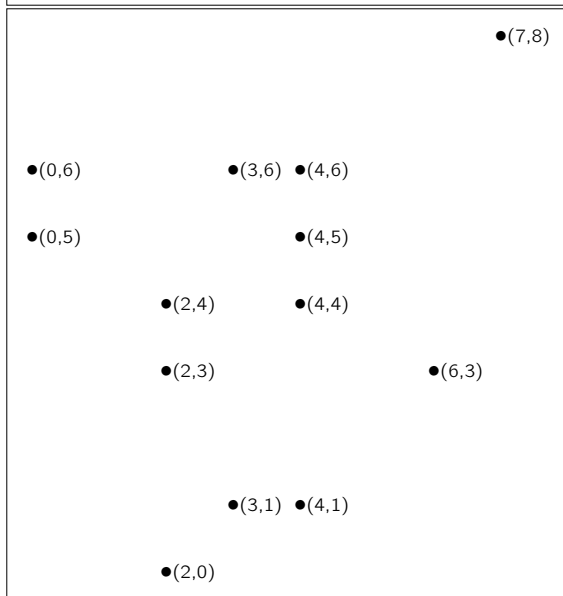
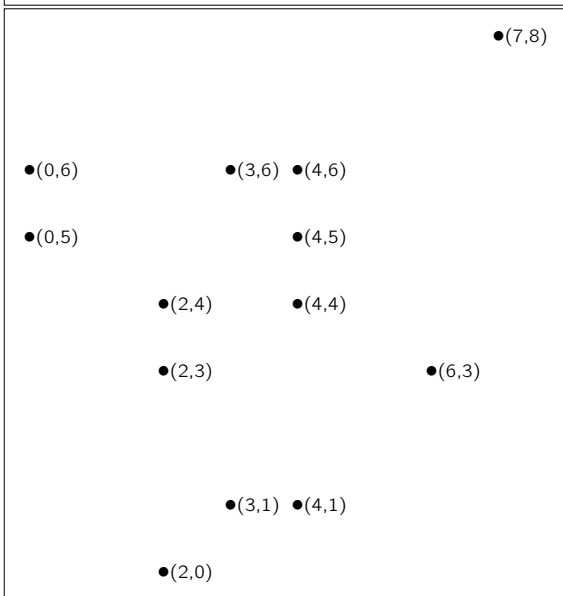
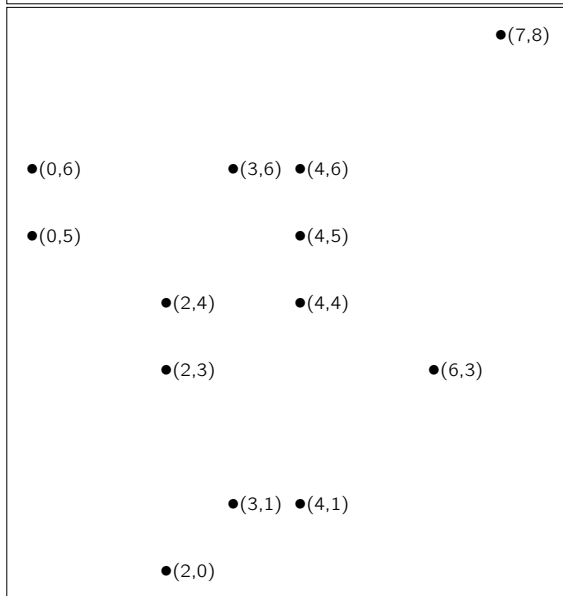
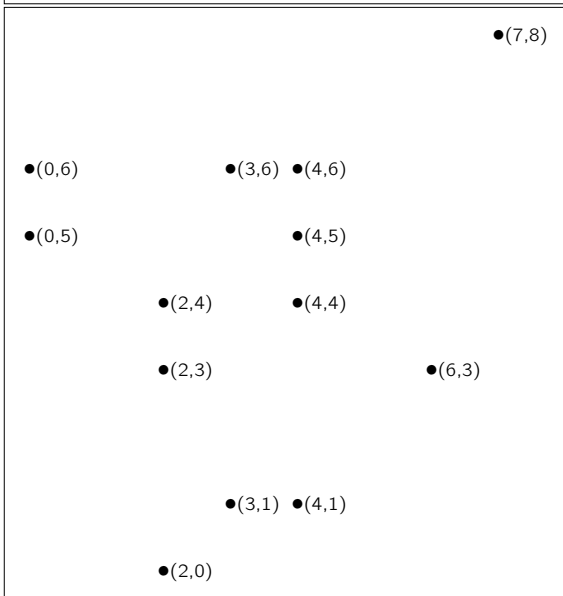
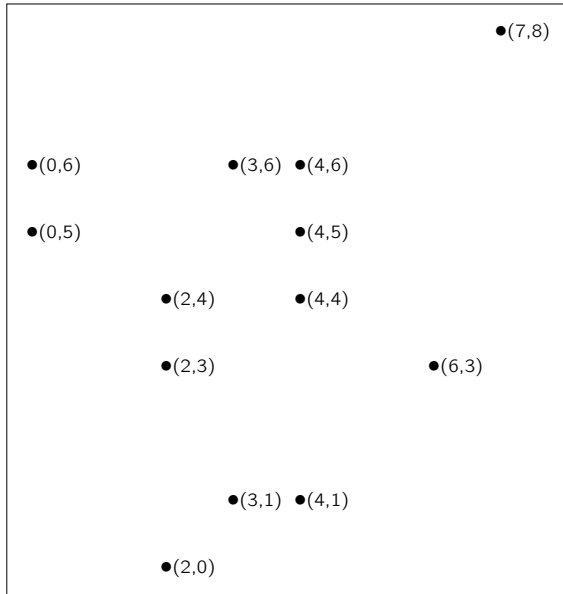
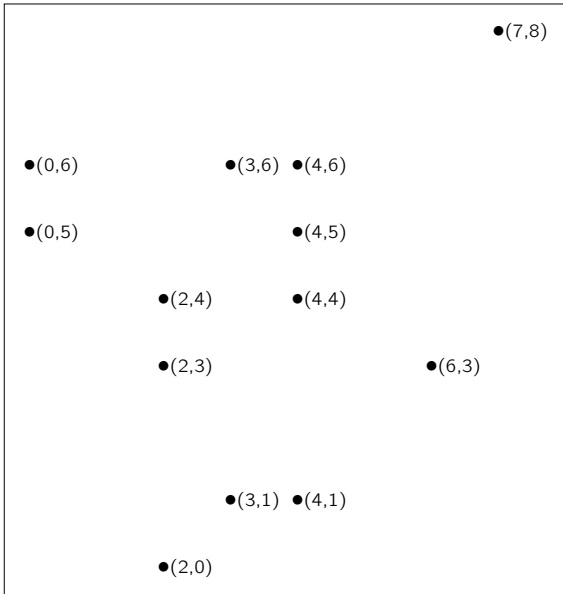
### Aufgabe 8 (Graham Scan):

(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.

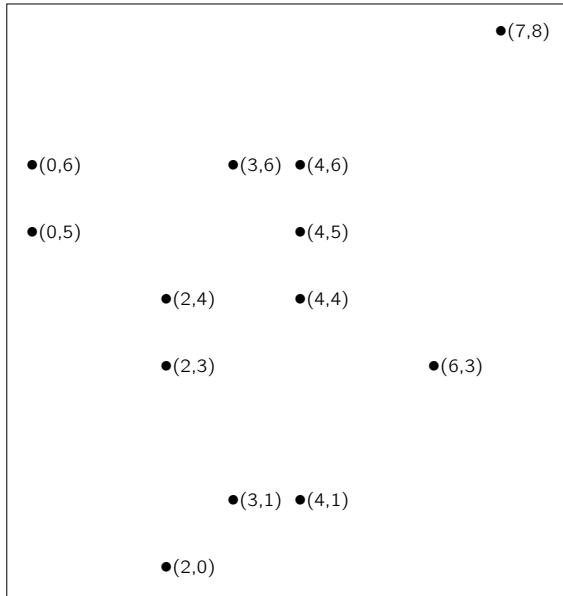
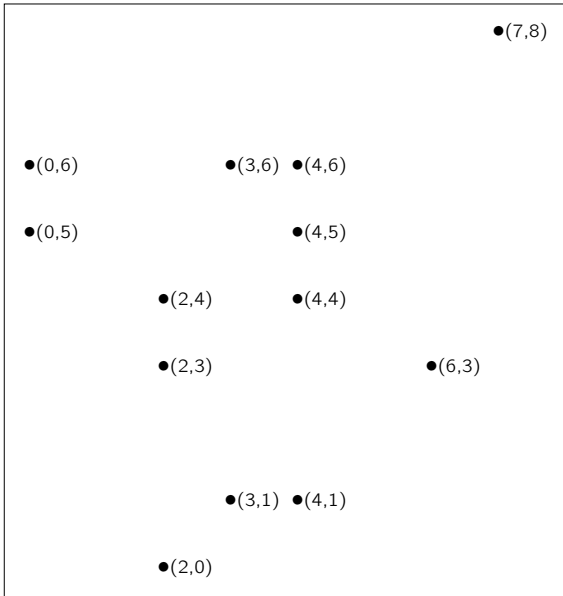
Hinweise:

- Für diese Aufgabe sind die Inhalte der Vorlesung "Algorithmische Geometrie" vom Montag, den 9.7. relevant.
- Nicht alle der unten aufgeführten Koordinatensysteme werden benötigt.



<p style="text-align: right;">●(7,8)</p> <p>●(0,6)            ●(3,6) ●(4,6)</p> <p>●(0,5)            ●(4,5)</p> <p>                 ●(2,4)    ●(4,4)</p> <p>                 ●(2,3)            ●(6,3)</p> <p>                         ●(3,1) ●(4,1)</p> <p>                 ●(2,0)</p>	<p style="text-align: right;">●(7,8)</p> <p>●(0,6)            ●(3,6) ●(4,6)</p> <p>●(0,5)            ●(4,5)</p> <p>                 ●(2,4)    ●(4,4)</p> <p>                 ●(2,3)            ●(6,3)</p> <p>                         ●(3,1) ●(4,1)</p> <p>                 ●(2,0)</p>
<p style="text-align: right;">●(7,8)</p> <p>●(0,6)            ●(3,6) ●(4,6)</p> <p>●(0,5)            ●(4,5)</p> <p>                 ●(2,4)    ●(4,4)</p> <p>                 ●(2,3)            ●(6,3)</p> <p>                         ●(3,1) ●(4,1)</p> <p>                 ●(2,0)</p>	<p style="text-align: right;">●(7,8)</p> <p>●(0,6)            ●(3,6) ●(4,6)</p> <p>●(0,5)            ●(4,5)</p> <p>                 ●(2,4)    ●(4,4)</p> <p>                 ●(2,3)            ●(6,3)</p> <p>                         ●(3,1) ●(4,1)</p> <p>                 ●(2,0)</p>
<p style="text-align: right;">●(7,8)</p> <p>●(0,6)            ●(3,6) ●(4,6)</p> <p>●(0,5)            ●(4,5)</p> <p>                 ●(2,4)    ●(4,4)</p> <p>                 ●(2,3)            ●(6,3)</p> <p>                         ●(3,1) ●(4,1)</p> <p>                 ●(2,0)</p>	<p style="text-align: right;">●(7,8)</p> <p>●(0,6)            ●(3,6) ●(4,6)</p> <p>●(0,5)            ●(4,5)</p> <p>                 ●(2,4)    ●(4,4)</p> <p>                 ●(2,3)            ●(6,3)</p> <p>                         ●(3,1) ●(4,1)</p> <p>                 ●(2,0)</p>





Lösung: \_\_\_\_\_

\_\_\_\_\_

