Tutoraufgabe 1 (Dynamische Programmierung):

Gegeben sei ein Rucksack mit **maximaler Tragkraft** 15 sowie 5 **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 $w_0 = 4$, $w_1 = 7$, $w_2 = 1$, $w_3 = 6$ und $w_4 = 5$ und die **Werte** $c_0 = 3$, $c_1 = 10$, $c_2 = 2$, $c_3 = 4$ und $c_4 = 5$. Geben Sie zudem die vom Algorithmus bestimmte Tabelle `C` und zeigen Sie anhand der Tabelle, welche Gegenstände mitgenommen werden um diesen maximalen Wert zu erreichen.

Lösung:

Die Tabelle `C` wird vom Algorithmus wie folgt gefüllt:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	↑ 0	0	3	3	3	3	3	3	3	3	3	3	3	3
2	0	0	↑ 0	← 0	← 3	← 3	← 3	← 10	← 10	← 10	10	13	13	13	13	13
3	0	2	2	2	3	5	5	10	12	↑ 12	← 12	13	15	15	15	15
4	0	2	2	2	3	5	5	10	12	12	↑ 12	13	15	15	16	16
5	0	2	2	2	3	5	7	10	12	12	↑ 12	← 13	← 15	← 17	← 17	← 17

Damit ergibt sich der maximale Wert 17 für den Fall, dass der 1., 2. und 4. Gegenstand mitgenommen werden. Dies lässt sich von der Tabelle wie folgt ablesen: Wenn die i-te Zeile einen Pfeil nach links enthält dann wird der $(i-1)$-te Gegenstand mitgenommen. Die Pfeile zeigen dabei an wie der folgende Algorithmus durch die Tabelle läuft:

```
int i = n; int j = M;
while (i > 0 && j > 0) {
    if (C[i][j] != C[i-1][j])
        for (int k = 0; k < w[i-1]; k++) j--;
    i--;
}
```

Aufgabe 2 (Dynamische Programmierung): (4* Punkte)

Gegeben sei ein Rucksack mit **maximaler Tragkraft** 15 sowie 5 **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 $w_0 = 6$, $w_1 = 9$, $w_2 = 1$, $w_3 = 3$ und $w_4 = 4$ und die **Werte** $c_0 = 2$, $c_1 = 8$, $c_2 = 1$, $c_3 = 4$ und $c_4 = 5$. Geben Sie zudem die vom Algorithmus bestimmte Tabelle `C` und zeigen Sie anhand der Tabelle, welche Gegenstände mitgenommen werden um diesen maximalen Wert zu erreichen.

Lösung:

Die Tabelle `C` wird vom Algorithmus wie folgt gefüllt:

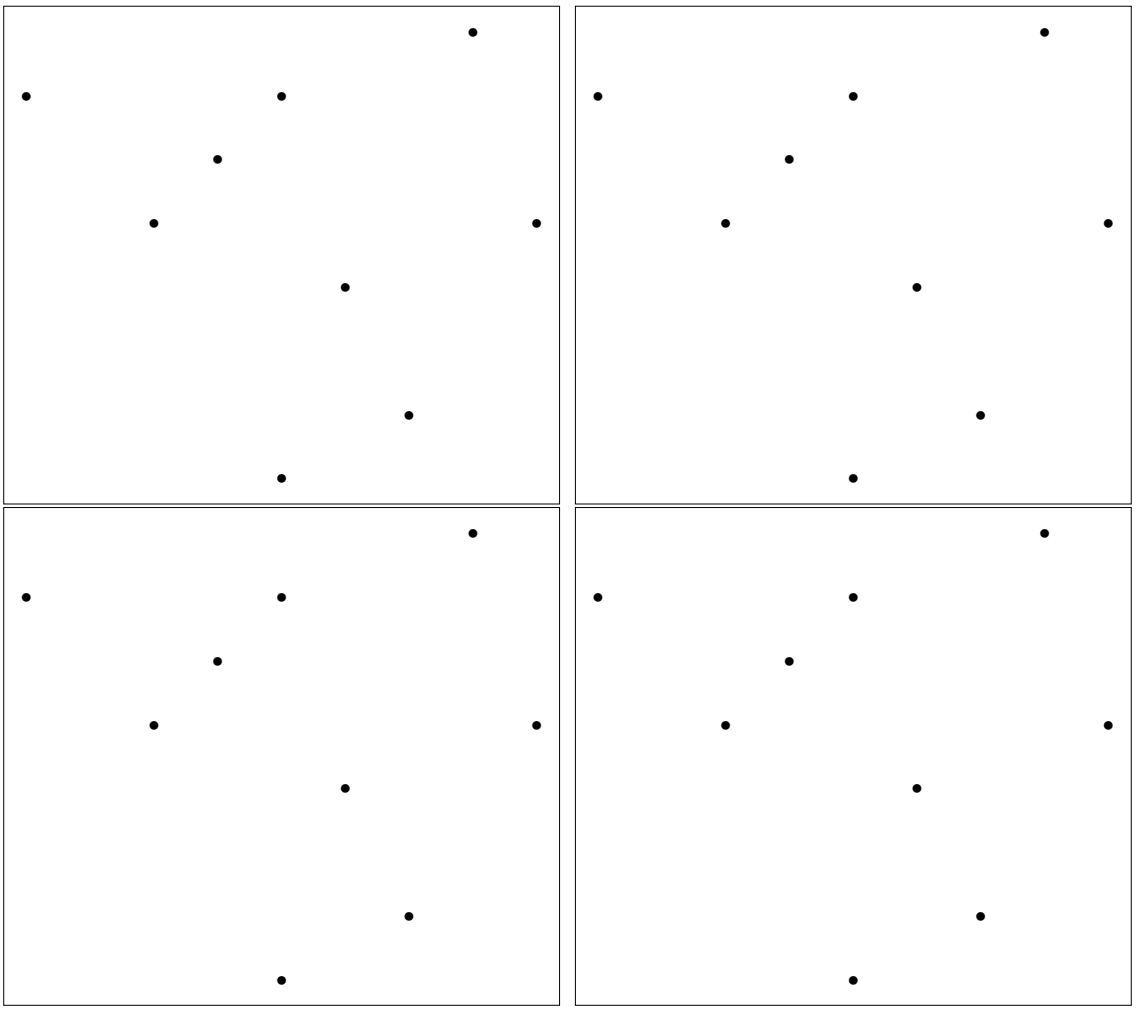
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	↑ 0	0	0	0	0	2	2	2	2	2	2	2	2	2	2
2	0	↑ 0	← 0	← 0	← 0	← 0	← 2	← 2	← 2	← 8	← 8	8	8	8	8	10
3	0	1	1	1	1	1	2	3	3	8	↑ 9	← 9	9	9	9	10
4	0	1	1	4	5	5	5	5	5	8	9	↑ 9	12	13	13	13
5	0	1	1	4	5	6	6	9	10	10	10	↑ 10	← 12	← 13	← 14	← 14

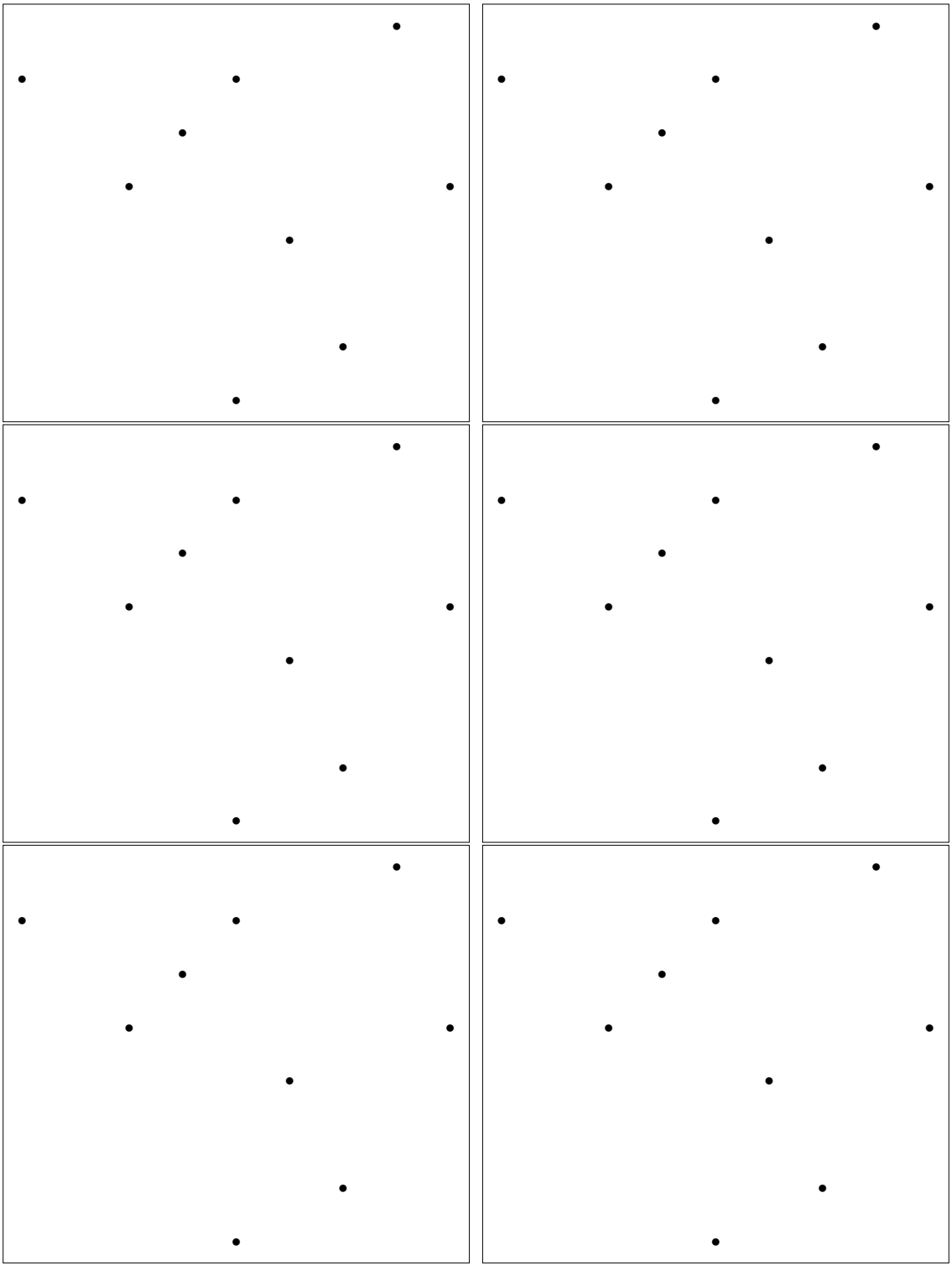
Damit ergibt sich der maximale Wert 14 für den Fall, dass der 1., 2. und 4. Gegenstand mitgenommen werden. Dies lässt sich von der Tabelle wie folgt ablesen: Wenn die i-te Zeile einen Pfeil nach links enthält dann wird der $(i-1)$-te Gegenstand mitgenommen. Die Pfeile zeigen dabei an wie der folgende Algorithmus durch die Tabelle läuft:

```
int i = n; int j = M;
while (i > 0 && j > 0) {
    if (C[i][j] != C[i-1][j])
        for (int k = 0; k < w[i-1]; k++) j--;
    i--;
}
```

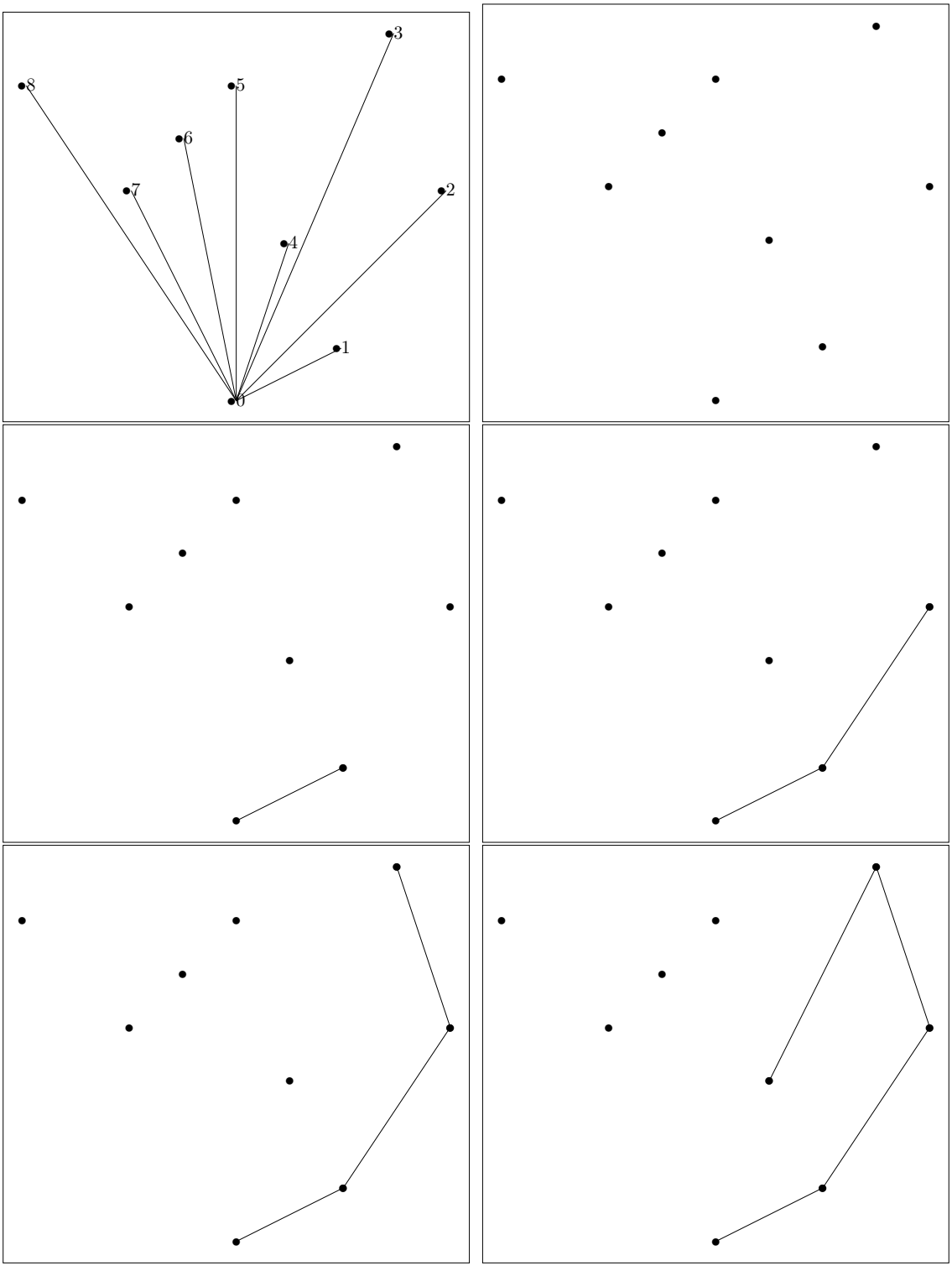
Tutoraufgabe 3 (Geometrische Algorithmen):

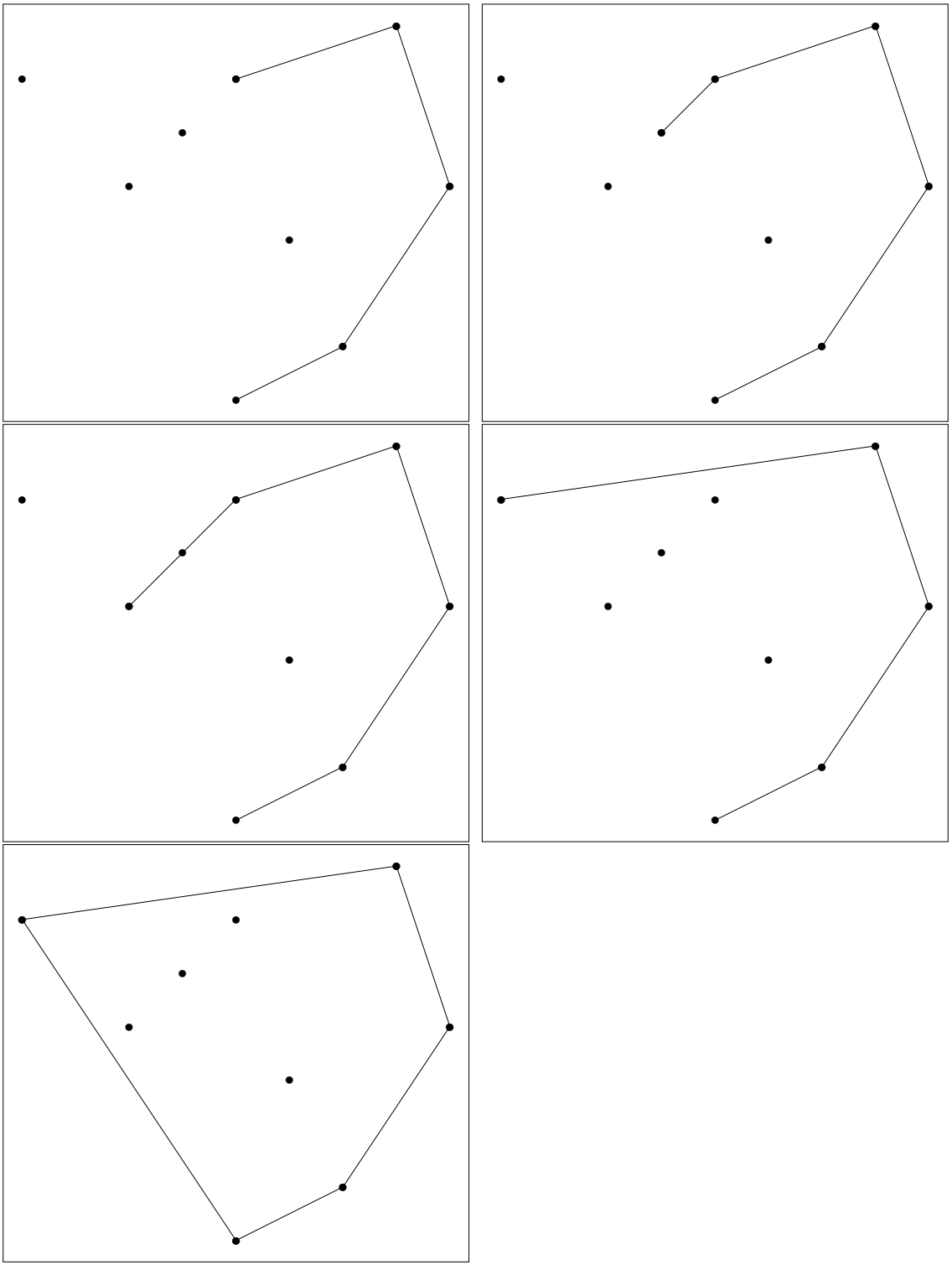
Berechnen Sie die konvexe Hülle der folgenden Punktmenge. Benutzen Sie dafür Grahams' Scan wie in der Vorlesung vorgestellt und geben Sie die Teilschritte nach jeder Iteration (also nach jedem neu hinzugefügten Punkt) an. Markieren Sie Punkte, die vom Algorithmus nicht betrachtet werden.





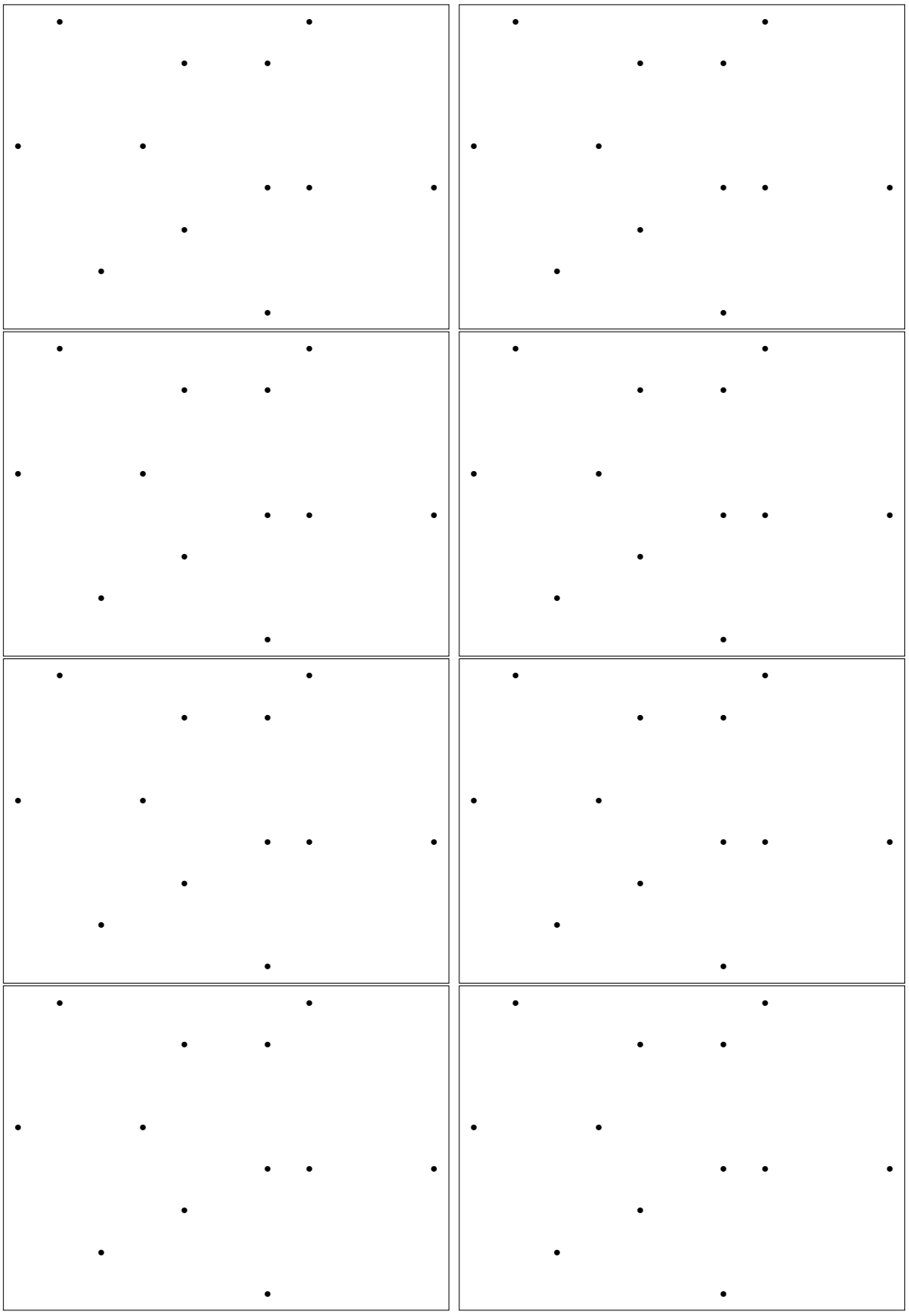
Lösung:

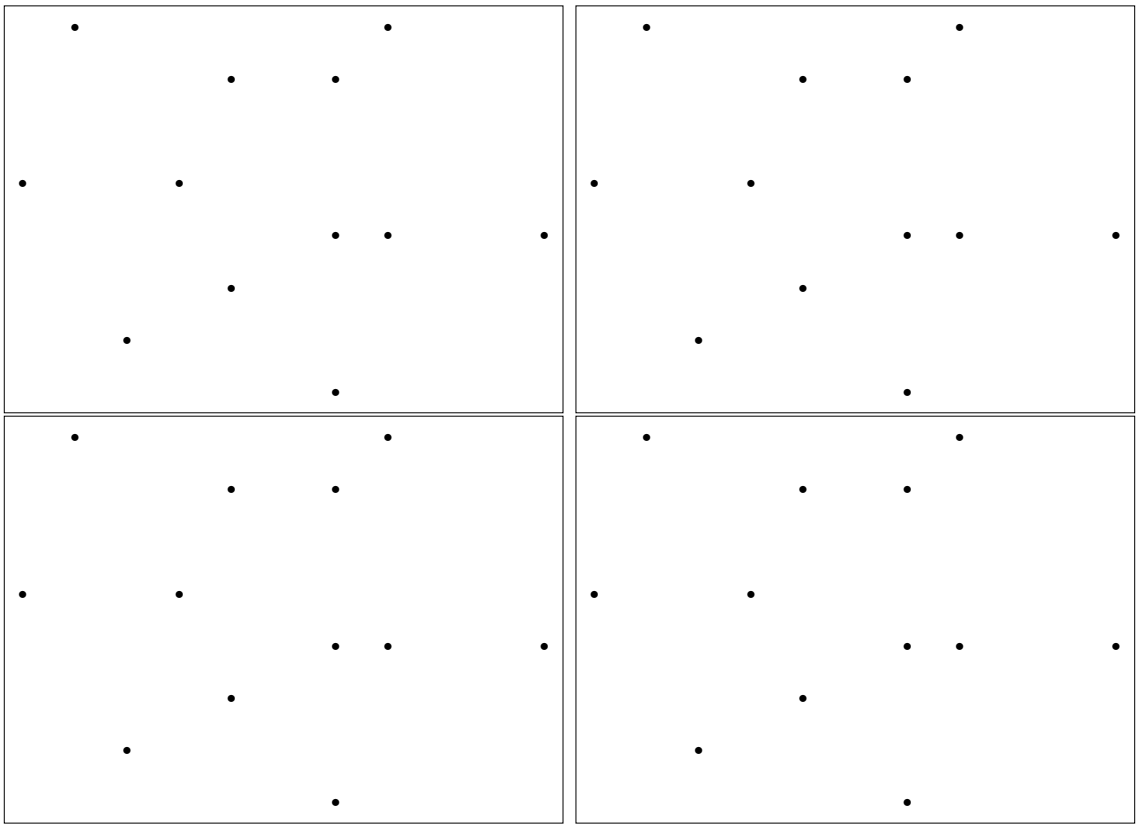

8
5
3
6
7
2
4
1
0



Aufgabe 4 (Geometrische Algorithmen): (3* 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. Markieren Sie Punkte, die vom Algorithmus nicht betrachtet werden.





Lösung: