Prof. Dr. Ir. Joost-Pieter Katoen

Christian Dehnert, Friedrich Gretz, Benjamin Kaminski, Thomas Ströder

Tutoraufgabe 1 (Longest Common Subsequence):

Bestimmen Sie die *längste gemeinsame Teilsequenz* der Sequenzen `AACHEN` und `ANDERNACH`. Benutzen Sie hierfür den in der Vorlesung vorgestellten Algorithmus mit dynamischer Programmierung und füllen Sie die folgende Tabelle aus.

Lösung:

Die Tabelle wird vom Algorithmus wie folgt gefüllt:

	∅	A	N	D	E	R	N	A	C	H
∅	0	0	0	0	0	0	0	0	0	0
A	0	↖ 1	← 1	← 1	← 1	← 1	← 1	1	1	1
A	0	1	1	1	1	1	1	↖ 2	2	2
C	0	1	1	1	1	1	1	2	↖ 3	3
H	0	1	1	1	1	1	1	2	3	↖ 4
E	0	1	1	1	2	2	2	2	3	↑ 4
N	0	1	2	2	2	2	3	3	3	↑ 4

Also erhalten wir die Sequenz `AACH` als längste gemeinsame Teilsequenz der Sequenzen `AACHEN` und `ANDERNACH`.

Tutoraufgabe 2 (Dynamische Programmierung):

a) Schreiben Sie eine Funktion, welche die Länge der Longest Common Subsequence (LCS) für zwei gegebene Zeichensequenzen berechnet. Die Zeichensequenzen seien dabei der Einfachheit halber `int` Arrays `seq1` und `seq2` mit den Längen `l1` and `l2`. Demnach soll Ihre Funktion die folgende Signatur haben:

```
int lcs(int seq1[], int l1, int seq2[], int l2)
```

b) Betrachten Sie folgendes Szenario. Ihnen wird eine Klausur gestellt, welche n Aufgaben umfasst. Jede dieser Aufgaben hat eine Punktzahl p_i und eine von Ihnen geschätzte Zeit t_i, die Sie zum Lösen der Aufgabe benötigen ($1 \leq i \leq n$). Geben Sie die maximale Punktzahl, welche Sie in T Zeiteinheiten erreichen können, als Rekursionsgleichung an (inklusive der passenden Argumente).

Lösung:

a)
```
int lcs(int seq1[], int l1, int seq2[], int l2) {
    int L[l1+1,l2+1];
    for (int i = 0; i <= l1; i++) {
        L[i,0] = 0;
    }
    for (int j = 0; j <= l2; j++) {
        L[0,j] = 0;
    }
    for (int i = 1; i <= l1; i++) {
        for (int j = 1; j <= l2; j++) {
            if (seq1[i] == seq2[j]) {
```

```
                L[i,j] = L[i - 1, j - 1] + 1;
            } else if (L[i - 1, j] > L[i, j - 1]) {
                L[i,j] = L[i - 1, j];
            } else {
                L[i,j] = L[i, j - 1];
            }
        }
    }
    return L[l1,l2];
}
```

b) Wir definieren die Funktion P in Abhängigkeit von der Anzahl i an betrachteten Aufgaben und der zur Verfügung stehenden Zeit T.

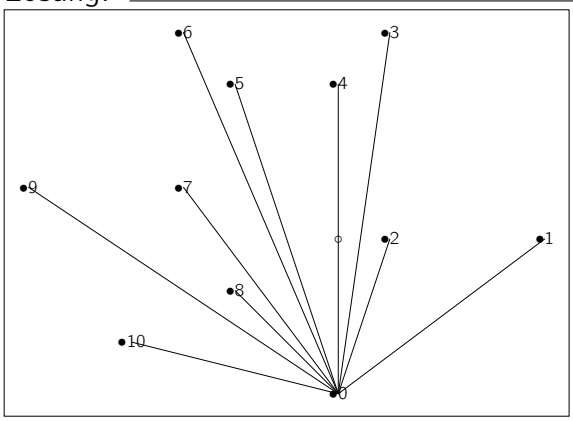
$$P(i,T) = \begin{cases} 0 & \text{falls } i = 0 \\ max(P(i-1,T), P(i-1,T-t_i) + p_i) & \text{falls } T \geq t_i \\ P(i-1,T) & \text{sonst} \end{cases}$$

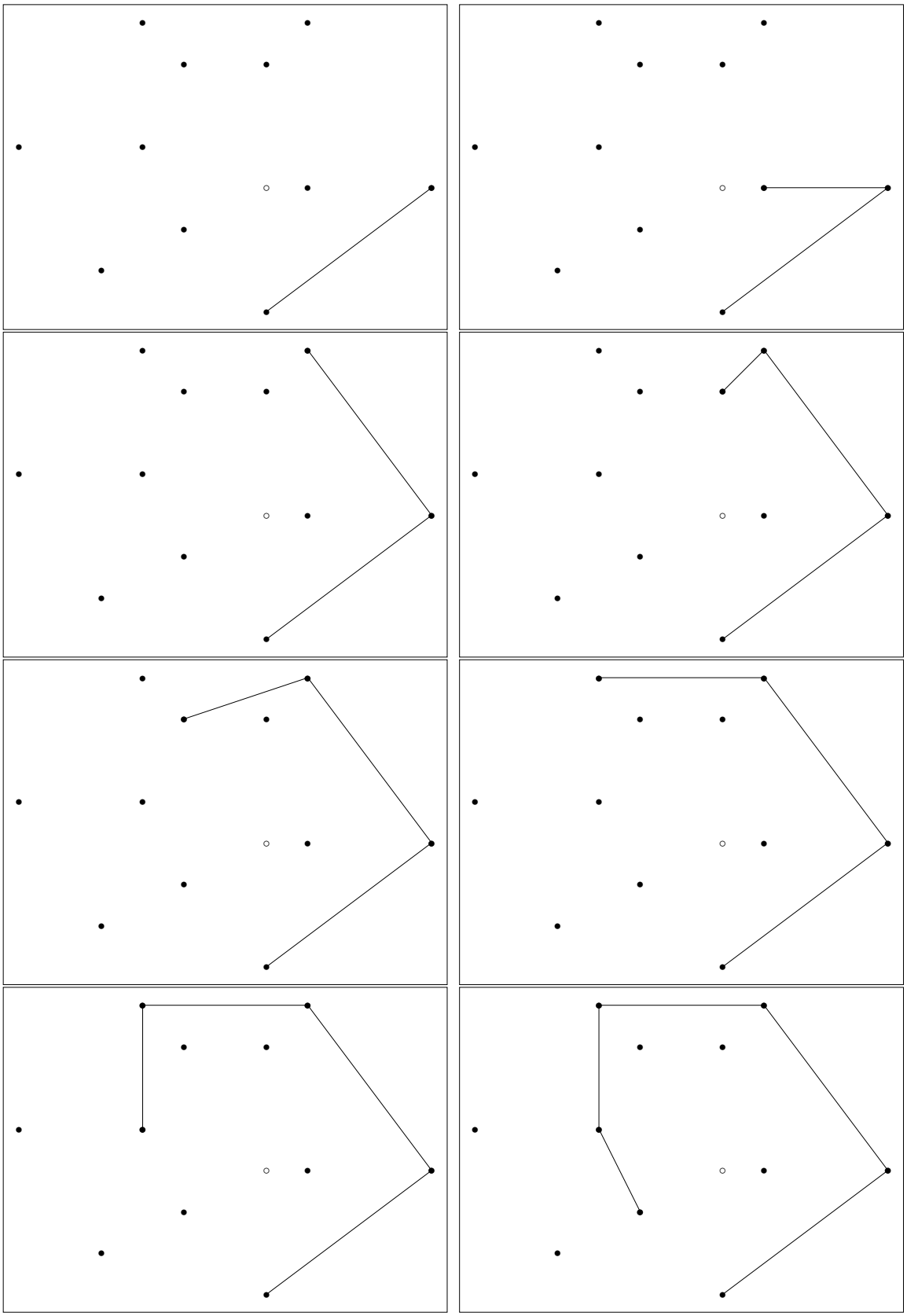
Die maximal erreichbare Punktzahl ist damit $P(n,T)$.

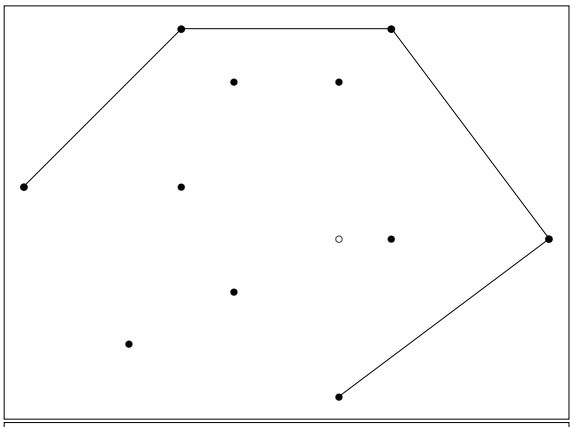
Tutoraufgabe 3 (Graham-Scan):

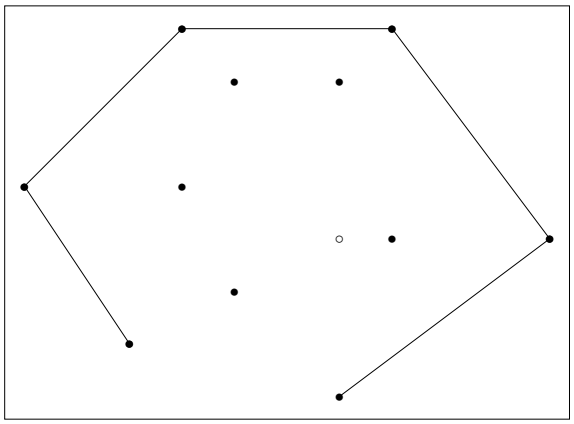
Berechnen Sie die konvexe Hülle der folgenden Punktmenge. Benutzen Sie dafür *Grahams Scan* wie *in der Vorlesung* vorgestellt und geben Sie die Teilschritte *nach jeder Iteration* (also nach jedem neu hinzugefügten Punkt) an. Umkreisen Sie die Punkte, die vom Algorithmus in der Iterationsschleife nicht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

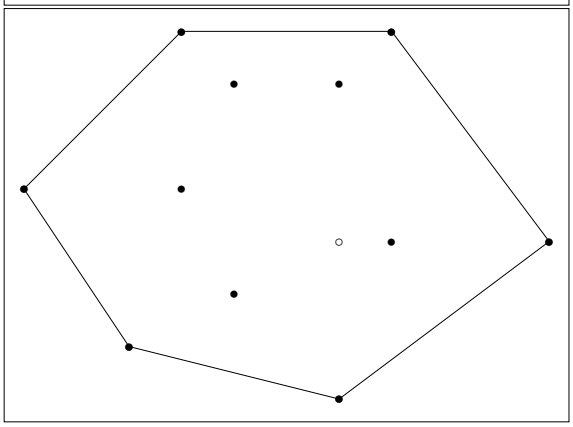
Lösung:











Aufgabe 4 (Rucksackproblem): (4 Punkte)

Gegeben sei ein Rucksack mit *maximaler Tragkraft* 10 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 $w_1 = 5$, $w_2 = 2$, $w_3 = 1$, $w_4 = 3$ und $w_5 = 4$ und die *Werte* $c_1 = 9$, $c_2 = 8$, $c_3 = 6$, $c_4 = 7$ und $c_5 = 5$. Geben Sie zudem die vom Algorithmus bestimmte Tabelle `c` und die mitzunehmenden Gegenstände an.

Lösung:

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	0	0	0	9	9	9	9	9	9
2	↑ 0	← 0	← 8	8	8	9	9	17	17	17	17
3	0	6	↑ 8	← 14	14	14	15	17	23	23	23
4	0	6	8	↑ 14	← 14	← 15	← 21	21	23	23	24
5	0	6	8	14	14	15	↑ 21	← 21	← 23	← 23	← 26

Damit ergibt sich der maximale Wert 26 für den Fall, dass der 2., 3., 4. und 5. Gegenstand mitgenommen werden.

Aufgabe 5 (Dynamische Programmierung): (4 + 4 + 3 + 3 = 14 Punkte)

Gegeben sei ein $(n \times m)$-Schachbrett (d. h. n Zeilen, m Spalten) mit $n, m \geq 4$. Auf jedem Feld (i, j) (d. h. Zeile i, Spalte j) liegt eine Münze mit dem Wert w_{ij}.

Ein Spieler spielt nun folgendes Spiel auf diesem Schachbrett: Der Spieler möchte einen *Springer* von einem beliebigen Feld in der linkesten Spalte zu einem beliebigen Feld in der rechtesten Spalte bewegen. Dabei darf er nur für Springer legale Züge tätigen, welche den Springer nach rechts bewegen. Immer wenn der Springer ein Feld besucht, nimmt der Spieler die auf dem Feld liegende Münze auf. Das Startfeld gilt auch als vom Springer besucht.

Ziel des Spielers ist es, den Wert der aufgenommenen Münzen zu maximieren. Eine optimale Strategie soll nun mittels dynamischer Programmierung gefunden werden.

a) Stellen Sie eine Rekursionsgleichung d_{ij} auf, mit deren Hilfe für jedes Feld (i, j) des Schachbrettes der maximale Wert aufgenommener Münzen ermittelt werden kann, die auf einem Pfad von einem beliebigen Startfeld zum Feld (i, j) liegen. Dabei müssen die Pfade natürlich den obigen Spielregeln entsprechen.

b) Gegeben sei ein Schachbrett mit folgenden Münzen:

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

Führen Sie dynamische Programmierung für das obige Schachbrett gemäß der von Ihnen aufgestellten Rekursionsgleichung durch. Stellen Sie dazu, wie in der Vorlesung, eine Tabelle auf.

c) Beschreiben Sie, wie man mittels Backtracking aus der bei der dynamischen Programmierung erstellten Tabelle das optimale Start- und Zielfeld und den optimalen Pfad über das Schachbrett ermitteln kann.

d) Ermitteln Sie mittels Backtracking auf der in Teil (b) erstellten Tabelle den optimalen Pfad über das Schachbrett.

Lösung:

a) Wir definieren zunächst folgende Menge:

$$\begin{aligned} p_{ij} \;=\; & \{(i+1, j-2) \mid i+1 \leq n, j-2 \geq 1\} \cup \{(i-1, j-2) \mid i-1 \geq 1, j-2 \geq 1\} \\ & \cup \{(i+2, j-1) \mid i+2 \leq n, j-1 \geq 1\} \cup \{(i-2, j-1) \mid i+2 \leq n, j-1 \geq 1\} \end{aligned}$$

Intuitiv gesprochen enthält die Menge p_{ij} alle Positionen von denen aus der Spieler mit dem Springer auf legale Weise auf das Feld (i, j) gelangen kann.

Unter Verwendung von p_{ij} lautet die Rekursionsformel dann wie folgt:

$$S[i, j] \;=\; \begin{cases} w_{i1}, & \text{falls } j = 1, \\ \max\limits_{(k,\ell) \in p_{ij}} S[k, \ell] + w_{ij}, & \text{andernfalls.} \end{cases}$$

b) Die mittels dynamischer Programmierung erstellte Tabelle lautet wie folgt:

	1	2	3	4	5	6	7	8
1	0	4	6	9	12	12	13	14
2	0	2	6	6	9	13	14	19
3	2	6	6	11	11	13	13	16
4	0	3	6	9	10	13	16	16
5	3	5	8	9	12	11	16	14
6	0	2	5	9	11	14	14	16
7	1	4	8	11	10	14	11	17
8	1	3	5	8	11	14	14	17

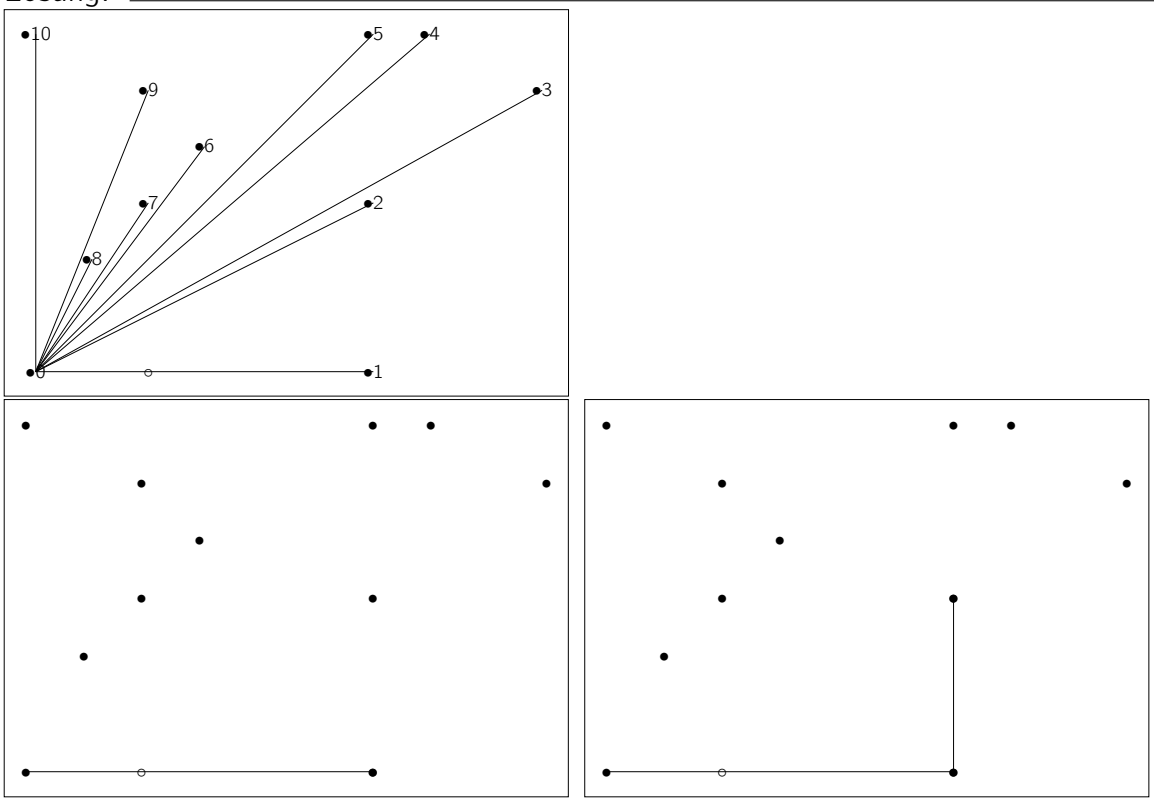
c)
1. Starte in Spalte m auf Feld mit maximalem Wert $S[i, m]$. Setze $(k, \ell) = (i, m)$. Füge (k, ℓ) vorne dem gesuchten Pfad hinzu.
2. Suche Rückwärts von welchem Feld (g, j) der Springer auf Feld (k, ℓ) gesprungen sein muss. Setze $(k, \ell) = (g, j)$. Füge (k, ℓ) vorne dem gesuchten Pfad hinzu.
3. Falls $k \neq 1$ gehe zu Schritt 2.

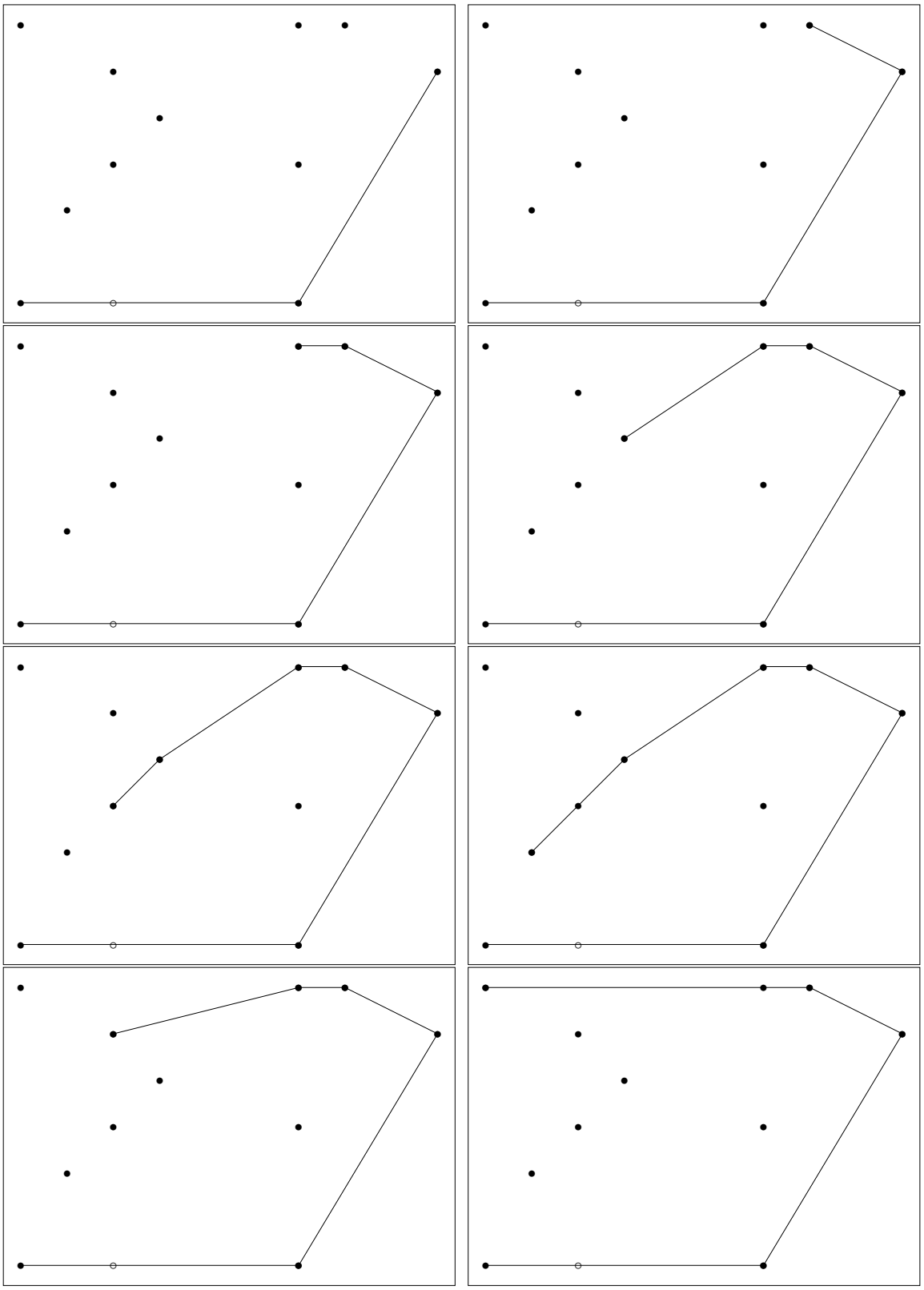
d) Ein optimaler Pfad ist: $(5, 1)$, $(3, 2)$, $(5, 3)$, $(7, 4)$, $(6, 6)$, $(4, 7)$, $(2, 8)$.

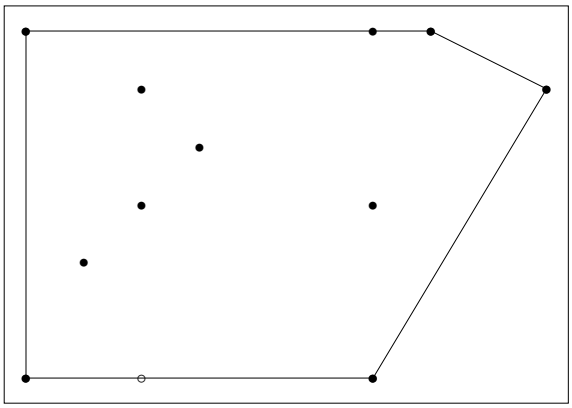
Aufgabe 6 (Graham-Scan): (3 Punkte)

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. Umkreisen Sie die Punkte, die vom Algorithmus in der Iterationsschleife nicht betrachtet werden.

Lösung:







Aufgabe 7 (Klausur): (20* Punkte)

a)* Entwerfen Sie eine Klausur (Aufgabenstellung und Musterlösung) für das Fach Datenstrukturen und Algorithmen mit einem zeitlichen Umfang von 120 Minuten. Die Klausur soll die wesentlichen Konzepte der Vorlesung abprüfen, einen angemessenen Schwierigkeitsgrad haben sowie klar und verständlich formuliert sein.

Beachten Sie, dass Plagiarismus zu einer Bewertung mit 0 Punkten führt.

Lösung:

a)* Siehe Probeklausur auf der Webseite.