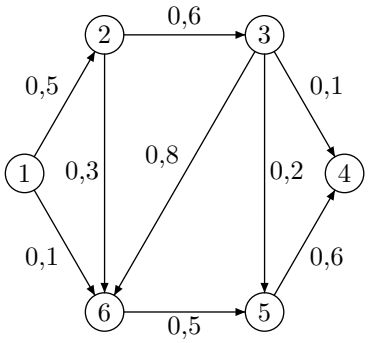
Aufgabenblatt zur 14. Übung

Zeitraum: 31.01. bis 04.02.2011

1. Aufgabe: (AGS 9.37)

Der gewichtete Graph $G = (V, E, c)$ sei durch folgende graphische Darstellung gegeben:



Es soll für den Graph G das *Zuverlässigkeitsproblem* gelöst werden.

(a) Geben Sie den geeigneten Semiring für die Lösung dieses Problems und die modifizierte Adjazenzmatrix mA_G von G an. Semiring: $\big($, , , , $\big)$

$$mA_G = \begin{pmatrix} & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \end{pmatrix}$$

(b) Berechnen Sie mithilfe des Aho-Algorithmus die Matrizen $D_G^{(1)}$, $D_G^{(2)}$ und $D_G^{(3)}$. Schreiben Sie hierbei nur die Matrixelemente auf, die sich gegenüber der jeweiligen Vorgängermatrix geändert haben. Nutzen Sie dafür die Notation

(Anfangsknoten; Endknoten; Gewicht). Wenn es keine Änderungen gegenüber der jeweiligen Vorgängermatrix gibt, geben Sie bitte eine *Begründung* an.

- $D_G^{(1)}$:
- $D_G^{(2)}$:
- $D_G^{(3)}$:

(c) Geben Sie die erste Zeile der Ergebnismatrix D_G des Aho-Algorithmus an.

- erste Zeile: ; ; ; ; ;

(d) Auf welchen Maximalwert kann die Kante von Knoten 3 zum Knoten 5 erhöht werden, ohne dass sich die Ergebnismatrix D_G ändert?

- Maximalwert:

2. Aufgabe: (AGS 10.12*)

Ein Dieb, der einen Safe ausraubt, findet in ihm n Typen von Gegenständen unterschiedlicher Größe und unterschiedlichen Wertes, hat aber nur einen (kleinen) Rucksack der Größe W zur Verfügung, um die Gegenstände zu tragen.
Das Rucksack–Problem besteht nun darin, diejenige Kombination von Gegenständen zu finden, die der Dieb für seinen Rucksack auswählen sollte, so dass der Gesamtwert der von ihm geraubten Gegenstände maximal wird.

Nun eine abgeleitete, spezifizierte Aufgabenstellung:
Gegeben seien n Objekttypen $O_1..O_n$, ihre Bewertungen p_i und ihre Gewichte w_i, wobei $i \in \{1,\ldots,n\}$ und $w_i, p_i \in I\!N$. Des Weiteren legen wir fest, dass von jedem Objekttyp O_i, $i \in \{1,\ldots,n\}$, unbegrenzt Exemplare zur Verfügung stehen. Das Gesamtgewicht des Rucksacks darf den Wert W, $W \in I\!N$, nicht überschreiten.

(a) Überlegen Sie sich unter Zuhilfenahme der Dynamischen Programmierung ein Lösungsverfahren für die Ermittlung der in den Rucksack aufzunehmenden Objektexemplare, so dass die Summe der Bewertungen maximiert und das vorgegebene Gesamtgewicht W nicht überschritten wird.

(b) Schreiben Sie ein C–Programm zur Lösung dieses Problems.

Nutzen Sie als Testfall:
Objekttypen: A, B, C, D, E mit den Gewichten $w_1 = 3, w_2 = 4, w_3 = 7, w_4 = 8, w_5 = 9$ und den Bewertungen $p_1 = 4, p_2 = 5, p_3 = 10, p_4 = 11, p_5 = 13$. Das zulässige Gesamtgewicht des Rucksack sei $W = 17$.

Hinweis zur Lösung: Konstruieren Sie die Lösung, indem Sie einerseits stufenweise die zum Einladen zugelassenen Objekttypen erweitern und andererseits für eine zugelassene Typmenge jeweils das Rucksackgesamtgewicht (von 1 beginnend) bis zum Endwert W schrittweise erhöhen und für jedes Rucksackgewicht eine suboptimale Beladung ermitteln.

3. Aufgabe: (AGS 10.13*)

Im Schach darf der Springer in einem Zug immer zwei Felder geradeaus und dann ein Feld links oder rechts davon gesetzt werden. Je nach Position des Springers ergeben sich somit bis zu acht verschiedene Zugmöglichkeiten. Diese seien mit a bis h bezeichnet und in Abbildung 1 dargestellt (wobei der Springer hier in der Mitte steht).

Betrachten Sie nun Abbildung 2. Finden Sie mit Hilfe des Backtracking-Verfahrens einen Weg (Zugfolge), um mit einem Springer von Feld 1 zum Feld 7 zu gelangen. Hierbei ist zu beachten, dass:

- schraffierte Felder nicht benutzt werden dürfen,
- kein Feld auf einem Weg mehrfach betreten werden darf.

Zeichnen Sie dazu den optimierten Berechnungsbaum bis zur ersten Lösung. Gehen Sie bei der Wegeerweiterung immer in der Reihenfolge a bis h vor, wenn es in einer Situation mehrere Zugmöglichkeiten gibt.

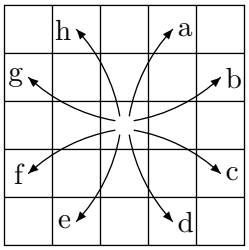


Abbildung 1

21	▒	▒	▒	▒	22
15	16	17	18	19	20
11	12	▒	▒	13	14
9	10	▒	▒	▒	▒
6	7	▒	8	▒	▒
1	2	▒	3	4	5

Abbildung 2

Zusatzaufgabe 1: (AGS 10.17*)

Gegeben sei folgendes Labyrinth, und es soll ein nichtzirkulärer Weg von Platz 3 nach Platz 34 gefunden werden.

42	41	40	39	38	37	36
35	34	*X*	32	*X*	30	29
28	*X*	26	25	24	23	22
X	20	*X*	18	17	16	15
14	13	12	11	10	9	8
X	6	*X*	*X*	3	2	1

Plätze, die mit einem X gekennzeichnet sind, dürfen hierbei nicht betreten werden. Von einem Platz darf die Fortsetzung des Weges nach links, nach oben, nach rechts und nach unten erfolgen (diagonale Richtungen sind also verboten).

(a) Geben Sie mit Hilfe des BACKTRACK- Algorithmus, beginnend mit dem Startplatz 3, den optimierten Berechnungsbaum der zulässigen Wegeerweiterungen bis zur 1. Lösung an.
Grundsätzlich soll gelten:

Erweiterungen des Weges von einem aktuellen Platz müssen immer in der Reihenfolge nach links, nach oben, nach rechts, nach unten getätigt werden.
Eine Erweiterung ist zulässig, wenn der Erweiterungsplatz
– nicht außerhalb einer Randlinie liegt,
– nicht durch ein X gesperrt ist und
– nicht auf dem bisherigen Weg liegt.

(b) Setzen Sie die Suche bis zum Auffinden der 2. Lösung fort; geben Sie dazu die notwendigen Erweiterungen im Berechnungsbaum der Aufgabenstellung (a) an. Gibt es noch weitere Lösungen? Begründen Sie kurz Ihre Antwort.

Zusatzaufgabe 2: (AGS 10.10*)

Sei M eine Menge von n Elementen. Eine *Permutation* von M ist dann eine (beliebige) Anordnung aller n Elemente von M (entspricht einem n-Tupel, wo jedes Element von M genau einmal vorkommt). Die Anzahl der bildbaren Permutationen ist bekanntlich n-Fakultät.

Die Menge $\{1, 2, 3\}$ besitzt zum Beispiel die Permutationen: 123, 132, 213, 231, 312, 321.

(a) Entwickeln Sie einen Algorithmus mit Hilfe des Backtracking–Prinzips, der alle Permutationen von M ermittelt und auflistet. Definieren Sie hierzu eine rekursive Vorschrift, bei der in jedem Rekursionsschritt versucht wird, eine bereits vorhandene Teilfolge von verschiedenen Zahlen um eine weitere Zahl zu ergänzen, die von allen Zahlen der Teilfolge verschieden ist. Nutzen Sie für die Reihefolge der Erweiterungen von Teillösungen die Ordnungsrelation von $\mathbb{N}$.

(b) Schreiben Sie den Berechnungsbaum bis zur 4. Lösung auf.

(c) Schreiben Sie zu Ihrem Algorithmus ein C-Programm, das nach Eingabe von $n \in \mathbb{N}$ alle Permutationen von $\{1, \ldots, n\}$ berechnet und ausgibt.

Hinweis: Begrenzen Sie n zunächst auf maximal 9, um die Aufgabe besser handhabbar zu machen.