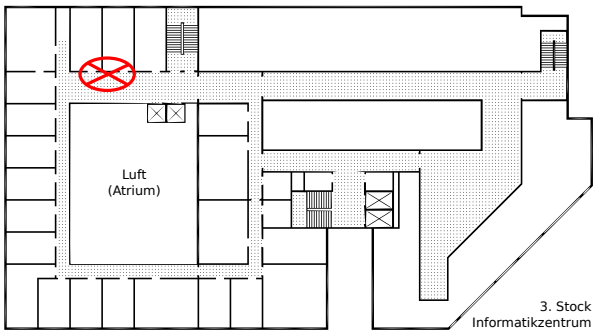
Algorithmen und Datenstrukturen 2 **SoSe 2023**
Prof. Dr. Sándor Fekete **Abgabe:** 28.06.2023
Dr. Phillip Keldenich **Rückgabe:** ab 04.07.2023

Hausaufgabenblatt 4

Abgabe der Lösungen bis zum 28.06.2023 um 13:30 Uhr im Hausaufgabenschrank bei Raum IZ 337 (siehe Skizze rechts). Es werden nur mit einem dokumentenechten Stift (kein Rot!) geschriebene Lösungen gewertet. **Bitte die Blätter zusammenheften und vorne deutlich mit Namen, Matrikel-, Übungs- und Gruppennummer versehen!**



Hausaufgabe 1 (Modellierung): **(8 Punkte)**

Eine wichtige Technik zum Lösen schwerer Probleme in der Praxis ist die sogenannte *Modellierung*. Dabei transformiert man eine Instanz I eines schweren Problems $\mathcal{A}$, an deren Lösung man interessiert ist, effizient in eine Instanz I' eines anderen schweren Problems $\mathcal{S}$. Die Instanz I' soll dabei zur Instanz I äquivalent sein in dem Sinne, dass sie eine Lösung hat genau dann, wenn I eine Lösung hat, und sich eine Lösung für I' effizient in eine Lösung für I transformieren lässt.

In der Regel verwendet man dabei Probleme $\mathcal{S}$, für die es bereits Programme gibt, die Instanzen des Problems in der Praxis gut und mit akzeptabler Geschwindigkeit lösen. Eine mögliche Option für $\mathcal{S}$ ist das Problem SAT aus der großen Übung.

Wir wollen uns in dieser Aufgabe mit dem verallgemeinerten Sudoku-Problem beschäftigen. Dabei besteht die Eingabe aus einer Zahl $k \geq 2$ und einem zweidimensionalen Feld $\mathcal{F}$ der Größe $k^2 \times k^2$ (normales Sudoku hat $k = 3$). Jede Zelle $z_{x,y}$ des gegebenen Feldes enthält einen Wert aus $\{\bot\} \cup \{1, \ldots, k^2\}$.

Das Feld ist unterteilt in eine Menge $\mathcal{B}$ von Blöcken aus je $k \times k$ Zellen, die analog zu den 3×3-Blöcken von Sudoku das gesamte Feld überdecken. Jede Zelle $z_{x,y}$ des Feldes ist Teil genau eines Blocks $B(x, y) \in \mathcal{B}$. Weiterhin ist jede Zelle $z_{x,y}$ Teil genau einer Zeile R_y und einer Spalte C_x. Zu einem gegebenen k und einem gegebenen Feld $\mathcal{F}$ wollen wir wissen, ob es eine Möglichkeit gibt, das Feld korrekt auszufüllen, also im gegebenen Feld $\mathcal{F}$ alle Einträge $\bot$ durch Zahlen aus $\{1, \ldots, k^2\}$ zu ersetzen, sodass in jedem Block, in jeder Spalte und in jeder Zeile alle Werte aus $\{1, \ldots, k^2\}$ genau einmal vorkommen, ohne dabei die gegebenen Zahlen zu ändern.

Gib eine Transformation an, der aus einer verallgemeinerten Sudoku-Instanz eine SAT-Instanz macht, die lösbar ist genau dann, wenn die verallgemeinerte Sudoku-Instanz lösbar ist. Die Transformation sollte effizient (mit einer polynomiell von k abhängigen Laufzeit) berechenbar sein (kurze Begründung reicht).
Hinweis: SAT erlaubt nur wahr/falsch-Variablen. Eine mögliche Lösung verwendet für ein reguläres Sudoku-Feld $81 \times 9 = 729$ Variablen.

Hausaufgabe 2 (Wiederholung: Dynamic Programming): **(6 Punkte)**
Wende das Dynamic Program für MAXIMUM KNAPSACK auf folgende Instanz an.

i	1	2	3	4
z_i	3	4	2	7
p_i	1	4	1	4

und $Z = 12$.

Hausaufgabe 3 (INTEGER KNAPSACK): **(3+3 Punkte)**
Betrachte die KNAPSACK-Variante INTEGER KNAPSACK, bei der wir jedes Objekt i beliebig (ganzzahlig) oft mitnehmen dürfen. Wir suchen also Werte $b_i \in \mathbb{Z}_{\geq 0}$, sodass die Gewichtsbedingung $\sum_i b_i z_i \leq Z$ eingehalten wird und der Gewinn $\sum_i b_i p_i$ maximiert wird. Wir wollen in dieser Aufgabe einen Branch-and-Bound-Algorithmus für dieses Problem entwickeln.

Die fraktionale Variante des Problems, bei der wir alle Objekte beliebig (nicht-negativ) oft mitnehmen dürfen (d.h. $b_i \in \mathbb{R}_{\geq 0}$), kann durch einen Greedy-Algorithmus in $O(n)$ Zeit gelöst werden (wir nehmen einfach soviele Einheiten des Objekts mit dem besten Preis pro Gewicht mit, wie wir können).

a) Beschreibe kurz einen Greedy-Algorithmus, der in $O(n \log n)$ Zeit eine ganzzahlige, zulässige Lösung zurückgibt, die nicht erweitert werden kann, d.h. das Hinzufügen eines beliebigen Elements macht die Lösung unzulässig.

b) Wandle den Pseudocode des Branch-and-Bound-Algorithmus für MAXIMUM KNAPSACK so ab, dass er für INTEGER KNAPSACK funktioniert.

②

i	1	2	3	4
z_i	3	4	2	7
p_i	1	4	1	4

$z = 12$

i \ x	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1	1	1	1
2	0	0	0	1	4	4	4	5	5	5	5	5	5
3	0	0	1	1	4	4	5	5	5	6	6	6	6
4	0	0	1	1	4	4	5	5	5	6	6	8	8

① Jede Zelle $z_{x,y}$ enthält einen Wert aus $\{\perp\} \cup \{1, \dots, k^2\}$.

Die Zelle $z_{x,y}$ Teil genau einer Zeile R_y und einer Spalte C_x.

Sei Aussagelogische Variable $Z^k_{i,j}$

$i := R_y$ also jede Zeile $\quad j := C_x$ also jede Spalte und

k Ziffern.

i) ~~Förd~~ Fordere für jede Zeile i und alle Ziffern k,

- Ziffer k mindestens einmal erscheint und erreiche dies mit dem Disjunktionsterm,

$$D^k_i := \bigvee_{j=1}^{9} Z^k_{i,j}$$ ✓

und k wir nicht zweimal vergeben, d.h.

$$Z^k_{i,j} \rightarrow \neg Z^k_{i,j^*}$$

Für alle Spalten $j \neq j^*$ mit $1 \leq j \leq j^* \leq 9$ und alle Ziffern k.

Umwandle der obige Ausdruck in KNF

$$\left(Z^k_{i,j} \rightarrow \neg Z^k_{i,j^*}\right) \equiv \left(\neg Z^k_{i,j} \vee \neg Z^k_{i,j^*}\right)$$ ✓

ii) Jede Ziffer k kommt genau einmal in jeder Zeile vor:

$$\text{Zeile} := \bigwedge_{i=1}^{9} \bigwedge_{k=1}^{9} \left(D^k_i \wedge \bigwedge_{j=1}^{8} \bigwedge_{j^*=j+1}^{9} \left(\neg Z^k_{i,j} \vee \neg Z^k_{i,j^*}\right)\right)$$ ✓

iii) Analog definiere KNFs in Spalte und Teilmatrix.

Für die Spalten und 3×3 Teilmatrizen, dass jede Ziffer in jeder Spalte, bzw. in jeder Teilmatrix genau einmal vorkommt. ✓

iv) Keine Zelle $z_{x,y}$ zwei oder mehr Ziffern enthält, d.h.

$$\neg z_{i,j}^{k} \vee \neg z_{i,j}^{k^*}$$

gilt für alle $1 \leq i,j \leq 9$ und alle verschiedenen k und k^*.
Mit der folgenden KNF fordere, dass in jeder Zelle $z_{x,y}$ höchstens eine Ziffer steht

$$\text{Einfach} := \bigwedge_{i=1}^{9} \bigwedge_{j=1}^{9} \bigwedge_{k=1}^{8} \bigwedge_{k^*=k+1}^{9} \left(\neg z_{i,j}^{k} \vee \neg z_{i,j}^{k^*} \right).$$

(✓)

v) Ist die Zelle $z_{x,y}$ mit dem Ziffern k bereits gesetzt, füge dann Disjunktionsterm $z_{i,j}^{k}$ hinzu.

Definiere KNF dann Bereit-da.

Insgesamt ist die Konjunktion dieser Disjunktionsterme für alle bereits gesetzten Zelle. (v)

$$\text{Sudoku} := \text{Zeile} \wedge \text{Spalte} \wedge \text{Teilmatrix} \wedge \text{Einfach} \wedge \text{Bereit-da}$$

Betrachte: Die Korrektheit:

Es gibt für diese Sudoku-Instanz eine Lösung genau dann wenn die obige Formel erfüllbar ist.

Beweis mit Kontraposition:

Sei die Formel unerfüllbar

$\Leftrightarrow$ Es existiert keine Belegung, die die Formel erfüllt.

d.h. Für jede Belegung gibt es mindestens eine Klausel mit dem boolschen Wert „false".

$\Leftrightarrow$ Für mögliche Eingabe der Werte trifft mindestens eine der 5 Aussagen nicht zu:

1) In jeder Zelle steht mindestens ein Wert.
2) In jeder Zelle gibt es maximal ein Wert.
3) In jeder Zeile wiederholt sich kein Wert.
4) In jeder Spalte wiederholt sich kein Wert.

c) In jedem Block wiederholt sich ebenfalls kein Wert.
d.h. diese Instanz hat keine mögliche Lösung.

Begründe die Laufzeit der Transformation!

Jetzt erstelle eine Sudoku-Instanz aus SAT-Formal

Sei das

$k^2 \times k^2 \times k^2 = k^6$ Variable ⇒ Insgesamt wird $O(k^6)$ Zeit gebraucht.

für	für	für jede
jede	jede	Zahl
Spalte	Zeile	mit
		dem
		Wert
		1 bis k^2

D.h. die Transformation braucht polynomielle Zeit.

7/

(3)

a) 1. Function Greedy Integer $(z_1, \dots, z_n, Z, p_1, \dots, p_n)$

2. Sortiere Objekte nach z_i/p_i aufsteigend, ~~dies~~
Dies ergibt die Permutation $\pi(1), \dots, \pi(n)$ } $O(n \log n)$

3. for $i = 1$ to n do

4. $x_{\pi(i)} = \left\lfloor \frac{Z}{z_{\pi(i)}} \right\rfloor$

5. $Z := Z - x_{\pi(i)} z_{\pi(i)}$ } $O(n)$

6. return $(x_1, \dots, x_n)$

Insgesamt hat das Algorithmus Laufzeit $O(n \log n)$.

b) procedure BRANCH-AND-BOUND (ℓ)

if $\left(\sum_{j=1}^{\ell-1} b_j z_j > Z\right)$ then return △ unzulässig

Compute $L := LB(b_1, \ldots, b_{\ell-1})$

if $L > P$ then $P := L$ △ Lösungswert verbessert

if $(\ell > n)$ then return △ Blatt im Baum erreicht

$U := UB(b_1, \ldots, b_{\ell-1})$

if $(U > P)$ then

Hier musst du alle Werte $0 \leq b_i \leq k$ durchgehen mit k ~~gleich~~ gleich dem maximal zulässigen Wert.

$b_\ell := 1$; $++b_j$; BRANCH-AND-BOUND (ℓ)

$b_\ell := 0$; BRANCH-AND-BOUND (ℓ+1)

Ist das nicht auch exakt ~~der gleich~~ original Algo ?

7-7-23 Klausur 5

① Gegeben: $k^2 \times k^2$ - Sudoku - Feld $\mathcal{F}$ $\Rightarrow$ k^4 Felder $z_{x,y}$.

Für jedes Feld $z_{x,y}$, k^2 Variablen, $v_{x,y,i}$

Zeilen R_y

Spalten C_x

Blöcke $B(x,y) \in \mathcal{B}$

1) Jede Zelle enthält mindestens einen Wert:

$$\forall z_{x,y} \in \mathcal{F} : (v_{x,y,1} \vee v_{x,y,2} \vee \dots \vee v_{x,y,k^2})$$

2) Jede Zelle enthält maximal einen Wert:

$$\forall z_{x,y} \in \mathcal{F} \quad \forall i<j : (\overline{v_{x,y,i}} \vee \overline{v_{x,y,j}})$$

3) Jede Zelle enthält jeden Wert genau einmal:

$$\forall R_y \quad \forall z_{x,y} \neq z_{w,y} \in R_y \quad \forall i : (\overline{v_{x,y,i}} \vee \overline{v_{w,y,i}})$$

4) Für Spalte:

$$\forall c_x \quad \forall z_{x,y} \neq z_{x,w} \in C_x \quad \forall i : \left(\overline{v_{x,y,i}} \vee \overline{v_{x,w,i}}\right)$$

5) Für Block:

$$\forall B(x,y) \in B \quad \forall z_{a,b} \neq z_{c,d} \in B(x,y) \quad \forall i : \left(\overline{v_{a,b,i}} \vee \overline{v_{c,d,i}}\right)$$

11A3)

a) 1. Sortiere Objekte nach $\frac{z_i}{p_i}$ aufsteigend.

2. Iteriere durch alle Objekte

2.1. Nimm Objekt i so häufig wie möglich:

$$\Rightarrow \left\lfloor \frac{Z - \sum_{j=1}^{i-1} z_j}{z_j} \right\rfloor =: x_i$$

b) Eingabe: z_i's , p_i's, Z

Globale Variable: P

procedure: Branch & Bound $(L, b_1, \dots, b_{L-1})$

$R = Z - \sum_{i=1}^{l-1} b_i z_i$

$L = LB(b_1, \dots, b_{L-1})$

if $P < L$

$P := L$

$U = UB(b_1, \dots, b_{L-1})$

if $U \leq P$

return

for j from 0 upto $\left\lfloor \frac{P}{z_L} \right\rfloor$

$b_L = j$

Branch & Bound $(L+1, b_1, \dots, b_L)$