

Řešení vybraných příkladů
k předmětu
PA163 Programování s omezujícími podmínkami

Hana Rudová

Fakulta informatiky, Masarykova univerzita

15. září 2019

Poděkování

Řešení příkladů jsou převzata z domácích úkolů studentů předmětu. Všem děkuji za pěkná řešení, která jsou zde použita.

Příklady 1. – zadání

1. Uvedený CSP převedte pomocí duálního problému na binární CSP. Máte zadány proměnné A, B, C, D, E, F , jejich doména je $\{0, 1\}$ a omezení jsou

$$c_1 : A + B + F = 1$$

$$c_2 : A - C + D = 1$$

$$c_3 : D + E - F > 0$$

$$c_4 : B + E - F = 0$$

2. Popište rozdíl mezi algoritmy AC-1 a AC-3 a ukažte podrobně, jak tyto algoritmy pracují na následujícím příkladu.

$$A \in \{1, \dots, 10\}, \quad B \in \{1, \dots, 10\}, \quad C \in \{1, \dots, 10\}, \quad B < A, \quad A \leq C + 4$$

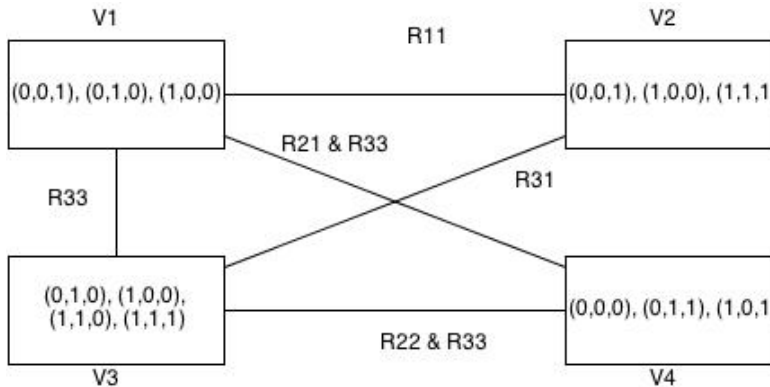
3. Ukažte, jak vypadají propagační pravidla konzistence mezi pro omezení $X = C + Y$ za předpokladu, že C je konstanta a X, Y jsou doménové proměnné. Ukažte průběh propagací (včetně aplikace Vámi navržených propagačních pravidel) při použití konzistence mezi v následujícím příkladu.

$$X \in \{1, \dots, 20\}, \quad Y \in \{1, \dots, 5\} \cup \{15, \dots, 20\}, \quad Y = 3 + X$$

Nastal by v příkladu nějaký rozdíl, pokud bychom místo konzistence mezi použili hranovou konzistenci?

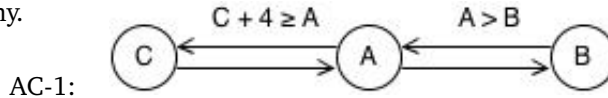
Příklady 1. – řešení

1.příklad



2.příklad

U algoritmu AC-1 dochází k revizi všech hran pokaždé, kdy se doména některé z proměnných změní. U algoritmu AC-3 se hrany k zrevidování berou z fronty. Do té se na začátku vloží hrany všechny a po každém průchodu se vloží pouze ty hrany, které mohou být zasaženy některou z přírodných změn domény.



$$A > B, A \leq C + 4 : (\underline{1} \dots \underline{10}, 1 \dots 10, 1 \dots 10) \xrightarrow{AB} (2 \dots 10, \underline{1} \dots \underline{10}, 1 \dots 10) \xrightarrow{BA} (2 \dots \underline{10}, 1 \dots 9, 1 \dots 10) \xrightarrow{AC} (2 \dots 10, 1 \dots 9, \underline{1} \dots \underline{10}) \xrightarrow{CA} (2 \dots 10, 1 \dots 9, 1 \dots 10)$$

Následně jsou ještě jednou zrevidovány všechny hrany, nedojde k žádné změně a tím algoritmus končí.

AC-3: Fronta by postupně vypadala takto:

$$(AB, BA, AC, CA) \rightarrow (BA, AC, CA) \rightarrow (AC, CA) \rightarrow (CA) \rightarrow ()$$

3.příklad

Pro případ $X = Y + C$ bych propagační pravidla zvolil následovně:

$$\min(X) \geq \min(Y) + C \tag{1}$$

$$\max(X) \leq \max(Y) + C \tag{2}$$

$$\min(Y) \geq \min(X) - C \tag{3}$$

$$\max(Y) \leq \max(X) - C \tag{4}$$

Pro případ kdy $Y \in \{1, \dots, 20\}$, $X \in \{1, \dots, 5\} \cup \{15, \dots, 20\}$, $X = 3 + Y$ (pro přehlednost byly proměnné přejmenovány: $X/Y, Y/X$) by propagace pomocí těchto pravidel vypadala následovně:

$$(\min(X), \max(X)), (\min(Y), \max(Y)) = (1, 20), (1, 20) \xrightarrow{(1)}$$

$$(\underline{4}, 20), (1, 20) \xrightarrow{(2)} (4, 20), (1, 20) \xrightarrow{(3)} (4, 20), (1, 20) \xrightarrow{(4)} (4, 20), (\underline{1}, 17)$$

Tedy $X \in \{4, 5\} \cup \{15, \dots, 20\}$ a $Y \in \{1, \dots, 17\}$.

Při použití hranové konzistence by z domény proměnné Y zmizely také hodnoty 3 až 11, jelikož ty by v doméně proměnné X neměly oporu. Konečný výsledek by tedy byl $X \in \{4, 5\} \cup \{15, \dots, 20\}$ a $Y \in \{1, 2\} \cup \{12, \dots, 17\}$

Příklady 2 – zadání

Ukažte, jak vypadá graf stavového prostoru pro řešení problému

- proměnné A, B, C mohou nabývat hodnot 0, 1, 2, 4
- omezení: $c_1 : A = B * 2$, $c_2 : B > C$

algoritmem

1. backtrackingu
2. kontroly dopředu (forward checking)
3. pohledu dopředu (look ahead) bez iniciální konzistence

při uspořádání proměnných A, B, C . V řešení uveďte pro každý algoritmus jeden graf stavového prostoru. **V každém uzlu grafu uveďte, které domény proměnných se v důsledku propagací změnilo a která omezení tyto změny způsobila.**

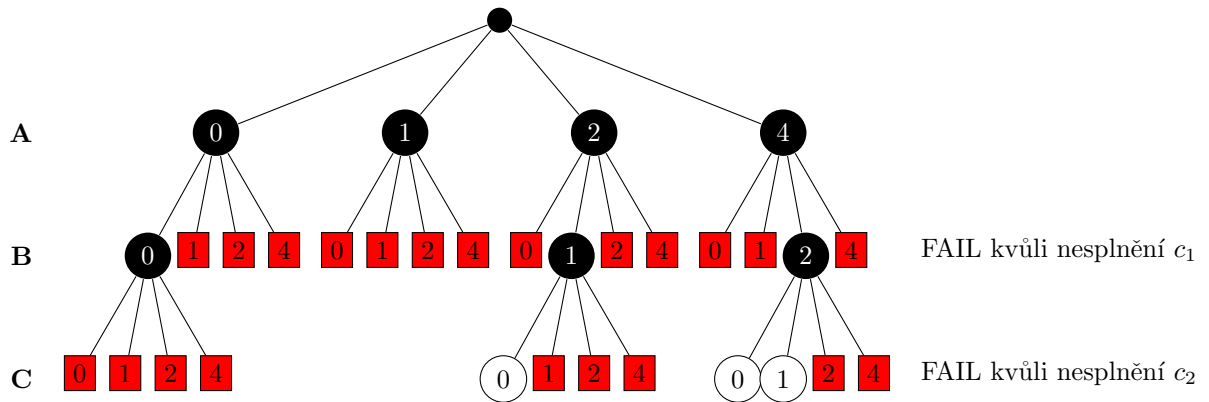
Příklady 2. – řešení

Notace Zachována notace z přednášky.

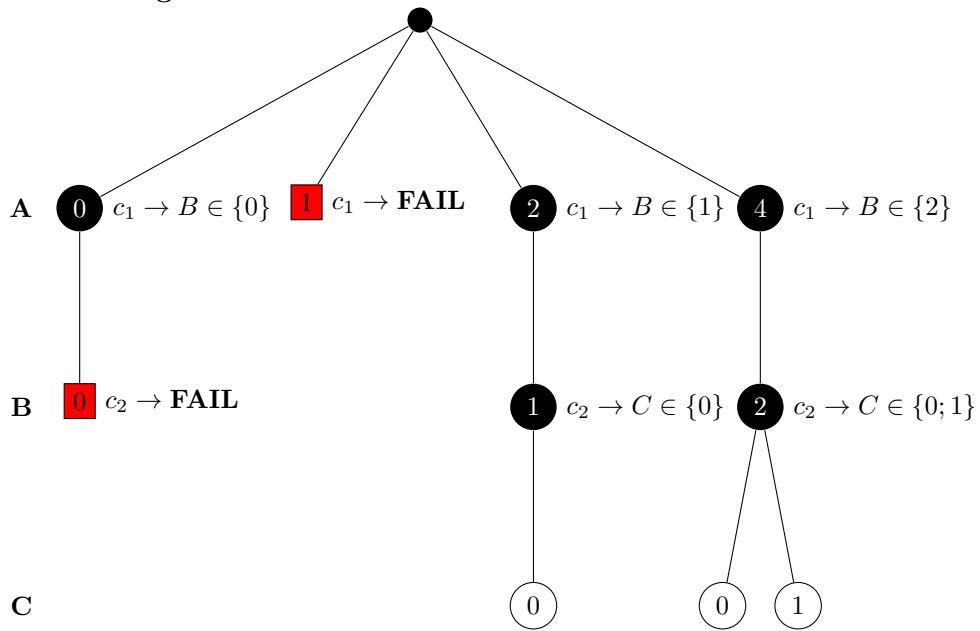
- Vnitřní stav: ●
- Splnitelné přiřazení: ○
- Fail stav: ■

U algoritmů *forward checking* a *look ahead* je u každého uzlu, kde je potřeba, zobrazeno omezení a změny, které byly propagovány.

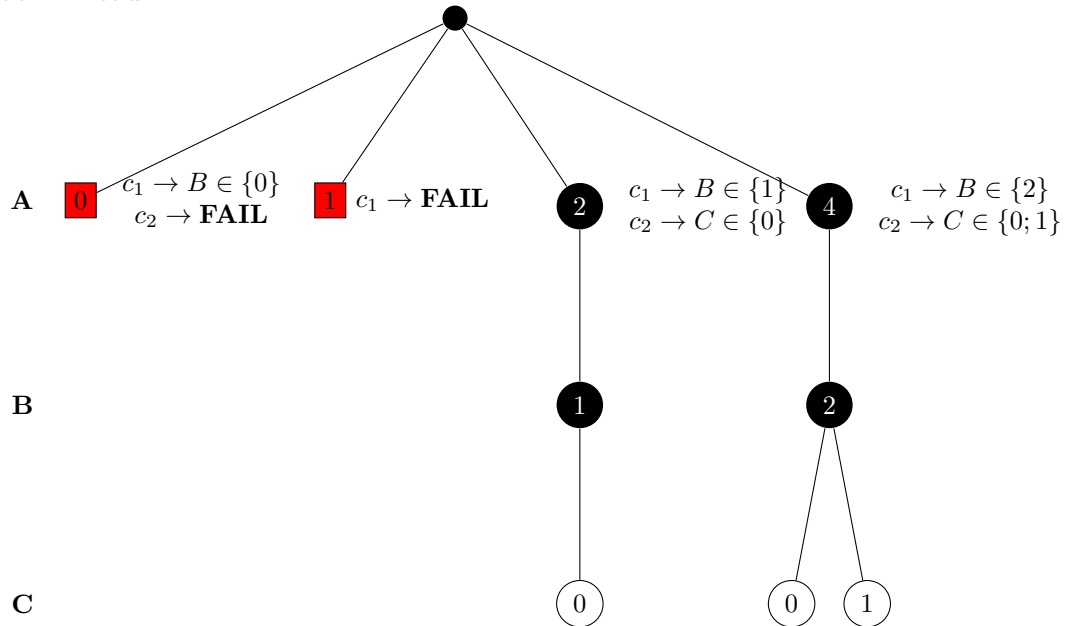
Backtracking



Forward Checking



Look Ahead



Příklady 3. – zadání

Máme zadán systém omezení $C = c1, c2, c3, c4, c5$ nad polookruhem $\langle \mathbb{N} \cup \{+\infty\}, \min, +, +\infty, 0 \rangle$

$c1: Time1 \in \{1..3\} @ Time1$	(preference odpovídá hodnotě)
$c2: Time2 \in \{1..3\} @ 2 * Time2$	(preference odpovídá dvojnásobku hodnoty)
$c3: Time3 \in \{1..3\} @ 3 * Time3$	(preference odpovídá trojnásobku hodnoty)
$c4: Time1 < Time2 @ (0, 5)$	(pokud omezení platí, ak je reference 0 pokud neplatí, pak je preference 5)
$c5: Time3 < Time1 @ (0, 5)$	(jako $c4$)

1. O jakou instanci omezení nad polookruhy se jedná?
2. Ukažte, jakým způsobem se spočítá úroveň konzistence tohoto problému.
3. Ukažte, jak se spočítá $\bigoplus C \downarrow_{\{Time1, Time2, Time3\}}$
4. Ukažte, jak se spočítá $\bigoplus C \downarrow_{\{Time1, Time2\}}$
5. Ukažte, jak se spočítá $\bigoplus C \downarrow_{\{Time1\}}$

Ve všech případech (body 2-5) uveďte postup řešení

Příklady 3. – řešení

1. Jedná se o CSP s váhami.

2. Úroveň konzistence problému odpovídá $\bigoplus C \downarrow_{\emptyset} =$

$$\begin{aligned} & \min(\mu_{\bigoplus C}(1, 1, 1), \mu_{\bigoplus C}(1, 1, 2), \mu_{\bigoplus C}(1, 1, 3), \mu_{\bigoplus C}(1, 2, 1), \\ & \mu_{\bigoplus C}(1, 2, 2), \mu_{\bigoplus C}(1, 2, 3), \mu_{\bigoplus C}(1, 3, 1), \mu_{\bigoplus C}(1, 3, 2), \mu_{\bigoplus C}(1, 3, 3), \\ & \mu_{\bigoplus C}(2, 1, 1), \mu_{\bigoplus C}(2, 1, 2), \mu_{\bigoplus C}(2, 1, 3), \mu_{\bigoplus C}(2, 2, 1), \\ & \mu_{\bigoplus C}(2, 2, 2), \mu_{\bigoplus C}(2, 2, 3), \mu_{\bigoplus C}(2, 3, 1), \mu_{\bigoplus C}(2, 3, 2), \mu_{\bigoplus C}(2, 3, 3), \\ & \mu_{\bigoplus C}(3, 1, 1), \mu_{\bigoplus C}(3, 1, 2), \mu_{\bigoplus C}(3, 1, 3), \mu_{\bigoplus C}(3, 2, 1), \\ & \mu_{\bigoplus C}(3, 2, 2), \mu_{\bigoplus C}(3, 2, 3), \mu_{\bigoplus C}(3, 3, 1), \mu_{\bigoplus C}(3, 3, 2), \mu_{\bigoplus C}(3, 3, 3)) = \\ & \min(16, 19, 22, 13, 16, 19, 15, 18, 21, 12, 20, 23, 14, 22, 25, 11, 19, 22, 13, 16, 24, 15, 18, 26, 17, 20, 28) \\ & = 11 \end{aligned}$$

3. $\bigoplus C \downarrow_{\{Time1, Time2, Time3\}}(1, 1, 1) = \min(\mu_{\bigoplus C}(1, 1, 1)) = \min(16) = 16$
 $\bigoplus C \downarrow_{\{Time1, Time2, Time3\}}(1, 1, 2) = \min(\mu_{\bigoplus C}(1, 1, 2)) = \min(19) = 19$
 $\bigoplus C \downarrow_{\{Time1, Time2, Time3\}}(1, 1, 3) = \min(\mu_{\bigoplus C}(1, 1, 3)) = \min(22) = 22$
 $\bigoplus C \downarrow_{\{Time1, Time2, Time3\}}(1, 2, 1) = \min(\mu_{\bigoplus C}(1, 2, 1)) = \min(13) = 13$
 $\bigoplus C \downarrow_{\{Time1, Time2, Time3\}}(1, 2, 2) = \min(\mu_{\bigoplus C}(1, 2, 2)) = \min(16) = 16$
 $\bigoplus C \downarrow_{\{Time1, Time2, Time3\}}(1, 2, 3) = \min(\mu_{\bigoplus C}(1, 2, 3)) = \min(19) = 19$
 $\bigoplus C \downarrow_{\{Time1, Time2, Time3\}}(1, 3, 1) = \min(\mu_{\bigoplus C}(1, 3, 1)) = \min(15) = 15$
 $\bigoplus C \downarrow_{\{Time1, Time2, Time3\}}(1, 3, 2) = \min(\mu_{\bigoplus C}(1, 3, 2)) = \min(18) = 18$
 $\bigoplus C \downarrow_{\{Time1, Time2, Time3\}}(1, 3, 3) = \min(\mu_{\bigoplus C}(1, 3, 3)) = \min(21) = 21$
 $\bigoplus C \downarrow_{\{Time1, Time2, Time3\}}(2, 1, 1) = \min(\mu_{\bigoplus C}(2, 1, 1)) = \min(12) = 12$
 $\bigoplus C \downarrow_{\{Time1, Time2, Time3\}}(2, 1, 2) = \min(\mu_{\bigoplus C}(2, 1, 2)) = \min(20) = 20$
 $\bigoplus C \downarrow_{\{Time1, Time2, Time3\}}(2, 1, 3) = \min(\mu_{\bigoplus C}(2, 1, 3)) = \min(23) = 23$
 $\bigoplus C \downarrow_{\{Time1, Time2, Time3\}}(2, 2, 1) = \min(\mu_{\bigoplus C}(2, 2, 1)) = \min(14) = 14$
 $\bigoplus C \downarrow_{\{Time1, Time2, Time3\}}(2, 2, 2) = \min(\mu_{\bigoplus C}(2, 2, 2)) = \min(22) = 22$
 $\bigoplus C \downarrow_{\{Time1, Time2, Time3\}}(2, 2, 3) = \min(\mu_{\bigoplus C}(2, 2, 3)) = \min(25) = 25$
 $\bigoplus C \downarrow_{\{Time1, Time2, Time3\}}(2, 3, 1) = \min(\mu_{\bigoplus C}(2, 3, 1)) = \min(11) = 11$
 $\bigoplus C \downarrow_{\{Time1, Time2, Time3\}}(2, 3, 2) = \min(\mu_{\bigoplus C}(2, 3, 2)) = \min(19) = 19$
 $\bigoplus C \downarrow_{\{Time1, Time2, Time3\}}(2, 3, 3) = \min(\mu_{\bigoplus C}(2, 3, 3)) = \min(22) = 22$
 $\bigoplus C \downarrow_{\{Time1, Time2, Time3\}}(3, 1, 1) = \min(\mu_{\bigoplus C}(3, 1, 1)) = \min(13) = 13$
 $\bigoplus C \downarrow_{\{Time1, Time2, Time3\}}(3, 1, 2) = \min(\mu_{\bigoplus C}(3, 1, 2)) = \min(16) = 16$
 $\bigoplus C \downarrow_{\{Time1, Time2, Time3\}}(3, 1, 3) = \min(\mu_{\bigoplus C}(3, 1, 3)) = \min(24) = 24$
 $\bigoplus C \downarrow_{\{Time1, Time2, Time3\}}(3, 2, 1) = \min(\mu_{\bigoplus C}(3, 2, 1)) = \min(15) = 15$
 $\bigoplus C \downarrow_{\{Time1, Time2, Time3\}}(3, 2, 2) = \min(\mu_{\bigoplus C}(3, 2, 2)) = \min(18) = 18$
 $\bigoplus C \downarrow_{\{Time1, Time2, Time3\}}(3, 2, 3) = \min(\mu_{\bigoplus C}(3, 2, 3)) = \min(26) = 26$
 $\bigoplus C \downarrow_{\{Time1, Time2, Time3\}}(3, 3, 1) = \min(\mu_{\bigoplus C}(3, 3, 1)) = \min(17) = 17$
 $\bigoplus C \downarrow_{\{Time1, Time2, Time3\}}(3, 3, 2) = \min(\mu_{\bigoplus C}(3, 3, 2)) = \min(20) = 20$
 $\bigoplus C \downarrow_{\{Time1, Time2, Time3\}}(3, 3, 3) = \min(\mu_{\bigoplus C}(3, 3, 3)) = \min(28) = 28$

4. $\bigoplus^C \Downarrow_{\{Time1, Time2\}} (1, 1) = \min(\mu_{\bigoplus^C}(1, 1, 1), \mu_{\bigoplus^C}(1, 1, 2), \mu_{\bigoplus^C}(1, 1, 3)) =$
 $= \min(16, 19, 22) = 16$
 $\bigoplus^C \Downarrow_{\{Time1, Time2\}} (1, 2) = \min(\mu_{\bigoplus^C}(1, 2, 1), \mu_{\bigoplus^C}(1, 2, 2), \mu_{\bigoplus^C}(1, 2, 3)) =$
 $= \min(13, 16, 19) = 13$
 $\bigoplus^C \Downarrow_{\{Time1, Time2\}} (1, 3) = \min(\mu_{\bigoplus^C}(1, 3, 1), \mu_{\bigoplus^C}(1, 3, 2), \mu_{\bigoplus^C}(1, 3, 3)) =$
 $= \min(15, 18, 21) = 15$
 $\bigoplus^C \Downarrow_{\{Time1, Time2\}} (2, 1) = \min(\mu_{\bigoplus^C}(2, 1, 1), \mu_{\bigoplus^C}(2, 1, 2), \mu_{\bigoplus^C}(2, 1, 3)) =$
 $= \min(12, 20, 23) = 12$
 $\bigoplus^C \Downarrow_{\{Time1, Time2\}} (2, 2) = \min(\mu_{\bigoplus^C}(2, 2, 1), \mu_{\bigoplus^C}(2, 2, 2), \mu_{\bigoplus^C}(2, 2, 3)) =$
 $= \min(14, 22, 25) = 14$
 $\bigoplus^C \Downarrow_{\{Time1, Time2\}} (2, 3) = \min(\mu_{\bigoplus^C}(2, 3, 1), \mu_{\bigoplus^C}(2, 3, 2), \mu_{\bigoplus^C}(2, 3, 3)) =$
 $= \min(11, 19, 22) = 11$
 $\bigoplus^C \Downarrow_{\{Time1, Time2\}} (3, 1) = \min(\mu_{\bigoplus^C}(3, 1, 1), \mu_{\bigoplus^C}(3, 1, 2), \mu_{\bigoplus^C}(3, 1, 3)) =$
 $= \min(13, 16, 24) = 13$
 $\bigoplus^C \Downarrow_{\{Time1, Time2\}} (3, 2) = \min(\mu_{\bigoplus^C}(3, 2, 1), \mu_{\bigoplus^C}(3, 2, 2), \mu_{\bigoplus^C}(3, 2, 3)) =$
 $= \min(15, 18, 26) = 15$
 $\bigoplus^C \Downarrow_{\{Time1, Time2\}} (3, 3) = \min(\mu_{\bigoplus^C}(3, 3, 1), \mu_{\bigoplus^C}(3, 3, 2), \mu_{\bigoplus^C}(3, 3, 3)) =$
 $= \min(17, 20, 28) = 17$
5. $\bigoplus^C \Downarrow_{\{Time1\}} (1) = \min(\mu_{\bigoplus^C}(1, 1, 1), \mu_{\bigoplus^C}(1, 1, 2), \mu_{\bigoplus^C}(1, 1, 3), \mu_{\bigoplus^C}(1, 2, 1),$
 $\mu_{\bigoplus^C}(1, 2, 2), \mu_{\bigoplus^C}(1, 2, 3), \mu_{\bigoplus^C}(1, 3, 1), \mu_{\bigoplus^C}(1, 3, 2), \mu_{\bigoplus^C}(1, 3, 3)) =$
 $= \min(16, 19, 22, 13, 16, 19, 15, 18, 21) = 13$
 $\bigoplus^C \Downarrow_{\{Time1\}} (2) = \min(\mu_{\bigoplus^C}(2, 1, 1), \mu_{\bigoplus^C}(2, 1, 2), \mu_{\bigoplus^C}(2, 1, 3), \mu_{\bigoplus^C}(2, 2, 1),$
 $\mu_{\bigoplus^C}(2, 2, 2), \mu_{\bigoplus^C}(2, 2, 3), \mu_{\bigoplus^C}(2, 3, 1), \mu_{\bigoplus^C}(2, 3, 2), \mu_{\bigoplus^C}(2, 3, 3)) =$
 $= \min(12, 20, 23, 14, 22, 25, 11, 19, 22) = 11$
 $\bigoplus^C \Downarrow_{\{Time1\}} (3) = \min(\mu_{\bigoplus^C}(3, 1, 1), \mu_{\bigoplus^C}(3, 1, 2), \mu_{\bigoplus^C}(3, 1, 3), \mu_{\bigoplus^C}(3, 2, 1),$
 $\mu_{\bigoplus^C}(3, 2, 2), \mu_{\bigoplus^C}(3, 2, 3), \mu_{\bigoplus^C}(3, 3, 1), \mu_{\bigoplus^C}(3, 3, 2), \mu_{\bigoplus^C}(3, 3, 3)) =$
 $= \min(13, 16, 24, 15, 18, 26, 17, 20, 28) = 13$

Příklady 4. – zadání

1. Uveďte plné i zkrácené názvy všech typů konzistence a konzistenčních algoritmů, které znáte. Uveďte jednotlivé pojmy a algoritmy *systematicky* se začleněním do skupin dle jejich typu. (Popisy algoritmů, typů konzistencí ani skupin neuvádějte, toto není součástí řešení, stačí uvádět pouze názvy se zkratkami.)
2. Ukažte podrobně, jak je aplikován algoritmus AC-4 pro zajištění hranové konzistence v následujícím příkladu (včetně *postupných* změn datových struktur).

$$A, B, C \in \{0, 1, 2, 3\}, \quad A < B, \quad C = B + 1$$

3. Aplikujte algoritmus DAC pro zajištění směrové hranové konzistence uvedeného problému při uspořádání proměnných ($X1, X2, X3, X4$). Ukažte také, jak je možné tento problém řešit pomocí algoritmu AC-3 pro zajištění hranové konzistence. Popište jednotlivé kroky řešení problému, aby bylo vidět, jak se *postupně* mění domény proměnných a *která* omezení tyto změny způsobila.

$$X1 \in \{1, 2, 4, 5\}, \quad X2 \in \{1, 2, 4, 5\}, \quad X3 \in \{1, 3, 4, 5\}, \quad X4 \in \{1, 2, 3, 5\}$$
$$X4 = X2, \quad X3 = X1, \quad X2 = X1$$

Příklady 4. – řešení

1.příklad

Unární podmínky

- vrcholová konzistence (NC, Node Consistency)
algoritmus: [unární podmínky převedeme do domén proměnných]

Binární podmínky

- hranová konzistence (AC, Arc Consistency)
algoritmy: AC-1, AC-3, AC-4, AC-3.1, AC-2001, [AC-5, AC-6, AC-7, ...]
- konzistence po cestě (PC, Path Consistency)
algoritmy: PC-1, PC-2, PC-4
- omezená konzistence po cestě (RPC, Restricted Path Consistency)
na pomezí mezi hranovou konzistencí a konzistencí po cestě
algoritmus: [založen na AC-4 + seznam cest pro PC]
- směrová hranová konzistence (DAC, Directional Arc Consistency)
využívá hranovou konzistenci hrany
algoritmus: DAC

Nebinární podmínky

- k-konzistence
algoritmy: [podobný princip jako AC-1 a PC-1]
- silná k-konzistence
- obecná hranová konzistence (doménová konzistence, GAC, General Arc Consistency)
- konzistence mezi (BC, Bounds Consistency)
- **směrové konzistence**
 - [viz směrová hranová konzistence pro binární podmínky]
 - směrová *i*-konzistence
 - silná směrová *i*-konzistence (DIC-*i*)
 - adaptivní konzistence (ADC)
algoritmus: ADC
- **použití různých typů konzistence**
 - algoritmy: konzistenční algoritmus pro nebinární podmínky s frontou proměnných,
konzistenční algoritmus s událostmi

2.příklad

Nejprve je třeba spustit algoritmus pro inicializaci podpor, který inicializuje datové struktury Q , S a $counter$. Na začátku jsou Q a množiny všech podpor S nastaveny na \emptyset a všechny $counter$ mají nulovou hodnotu. Iterace vnějšího cyklu probíhá pro každou hranu.

Hrana (V_A, V_B) :

$$S_{B,1} = \{(A, 0)\}$$

$$S_{B,2} = \{(A, 0), (A, 1)\}$$

$$S_{B,3} = \{(A, 0), (A, 1), (A, 2)\}$$

$$counter[(A, B), 0] = 3$$

$$counter[(A, B), 1] = 2$$

$$counter[(A, B), 2] = 1$$

$$counter[(A, B), 3] = 0 \Rightarrow \text{z } D_A \text{ odstraníme hodnotu 3 a do } Q \text{ přidáme } (A, 3)$$

Celkem: $A \in \{0, 1, 2\}, B \in \{0, 1, 2, 3\}, C \in \{0, 1, 2, 3\} \quad Q = \{(A, 3)\}$

Hrana (V_B, V_A) :

$$S_{A,0} = \{(B, 1), (B, 2), (B, 3)\}$$

$$S_{A,1} = \{(B, 2), (B, 3)\}$$

$$S_{A,2} = \{(B, 3)\}$$

$$counter[(B, A), 0] = 0 \Rightarrow \text{z } D_B \text{ odstraníme hodnotu 0 a do } Q \text{ přidáme } (B, 0)$$

$$counter[(B, A), 1] = 1$$

$$counter[(B, A), 2] = 2$$

$$counter[(B, A), 3] = 3$$

Celkem: $A \in \{0, 1, 2\}, B \in \{1, 2, 3\}, C \in \{0, 1, 2, 3\} \quad Q = \{(A, 3), (B, 0)\}$

Hrana (V_B, V_C) :

komentář: $S_{C,1}$ zůstane prázdná, protože 0 už není v doméně B

$$S_{C,2} = \{(B, 1)\}$$

$$S_{C,3} = \{(B, 2)\}$$

komentář: $counter[(B, C), 0]$ se nemění (je stále 0), protože 0 už není v doméně B

$$counter[(B, C), 1] = 1$$

$$counter[(B, C), 2] = 1$$

$$counter[(B, C), 3] = 0 \Rightarrow \text{z } D_B \text{ odstraníme hodnotu 3 a do } Q \text{ přidáme } (B, 3)$$

Celkem: $A \in \{0, 1, 2\}, B \in \{1, 2\}, C \in \{0, 1, 2, 3\} \quad Q = \{(A, 3), (B, 0), (B, 3)\}$

Hrana (V_C, V_B) :

$$S_{B,1} = S_{B,1} \cup \{(C, 2)\} = \{(A, 0), (C, 2)\}$$

$$S_{B,2} = S_{B,2} \cup \{(C, 3)\} = \{(A, 0), (A, 1), (C, 3)\}$$

$$counter[(C, B), 0] = 0 \Rightarrow \text{z } D_C \text{ odstraníme hodnotu 0 a do } Q \text{ přidáme } (C, 0)$$

$$counter[(C, B), 1] = 0 \text{ (komentář: v } D_B \text{ už není hodnota 0)} \Rightarrow \text{z } D_C \text{ odstraníme}$$

hodnotu 1 a do Q přidáme $(C, 1)$

$$counter[(C, B), 2] = 1$$

$$counter[(C, B), 3] = 1$$

Celkem: $A \in \{0, 1, 2\}, B \in \{1, 2\}, C \in \{2, 3\} \quad Q = \{(A, 3), (B, 0), (B, 3), (C, 0), (C, 1)\}$

Nyní je inicializace ukončena a pokračuje se v hlavním cyklu postupným výběrem prvků z Q , abychom zareagovali na odstranění jednotlivých hodnot proměnných v Q .

Vybereme a smažeme $(A, 3)$:

$$Q = \{(B, 0), (B, 3), (C, 0), (C, 1)\}$$

Nic se nemění, $(A, 3)$ nepodporoval žádný prvek ($S_{A,3} = \emptyset$).

Vybereme a smažeme $(B, 0)$:

$$Q = \{(B, 3), (C, 0), (C, 1)\}$$

Nic se nemění, $(B, 0)$ nepodporoval žádný prvek ($S_{B,0} = \emptyset$).

Vybereme a smažeme $(B, 3)$:

$$\text{counter}[(A, B), 0] = 2$$

$$\text{counter}[(A, B), 1] = 1$$

$$\text{counter}[(A, B), 2] = 0 \Rightarrow \text{z } D_A \text{ odstraníme hodnotu 2 a do } Q \text{ přidáme } (A, 2)$$

$$A \in \{0, 1\}, B \in \{1, 2\}, C \in \{2, 3\}$$

$$Q = \{(C, 0), (C, 1), (A, 2)\}$$

Vybereme a smažeme $(C, 0)$:

$$Q = \{(C, 1), (A, 2)\}$$

Nic se nemění, $(C, 0)$ nepodporoval žádný prvek ($S_{C,0} = \emptyset$).

Vybereme a smažeme $(C, 1)$:

$$Q = \{(A, 2)\}$$

Nic se nemění, $(C, 1)$ nepodporoval žádný prvek ($S_{C,1} = \emptyset$).

Vybereme a smažeme $(A, 2)$:

$$Q = \emptyset$$

$(A, 2)$ je podporou pro hodnotu, která byla odstraněna ($S_{A,2} = \{(B, 3)\}$), snížíme

$\text{counter}[(B, A), 3]$, do Q se nic nepřidává

Výsledek: $A \in \{0, 1\}, B \in \{1, 2\}, C \in \{2, 3\}$.

3.příklad

Řešení pro algoritmus DAC

Hrany grafu jsou $(X1, X2)$, $(X2, X1)$, $(X1, X3)$, $(X3, X1)$, $(X2, X4)$, $(X4, X2)$, na běh algoritmu mají vliv pouze hrany $(X1, X2)$, $(X1, X3)$, $(X2, X4)$.

Algoritmus prochází proměnné v opačném pořadí než je jejich uspořádání:

Procházíme proměnnou $X4$:

Jediná zasažená hrana je $(X2, X4)$, z domény $X2$ odstraníme hodnotu 4, $X2 \in \{1, 2, 5\}$

Procházíme proměnnou $X3$:

Jediná zasažená hrana je $(X1, X3)$, z domény $X1$ odstraníme hodnotu 2, $X1 \in \{1, 4, 5\}$

Procházíme proměnnou $X2$:

Jediná zasažená hrana je $(X1, X2)$, z domény $X1$ odstraníme hodnotu 4, $X1 \in \{1, 5\}$

Procházíme proměnnou $X1$:

Není zasažena žádná hrana, nic se nemění.

Výsledek DAC: $X1 \in \{1, 5\}$, $X2 \in \{1, 2, 5\}$, $X3 \in \{1, 3, 4, 5\}$, $X4 \in \{1, 2, 3, 5\}$.

Řešení pro algoritmus AC

Na začátku inicializujeme frontu:

$Q = \{(X4, X2), (X2, X4), (X3, X1), (X1, X3), (X2, X1), (X1, X2)\}$. Hlava fronty je vlevo.

Vybereme a smažeme $(X4, X2)$

$X4 \in \{1, 2, 5\}$

$Q = \{(X2, X4), (X3, X1), (X1, X3), (X2, X1), (X1, X2)\}$

Vybereme a smažeme $(X2, X4)$

$X2 \in \{1, 2, 5\}$

$Q = \{(X3, X1), (X1, X3), (X2, X1), (X1, X2)\}$

Vybereme a smažeme $(X3, X1)$

$X3 \in \{1, 4, 5\}$

$Q = \{(X1, X3), (X2, X1), (X1, X2)\}$

Vybereme a smažeme $(X1, X3)$

$X1 \in \{1, 4, 5\}$

$Q = \{(X2, X1), (X1, X2)\}$

Vybereme a smažeme $(X2, X1)$

$X2 \in \{1, 5\}$, do fronty přidáme $(X4, X2)$

$Q = \{(X1, X2), (X4, X2)\}$

Vybereme a smažeme $(X1, X2)$

$X1 \in \{1, 5\}$, do fronty přidáme $(X3, X1)$

$Q = \{(X4, X2), (X3, X1)\}$

Vybereme a smažeme $(X4, X2)$

$X4 \in \{1, 5\}$

$Q = \{(X3, X1)\}$

Vybereme a smažeme $(X3, X1)$

$X3 \in \{1, 5\}$

$Q = \emptyset$

Výsledek AC: $X1 \in \{1, 5\}$, $X2 \in \{1, 5\}$, $X3 \in \{1, 5\}$, $X4 \in \{1, 5\}$.

Příklady 5. – zadání

1. Ukažte, jak vypadá *graf stavového prostoru* pro řešení problému

- proměnné: $A \in \{2, 3\}$, $B \in \{1, 2, 3\}$, $C \in \{1, 2, 3\}$, $D \in \{1, 2\}$
- omezení: $c_1 : A \neq B$, $c_2 : B = C$, $c_3 : B \neq D$, $c_4 : C \neq D$

algoritmem

- (a) *backtrackingu*
- (b) *kontroly dopředu (forward checking)*

při uspořádání proměnných A, B, C a při uspořádání hodnot od nejmenší k největší. Ve vašem řešení uveďte pro každý algoritmus jeden graf stavového prostoru. V každém uzlu grafu uveďte, které domény proměnných se v důsledku propagací změnily a která omezení tyto změny způsobila.

2. Nalezněte *první řešení* problému

- proměnné: $X1 \in \{a, b, c\}$, $X2 \in \{b, c\}$, $X3 \in \{a, b\}$, $X4 \in \{c, d\}$
- omezení: $c_1 : X1 = X2$, $c_2 : X1 \neq X3$, $c_3 : X1 = X4$

algoritmem *Gaschnigova skoku zpět* při uspořádání proměnných $X1, X2, X3, X4$ a při uspořádání hodnot a, b, c, d . Jako součást řešení popište postupné změny hodnot proměnných, hodnot $latest_i$, hodnoty i (hlavní cyklus algoritmu) a hodnoty k (při výběru hodnoty) a dále uveďte, které omezení bylo kontrolováno v daném kroku (včetně uvedení úspěchu resp. neúspěchu), např. postupným zaznamenáváním změn hodnot a uvedením kontrolovaných omezení do tabulky se sloupci

i , $X1, latest_1$, $X2, latest_2$, $X3, latest_3$, $X4, latest_4$, k , *omezení, úspěch/neúspěch*

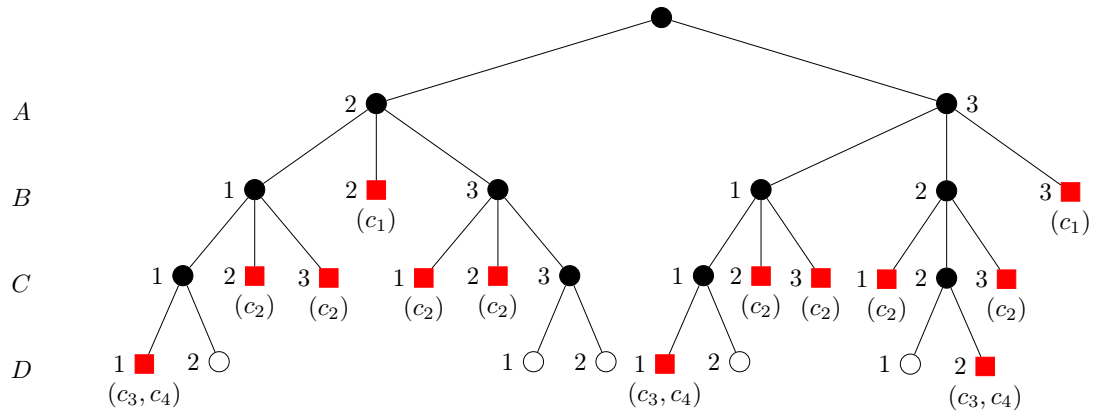
nebo postupným uváděním těchto informací (ve strukturovaném textu).

Příklady 5. – řešení

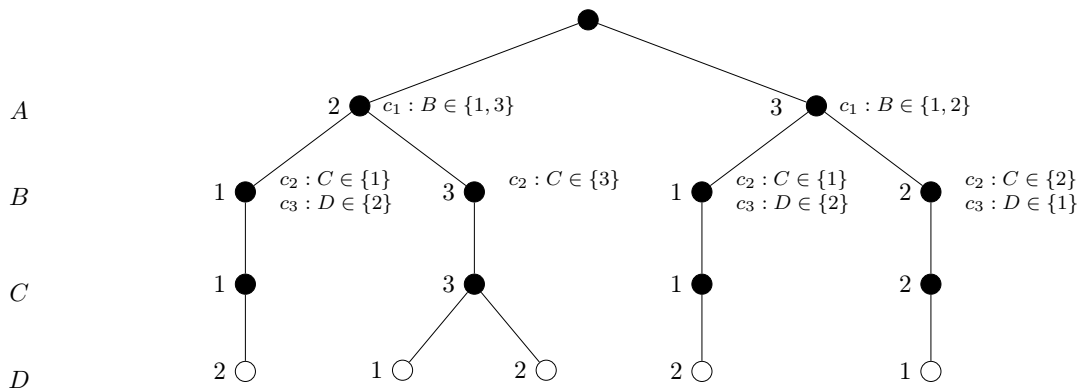
1.příklad

Používané značení: ● vnitřní uzel, ○ řešení, ■ chybné přiřazení.

Backtracking



Kontrola dopředu



2.příklad

Slovní popis:

Z D'_1 se vybere: hodnota a

Z D'_2 se vybere: hodnota b – není konzistentní s $X1$ (omezení c_1)

hodnota c – není konzistentní s $X1$ (omezení c_1)

Skok zpět k $X1$: z D'_1 se vybere hodnota b

Z D'_2 se vybere: hodnota b – je konzistentní s $X1$ (omezení c_1)

Z D'_3 se vybere: hodnota a – je konzistentní s $X1$ (omezení c_2)

Z D'_4 se vybere: hodnota c – není konzistentní s $X1$ (omezení c_3)

hodnota d – není konzistentní s $X1$ (omezení c_3)

Skok zpět k $X1$: z D'_1 se vybere hodnota c

Z D'_2 se vybere: hodnota b – není konzistentní s $X1$ (omezení c_1)

hodnota c – je konzistentní s $X1$ (omezení c_1)

Z D'_3 se vybere: hodnota a – je konzistentní s $X2$ (omezení c_2)

Z D'_4 se vybere: hodnota c – je konzistentní s $X1$ (omezení c_3)

První řešení: $X1 = c$, $X2 = c$, $X3 = a$, $X4 = c$.

V jednom kroku je nastavena hodnota proměnné, nastaven čítač k , realizován konzistenční test a aktualizovaná hodnota $latest_i$ (prvotní inicializace $latest_i$ na 0 není uvedena v samostatném kroku).

V tabulce uvedena vybraná hodnota z D'_i jako X_i , nicméně ke skutečnému přiřazení by došlo až po ukončení procedury výběru hodnoty.

i	$X1, latest_1$	$X2, latest_2$	$X3, latest_3$	$X4, latest_4$	k	omezení, úspěch/neúspěch
1	$a, 0$	–	–	–	1	–
2	$a, 0$	$b, 1$	–	–	1	c_1 , neúspěch
2	$a, 0$	$c, 1$	–	–	1	c_1 , neúspěch
1	$b, 0$	–	–	–	1	–
2	$b, 0$	$b, 1$	–	–	1	c_1 , úspěch
3	$b, 0$	$b, 1$	$a, 1$	–	1	c_2 , úspěch
3	$b, 0$	$b, 1$	$a, 2$	–	2	–
4	$b, 0$	$b, 1$	$a, 2$	$c, 1$	1	c_3 , neúspěch
4	$b, 0$	$b, 2$	$a, 2$	$d, 1$	1	c_3 , neúspěch
1	$c, 0$	–	–	–	1	–
2	$c, 0$	$b, 1$	–	–	1	c_1 , neúspěch
2	$c, 0$	$c, 1$	–	–	1	c_1 , úspěch
3	$c, 0$	$c, 1$	$a, 1$	–	1	c_2 , úspěch
3	$c, 0$	$c, 1$	$a, 2$	–	2	–
4	$c, 0$	$c, 1$	$a, 2$	$c, 1$	1	c_3 , úspěch
4	$c, 0$	$c, 1$	$a, 2$	$c, 2$	2	–
4	$c, 0$	$c, 1$	$a, 2$	$c, 3$	3	–

Příklady 6. – zadání

1. Uveďte plné i zkrácené názvy všech prohledávacích algoritmů, které znáte. Uveďte jednotlivé algoritmy *systematicky* se začleněním do skupin dle jejich typu.
(POZOR: Popisy algoritmů, skupin ani další pojmy neuvádějte! Toto není součástí řešení, je nutné uvádět pouze názvy a případné zkratky.)

2. Napište řešení následujícího příkladu v OPL. Kód řádně okomentujte.

Určete čas a místnost pro výuku množiny předmětů. Problém řešte obecně, přičemž si vytvořte jednu datovou instanci, která bude mít následující hodnoty parametrů:

- (a) rozvrh tvoříte pro 3 vyučovací dny a každý den má 10 hodin
- (b) máte zadáno 14 předmětů
- (c) délka výuky předmětu je 4 nebo 5 hodin
- (d) máte k dispozici 3 místnosti
- (e) máte zadány 3 třídy (skupiny žáků)
- (f) každá třída má v zadání určeno 4 až 5 předmětů, které bude navštěvovat

Do řešení zahrňte tyto omezující podmínky:

- (g) výuka předmětu musí probíhat souvisle bez přerušení (tj. nesmí např. začínat jeden den večer a končit druhý den ráno)
- (h) v každé místnosti je nejvýše jeden předmět v danou dobu
- (i) pro každou třídu je zadána množina jejích předmětů; každá třída může mít vždy nejvýše jeden z těchto předmětů v danou dobu

A aplikujte následující účelovou funkci:

- (j) Jednotlivé předměty by měly být do rozvrhu umístěny tak, aby výuka všech tříd skončila co nejdříve (např. třetí den dopoledne). Minimalizujte tedy nejpozdější koncový čas výuky posledních předmětů všech tříd.

Příklady 6. – řešení

1.příklad

- Generuj a testuj – Generate and Test
- **Stromové prohledávání**
 - Backtracking (BT)
 - **Pohled dopředu – Look–Ahead (LA)**
 - * Kontrola dopředu – Forward Checking (FC)
 - * Částečný pohled dopředu – Partial Look–Ahead (PLA)
 - * Úplný pohled dopředu – Full Look–Ahead (FLA)
 - * Opravdový úplný pohled dopředu – Real Full Look–Ahead (RFLA)
 - **Pohled zpět – Look–Back**
 - * Gaschnigův skok zpět – Gashnig’s Backjumping (GBJ)
 - * Konflikty řízený skok zpět – Conflict–Directed Backjumping (CBJ)
 - * Učení skoku zpět – Jumpback Learning
 - * Dynamický backtracking (DB)
 - * Backmarking
 - **Neúplná stromová prohledávání**
 - * Randomizovaný backtracking – Random Search (RS)
 - * Randomizovaný backtracking s učením
 - * Omezení počtu návratů – Bounded–Backtrack Search (BBS)
 - * Omezení hloubky – Depth–Bounded Search (DBS)
 - * Prohledávání s kreditem – Credit Search (CS)
 - * Iterativní rozšiřování – Iterative Broadening (IB)
 - * **Prohledávání s diskrepancemi**
 - Omezené diskrepance – Limited Discrepancy Search (LDS)
 - Vylepšené omezené diskrepance – Improved Limited Discrepancy Search (ILDS)
 - Hloubkou omezené diskrepance – Depth–Bounded Discrepancy Search (DDS)
- **Lokální prohledávání (LS)**
 - Největšího stoupání – Hill Climbing (HC)
 - Minimalizace konfliktů – Minimum Conflict (MC)
 - Náhodná procházka – Random Walk (RW)
 - Tabu prohledávání – Tabu Search (TS)
 - Simulované žhání – Simulated Annealing (SA)
 - Kombinace algoritmů (HC+MC, HC+TS, MC + RW, HC + RW, ...)
 - Greedy SAT (GSAT)
- Iterativní dopředné prohledávání – Iterative Forward Search (IFS)
- **Hybridní prohledávání**
 - Lokální prohledávání před stromovým prohledáváním
 - Stromové prohledávání před lokálním prohledáváním
 - Stromové prohledávání doplněné lokálním prohledáváním
 - Lokální prohledávání doplněné stromovým prohledáváním

2.příklad

```

/***** * .dat soubor *****/
Dny = 3;
Hodin = 10;
Predmetu = 14;
Trvani = [4,5,5,4,4,5,4,5,4,5,4,5,4,5];
Mistnosti = 3;
Tridy = 3;
PredmetyTrid = { <1,1>, <1,2>, <1,3>, <1,4>, <1,5>, <2,6>, <2,7>, <2,8>, <2,9>,
                 <3,10>, <3,11>, <3,12>, <3,13>, <3,14> };

/***** * .mod soubor *****/
using CP;

// přirazení všech hodnot z datového souboru
int Dny = ...;
int Hodin = ...;
int Predmetu = ...;
int Trvani[1..Predmetu] = ...;
int Mistnosti = ...;
int Tridy = ...;

// pomocné range
range rTridy = 1..Tridy;
range rPredmety = 1..Predmetu;
range rMistnosti = 1..Mistnosti;

// tuple zaznamenávající přiřazení předmětu ke třídám
tuple predmetyTrid{
    int trida;
    int predmet;
}

// naplnění tuple z datového souboru
{predmetyTrid} PredmetyTrid = ...;

// přiřazení času předmětům v maximálním rozmezí Dny*Hodin, přiřazení trvání
dvar interval casy[p in rPredmety] in 0 .. Dny * Hodin size Trvani[p];

// volitelný interval pro přiřazení předmětů a tříd do místnosti
dvar interval prirazeni[rMistnosti][PredmetyTrid] optional;

// sekvence rozvrhu tříd, z důvodu nepřekrývání
dvar sequence rozvrhTridy[t in rTridy] in all (m in rMistnosti, pt in
PredmetyTrid : pt.trida == t) prirazeni[m][pt];

// sekvence rozvrhu místnosti, z důvodu nepřekrývání
dvar sequence rozvrhMistnosti[m in rMistnosti] in all (pt in PredmetyTrid)
prirazeni[m][pt];

// minimalizace konce posledního předmětu přes všechny třídy
minimize max(p in rPredmety) endOf(casy[p]);

subject to{

// promazání domén, tak aby předmět nezačínal jeden den a končil druhý
forall(p in rPredmety)
    startOf(casy[p]) % Hodin <= (Hodin-Trvani[p]);

```

```
// zamezení výuky dvou předmětů pro jednu třídu zároveň
forall(t in rTridy)
    noOverlap(rozvrhTridy[t]);

// zamezení výuky dvou předmětů v jedné místnosti
forall(m in rMistnosti)
    noOverlap(rozvrhMistnosti[m]);

// pro každý předmět přiřazení jednoho volitelného intervalu (místnosti)
forall(pt in PredmetyTrid)
    alternative(casy[pt.predmet], all(m in rMistnosti)prirazeni[m][pt]);
}

// výpis pro kontrolu řešení
execute{
    for(var t=1; t <= Tridy; t++){
        writeln("Rozvrh "+t+" třídy:");
        for(var p in PredmetyTrid){
            if(p.trida==t){
                write("Předmět " + p.predmet + ": " + casy[p.predmet].start + "-" +
casy[p.predmet].end);
                for(var m=1; m<=Mistnosti; m++){
                    if(prirazeni[m][p].present)writeln(" v " + m + ". místnosti");
                }
            }
        }
        writeln();
    }
}
```

Příklady 7. – zadání

1. Uvedený CSP převed'te pomocí duálního problému na binární CSP. Máte zadány proměnné A, D, E, F s doménou $\{0, 1\}$, dále $B \in \{1, 2\}$, $C \in \{1, 2, 3\}$ a omezení

$$\begin{aligned} c_1 : & \quad A + B = C \\ c_2 : & \quad A + D + E = 1 \\ c_3 : & \quad A + E - F > 0 \\ c_4 : & \quad C + E > 2 \end{aligned}$$

2. Uveďte všechny podpory hodnot pro následující problém

- proměnné A, B, C mohou nabývat hodnot 3, 4, 5,
- $A \leq B$, $B > C$.

3. Ukažte, jak vypadají propagační pravidla konzistence mezi pro omezení $Y = X - C$ za předpokladu, že C je konstanta a X, Y jsou doménové proměnné. Ukažte průběh propagací (včetně aplikace Vámi navržených propagačních pravidel) při použití konzistence mezi v následujícím příkladu.

$$Y \in \{0, \dots, 10\} \quad X \in \{0, \dots, 5, 8, 9, 10\} \quad Y = X - 3$$

Nastal by v příkladu nějaký rozdíl, pokud bychom místo konzistence mezi použili hranovou konzistenci?

4. Uveďte jednotlivé kroky algoritmu DAC na následujícím příkladu při uspořádání proměnných D, C, B, A . Proměnné a omezení: $A, B, C, D \in \{1, 2, 3, 4\}$

$$\begin{aligned} c_1 : & \quad A + 1 = B \\ c_2 : & \quad C = A \\ c_4 : & \quad B + 1 = D \end{aligned}$$

Uveďte postup, jakým způsobem najdeme jednotlivá řešení bez navracení.

Příklady 7. – řešení

1.příklad

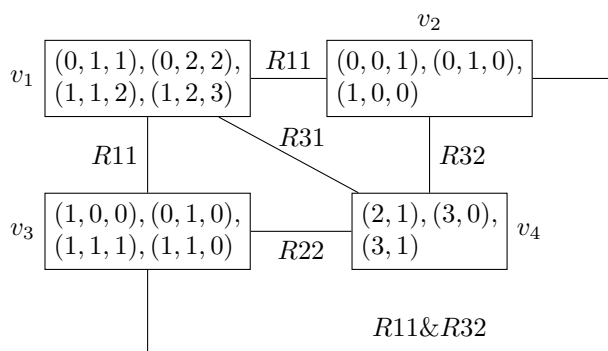
Podmíenku c_1 prevedieme na duálnu premennú v_1 s doménou $\{(0, 1, 1), (0, 2, 2), (1, 1, 2), (1, 2, 3)\}$.

Podmíenku c_2 prevedieme na duálnu premennú v_2 s doménou $\{(0, 0, 1), (0, 1, 0), (1, 0, 0)\}$.

Podmíenku c_3 prevedieme na duálnu premennú v_3 s doménou $\{(1, 0, 0), (0, 1, 0), (1, 1, 1), (1, 1, 0)\}$.

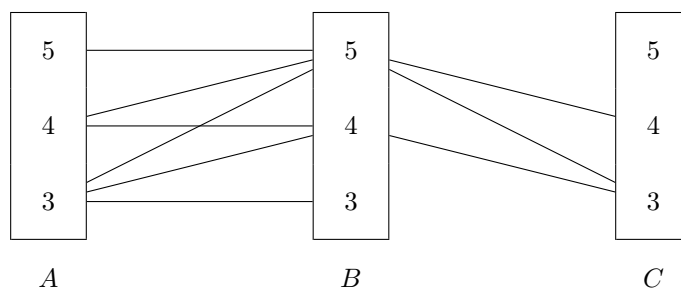
Podmíenku c_4 prevedieme na duálnu premennú v_4 s doménou $\{(2, 1), (3, 0), (3, 1)\}$.

Pre každú dvojicu podmienok c_i a c_j zdieľajúcu premenné, zavedieme binárnu podmienku medzi v_i a v_j , obmedzujúcu duálne premenné na k -tice, v ktorých majú zdieľané premenné rovnakú hodnotu.



Obrázek 1: Diagram duálnych premenných a binárnych podmienok medzi nimi

2.příklad



Obrázek 2: Diagram podpôr jednotlivých hodnôt

Zápis $\langle A, a \rangle$ znamená hodnota a z domény premennej A .

Hodnota $3 \in A$ má podpory $\{\langle B, 3 \rangle, \langle B, 4 \rangle, \langle B, 5 \rangle\}$.

Hodnota $4 \in A$ má podpory $\{\langle B, 4 \rangle, \langle B, 5 \rangle\}$.

Hodnota $5 \in A$ má podpory $\{\langle B, 5 \rangle\}$.

Hodnota $3 \in B$ má podpory $\{\langle A, 3 \rangle\}$.

Hodnota $4 \in B$ má podpory $\{\langle A, 3 \rangle, \langle A, 4 \rangle, \langle C, 3 \rangle\}$.

Hodnota $5 \in B$ má podpory $\{\langle A, 3 \rangle, \langle A, 4 \rangle, \langle A, 5 \rangle, \langle C, 3 \rangle, \langle C, 4 \rangle\}$.

Hodnota $3 \in C$ má podpory $\{\langle B, 4 \rangle, \langle B, 5 \rangle, \}$.

Hodnota $4 \in C$ má podpory $\{\langle B, 5 \rangle\}$.

Hodnota $5 \in C$ nemá žiadnu podporu.

3.příklad

Pravidlá: $\min(Y) \geq \min(X) - C \quad \min(X) \geq \min(Y) + C$
 $\min(X) \geq \min(Y) + C \quad \max(X) \leq \max(Y) + C$

Postup: $Y = X - 3 \Rightarrow \min(Y) \geq 0 - 3, \quad \max(Y) \leq 10 - 3 \Rightarrow Y \in \{0, \dots, 7\}$
 $\Rightarrow \min(X) \geq 0 + 3, \quad \max(X) \leq 10 + 3 \Rightarrow X \in \{3, 4, 5, 8, 9, 10\}$

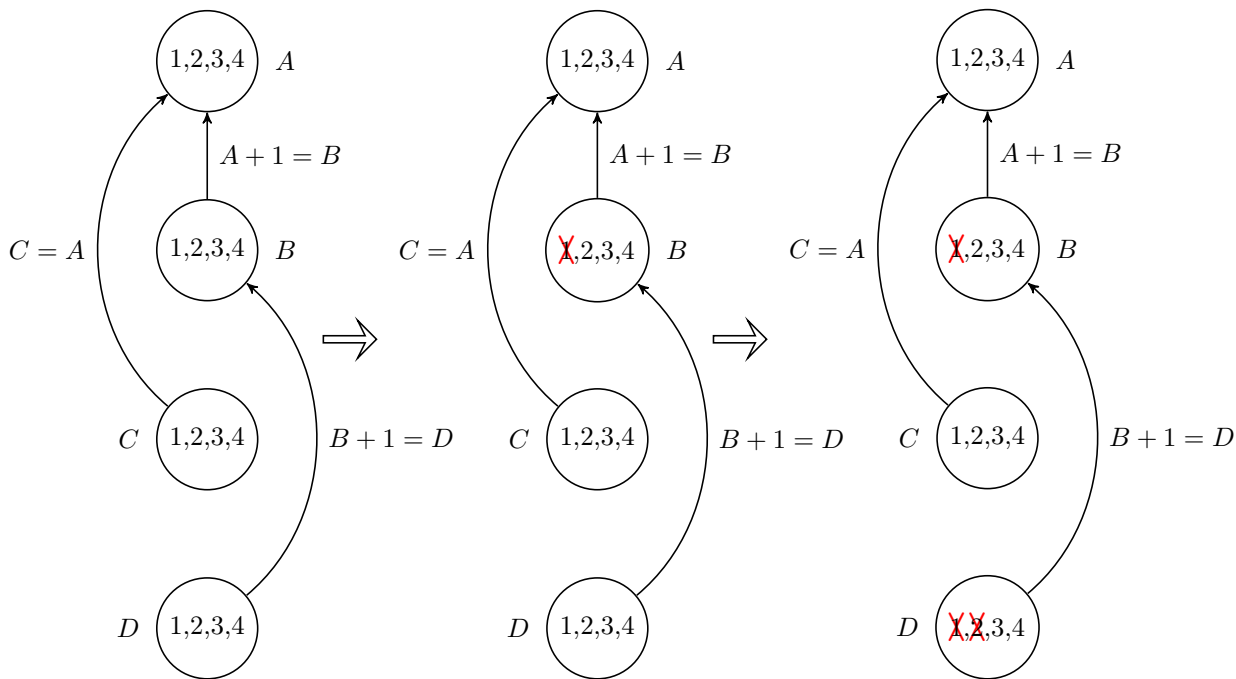
Po ukončení propagácie pri použití konzistencie medzi má premenná Y doménu $\{0, \dots, 7\}$ a premenná X doménu $\{3, 4, 5, 8, 9, 10\}$.

Ak by sme v príklade použili namiesto konzistencie medzi hranovú konzistenciu, hodnoty 3 a 4 by z domény premennej Y vypadly, pretože nemajú žiadnu doménovú podporu.

4.příklad

Algoritmus DAC začína vo vrchole/premennej A (posledná premenná v danom usporiadaní). Následne sa zrevidujú všetky hrany, ktoré do vrcholu A vedú. Pri revízií hrany (B, A) zistíme, že z domény premennej B môžeme odstrániť hodnotu 1 pretože v doméne premennej A neexistuje hodnota, ktorá by s ňou v na základe podmienky c_1 bola konzistentná. Takisto sa zreviduje hrana (C, A) , no žiadnu hodnotu z domény C neodstraňujeme. Pokračujeme vrcholom B a proces sa opakuje. Do vrcholu B vedie len hrana (D, B) . Pri revízií tejto hrany zistíme, že z domény premennej D môžeme odstrániť hodnoty 1 a 2. Ďalej skontrolujeme vrchol C , do ktorého vedie len hrana (D, C) . Po jej revízií nenastáva žiadna zmena. Do posledného vrcholu D už nevedú žiadne hrany a algoritmus teda skončí.

Po algoritme DAC vypadajú teda domény následovne: $A \in \{1, 2, 3, 4\}, B \in \{2, 3, 4\}, C \in \{1, 2, 3, 4\}$ a $D \in \{3, 4\}$.



Obrázek 3: Postup algoritmu DAC

Napriek tomu, že tento problém má niekoľko riešení (napr. $A = 1, B = 2, C = 1$ a $D = 3$), nie je možné nájsť riešenie bez navrátenia vzhľadom na dané usporiadanie premenných. Toto usporiadanie premenných má šírku 2, DAC nám teda na zabezpečenie nájdenie riešenia bez vracania nestačí.

Příklady 8. – zadání

1. Ukažte, jak vypadá graf stavového prostoru pro řešení problému

- proměnné: $A \in \{1, 2, 3\}$, $B \in \{2, 3\}$, $C \in \{1, 2, 3, 4\}$, $D \in \{2, 3, 4\}$
- omezení: $c_1 : A = B$, $c_2 : C \geq B$, $c_3 : D < B$

algoritmem

- (a) backtrackingu
- (b) kontroly dopředu (forward checking)
- (c) pohledu dopředu (look ahead) bez iniciální konzistence

při uspořádání proměnných A, B, C, D a při uspořádání hodnot od nejmenší k největší. Ve vašem řešení uveďte pro každý algoritmus jeden graf stavového prostoru. V každém uzlu grafu uveďte, které domény proměnných se v důsledku propagací změnilly a která omezení tyto změny způsobila.

2. Napište řešení následujícího příkladu v OPL. Kód řádně okomentujte,

Vyřešte následující problém plánování projektu. Projekt se skládá z množiny úkolů \mathcal{U} a pracuje na něm m lidí tak, že každý člověk c pracuje pouze na svých předem zadaných úkolech U_c , tj. $\mathcal{U} = \bigcup_{c=1}^m U_c$ a pro každé dva lidi a, b platí $U_a \cap U_b = \emptyset$. Každý úkol má tedy stanoveného člověka, který na něm pracuje, a dále má stanovenou dobu trvání.

Úkoly jsou na sobě časově závislé. Časové závislosti jsou následujícího typu:

- k -tý úkol a l -tý úkol musí začínat zároveň,
- i -tý úkol musí začít až po ukončení úkolu j -tého úkolu.

Každý člověk c smí pracovat nejvýše na w_c úkolech zároveň.

Určete, kdy mají být jednotlivé úkoly v projektu realizovány tak, aby byl minimalizován součet koncových časů všech úkolů v projektu.

Řešení otestujte na následujícím problému (tj. musíte použít odpovídající datovou sadu, kterou odevzdejte jako součást řešení):

úkol	1	2	3	4	5	6	7	8	9	10	11	12	13	14
doba trvání	3	2	1	4	2	4	2	1	1	3	2	2	1	1
člověk	1	1	1	1	1	2	2	2	2	2	3	3	3	3

Kapacita lidí: $w_1 = 2, w_2 = 2, w_3 = 1$

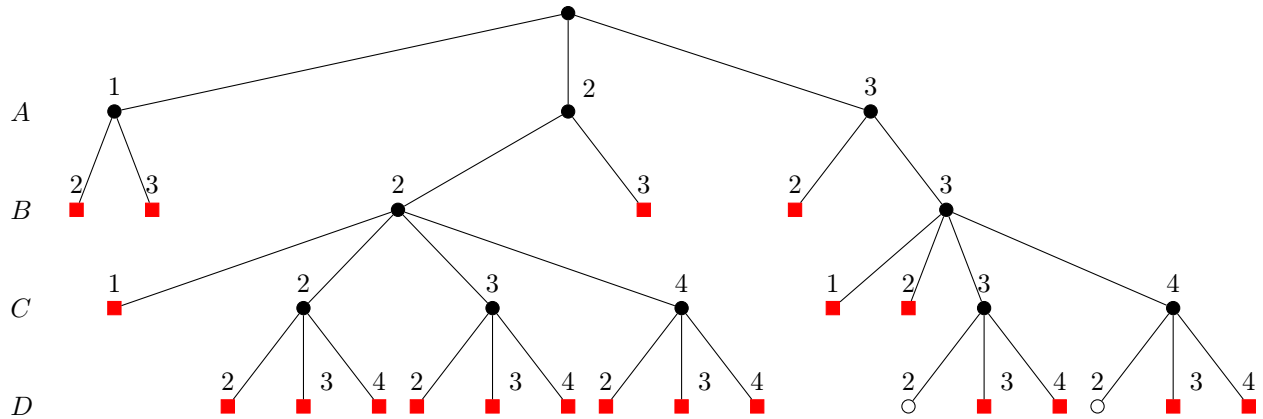
Úkoly začínající zároveň (k a l): 1 a 2, 7 a 8

Úkoly po sobě (i po j): 3 po 1, 6 po 7

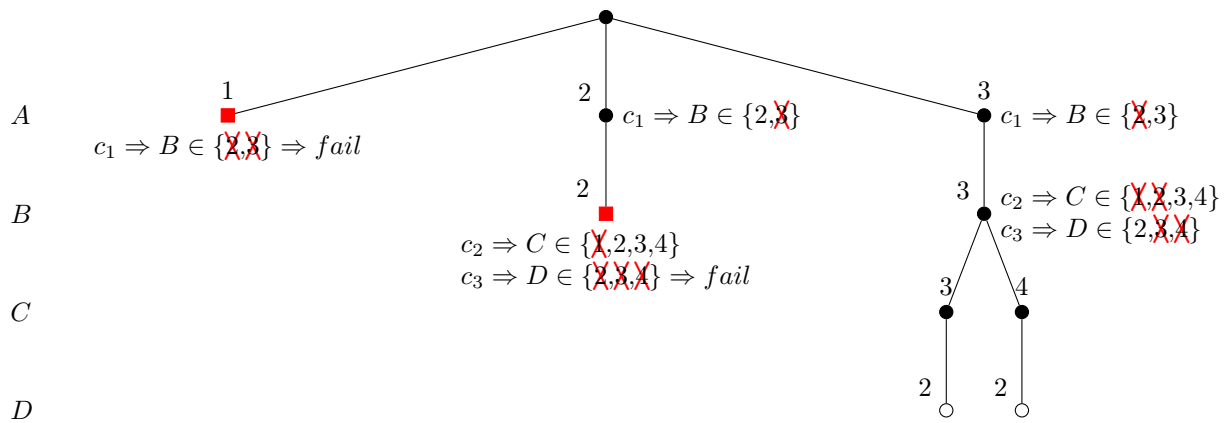
Příklady 8. – řešení

1.příklad

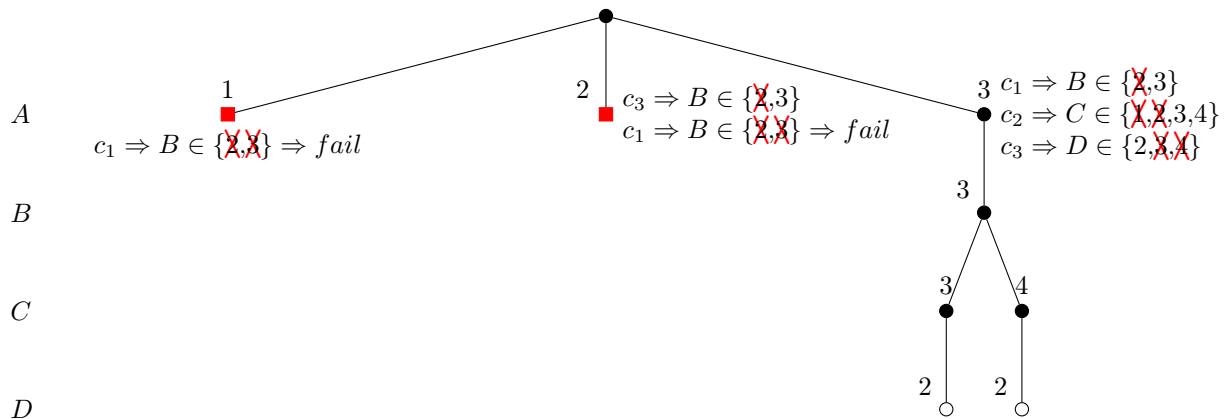
Graf stavového prostoru pro řešení problému algoritmem *backtracking*. Fail na úrovni stromu pro proměnnou *B* nastává z důvodů failu omezení c_1 . Fail na úrovni stromu pro proměnnou *C* nastává z důvodů failu omezení c_2 . Fail na úrovni stromu pro proměnnou *D* nastává z důvodů failu omezení c_3 .



Graf stavového prostoru pro řešení problému algoritmem *forward checking*.



Graf stavového prostoru pro řešení problému algoritmem *look ahead*.



2.příklad

```
/****** * .dat soubor ******/
pocetUkolu = 14;

pocetOsob = 3;

ukoly = [<3,1>,<2,1>,<1,1>,<4,1>,<2,1>,
        <4,2>,<2,2>,<1,2>,<1,2>,<3,2>,
        <2,3>,<2,3>,<1,3>,<1,3>];

limit = [2,2,1];

prec = {<3,1>,<6,7>};

same = {<1,2>,<7,8>};
```

```
/****** *.mod soubor *****/
using CP;

tuple Ukol {
    int trvani;
    int osoba;
}
tuple Prec {
    int Po;
    int Pred;
}
tuple Same {
    int Prvni;
    int Druha;
}

int pocetUkolu = ...;
int pocetOsob = ...;

Ukol ukoly[1..pocetUkolu] = ...;
int limit[1..pocetOsob] = ...;
{Prec} prec = ...;
{Same} same = ...;

// domenove promenne pro startovni casy
dvar interval casy [u in 1..pocetUkolu] size ukoly[u].trvani;

// kumulativni funkce pocitajici vytizeni osob v case
cumulFunction osoby[o in 1..pocetOsob] =
    sum (u in 1..pocetUkolu: ukoly[u].osoba == o) pulse(casy[u],1);

// minimalizace souctu koncovych casu
minimize sum (u in 1..pocetUkolu) endOf(casy[u]);

subject to {

    // omezime maximalni zatizeni kazde osoby v kazdem case
    forall (o in 1..pocetOsob) osoby[o] <= limit[o];

    // precedencni podminky
    forall (p in prec) endBeforeStart(casy[p.Pred],casy[p.Po]);

    // stejne startovni casy
    forall (s in same) startAtStart(casy[s.Prvni],casy[s.Druha]);
}
```

Příklady 9. – zadání

1. Ukažte podrobně, jak pracuje algoritmus pro řešení stromového CSP na následujícím příkladu při uspořádání proměnných $(V1, V2, V3, V4, V5)$:

- všechny proměnné mají doménu $\{0, 1, 2, 3\}$,
- $V1 < V2, V1 * V3 < 2, V4 = V2 + 1, V3 > V5$.

Jako součást řešení uveďte stromovou reprezentaci tohoto CSP problému. Dále odvoďte algoritmem řešení bez navracení *všechna* řešení problému, kdy pro proměnné postupně vybírejte hodnoty od nejmenší k největší.

2. Ukažte podrobně, jak pracuje algoritmus PC-2 na následujícím příkladu a uveďte výsledné domény proměnných.

- Proměnné $X1, X2, X3$ mají doménu $\{0, 1, 2, 3, 4, 5, 6\}$.
- Omezení: $X1 + 2 < X2, X2 + 1 < X3$.

Je nějaké nové omezení odvozeno jako důsledek konzistence po cestě?

3. Vyřešte následující rozvrhovací problém v OPL. V jedné místnosti probíhá výuka několika přednášek, které je třeba rozvrhnout bez překrytí. Přednášky mají stanoven nejdřívejší čas zahájení a nejpozdější čas dokončení. Některé přednášky na sebe navazují, proto musí být dodrženo pořadí těchto přednášek. Výuku poslední přednášky ukončete co nejdříve.

Všechna vstupní data musí být uvedena v samostatném datovém souboru. Při návrhu doménových proměnných, omezení i účelové funkce berte v úvahu efektivitu výpočtu. Řešení otestujte na následujícím problému.

přednáška	1	2	3	4	5
doba trvání	1	2	1	3	1
nejdřívejší čas	8	8	9	10	10
nejpozdější čas	16	16	16	20	20

Precedence: 2 před 1, 3 před 2, 5 před 4.

4. Ukažte, jak vypadá graf stavového prostoru pro řešení problému

- proměnné: $A, B, C, D \in \{0, 1, 2, 3, 4, 5\}$,
- omezení: $c1 : A > 2 * B, c2 : B \geq C, c3 : C = D + 1$

algoritmem

- kontroly dopředu (forward checking) a
- pohledu dopředu (look ahead) bez iniciální konzistence

při uspořádání proměnných A, B, C, D a při uspořádání hodnot od nejmenší k největší. Ve vašem řešení uveďte pro každý algoritmus jeden graf stavového prostoru. V každém uzlu grafu uveďte, nejen které domény proměnných se v důsledku propagací změnily, ale i jak byly tyto domény **postupně** upravovány jednotlivými omezeními.

5. Napište řešení následujícího příkladu v OPL. Je zadán počet dnů a počet zaměstnanců, kteří každý den pracují na jedné směně. Všechny směny jsou stejně dlouhé. Každý zaměstnanec má přitom určeno předem, kdy jeho směna během dne začíná, a každý zaměstnanec pracuje každou směnu právě na jednom z několika alternativních úkolů. Pro každý úkol máme určeno:
- jakou má preferenci,
 - max. počet hodin, který na něm může každý zaměstnanec za celou dobu pracovat a
 - maximální počet zaměstnanců, kteří mohou na úkolu pracovat zároveň.

Přiřaďte úkol každé směně zaměstnance tak, aby byly minimalizován součet preferencí realizovaných hodin úkolů za celou dobu.

Při návrhu doménových proměnných, omezení i účelové funkce berte v úvahu efektivitu výpočtu. Řešení otestujte na následujícím problému (tj. musíte použít odpovídající datovou sadu, kterou odevzdejte jako součást řešení):

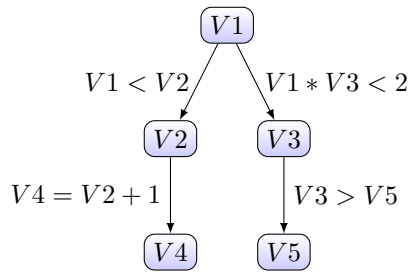
úkol	1	2	3	4
preferance	1	2	3	4
max. počet hodin	8	16	8	16
max. počet zaměstnanců	1	2	1	2

- počet dnů 5, délka dne: 24, délka směny: 8, počet zaměstnanců: k
- počáteční čas směny pro jednotlivé zaměstnance: 6,6,14,14,6,14,6,14,6

Příklady 9. – řešení

1.příklad

Grafová reprezentace:



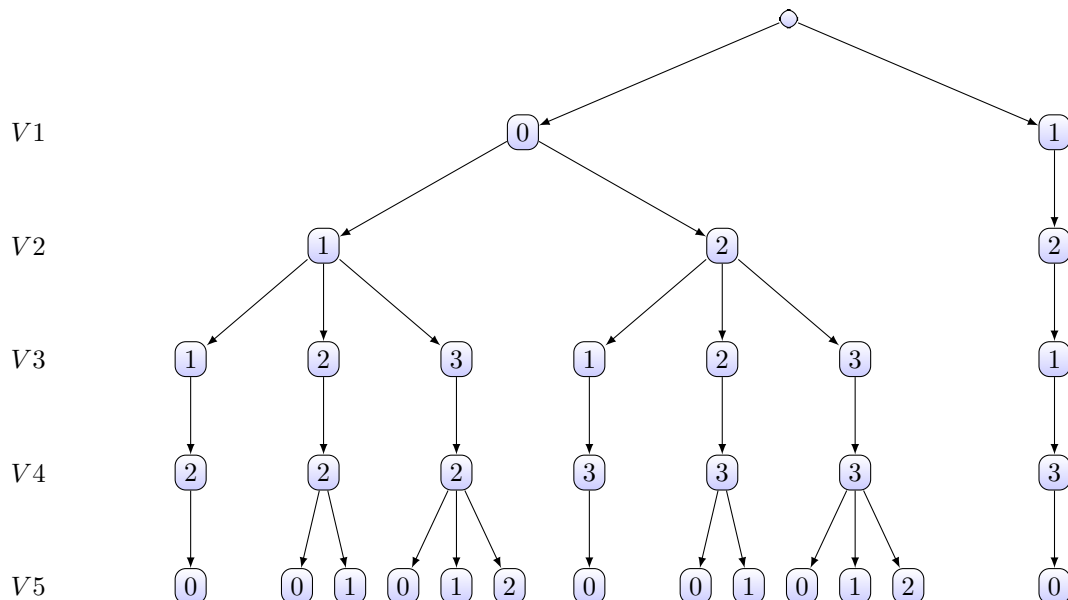
Algoritmus prochází strom od poslední proměnné v uspořádání po první a v každém kroku upravuje (omezuje) doménu rodičovské proměnné.

– domény na začátku: $V1, V2, V3, V4, V5 \in \{0, 1, 2, 3\}$

1. $V3 > V5 \Rightarrow V3 \in \{1, 2, 3\}$
2. $V4 = V2 + 1 \Rightarrow V2 \in \{0, 1, 2\}$
3. $V1 * V3 < 2 \Rightarrow V1 \in \{0, 1\}$
4. $V1 < V2 \Rightarrow V1 \in \{0, 1\}$

– domény na konci: $V1 \in \{0, 1\}, V2 \in \{0, 1, 2\}, V3 \in \{1, 2, 3\}, V4 \in \{1, 2, 3\}, V5 \in \{0, 1, 2, 3\}$

Dále jsou konzistentně instanciovány proměnné v pořadí od $V1$ k $V5$. Je zaručeno, že problém je řešitelný bez navracení díky směrové hranové konzistenci zaručené po zpracování algoritmem pro stromové CSP.



Všechna možná řešení jsou tedy:

V1	V2	V3	V4	V5
0	1	1	2	0
0	1	2	2	0
0	1	2	2	1
0	1	3	2	0
0	1	3	2	1
0	1	3	2	2
0	2	1	3	0
0	2	2	3	0
0	2	2	3	1
0	2	3	3	0
0	2	3	3	1
0	2	3	3	2
1	2	1	3	0

2.příklad

Jsou inicializovány binární relace R_{ij} mezi dvojicemi i, j na základě omezení a domén ze zadání:

$X1$	$\{0, \dots, 6\}$	<ul style="list-style-type: none"> • $X1 + 2 < X2$ • $X2 + 1 < X3$
$X2$	$\{0, \dots, 6\}$	
$X3$	$\{0, \dots, 6\}$	

$R_{1,2}$	$\{(0, 3), (0, 4), (0, 5), (0, 6), (1, 4), (1, 5), (1, 6), (2, 5), (2, 6), (3, 6)\}$
$R_{1,3}$	$\{(i, j) i, j \in \{0, 1, 2, 3, 4, 5, 6\}\}$
$R_{2,3}$	$\{(0, 2), (0, 3), (0, 4), (0, 5), (0, 6), (1, 3), (1, 4), (1, 5), (1, 6), (2, 4), (2, 5), (2, 6), (3, 5), (3, 6), (4, 6)\}$

Do fronty Q algoritmu PC-2 jsou iniciálně vloženy tyto cesty pro revizi:

Q	$[(1, 3, 2), (1, 2, 3), (2, 1, 3)]$
-----	-------------------------------------

Z fronty Q jsou odstraňovány trojice vrcholů (i, m, j) pro revizi, dokud není fronta prázdná. Když dojde k revizi, do fronty jsou přidány všechny trojice vrcholů, které mohou být ovlivněny právě provedenou revizí a které se ve frontě ještě nevskytují. Revize postupně omezují domény proměnných odstraňováním prvků z binárních relací R_{ij} .

Inicializace

fronta Q
$[(1, 3, 2), (1, 2, 3), (2, 1, 3)]$

$R_{1,2}$				
(0, 3)	(0, 4)	(0, 5)	(0, 6)	
	(1, 4)	(1, 5)	(1, 6)	
		(2, 5)	(2, 6)	
			(3, 6)	

$R_{2,3}$					
(0, 2)	(0, 3)	(0, 4)	(0, 5)	(0, 6)	
	(1, 3)	(1, 4)	(1, 5)	(1, 6)	
		(2, 4)	(2, 5)	(2, 6)	
			(3, 5)	(3, 6)	
				(4, 6)	

$R_{1,3}$						
(0, 0)	(0, 1)	(0, 2)	(0, 3)	(0, 4)	(0, 5)	(0, 6)
(1, 0)	(1, 1)	(1, 2)	(1, 3)	(1, 4)	(1, 5)	(1, 6)
(2, 0)	(2, 1)	(2, 2)	(2, 3)	(2, 4)	(2, 5)	(2, 6)
(3, 0)	(3, 1)	(3, 2)	(3, 3)	(3, 4)	(3, 5)	(3, 6)
(4, 0)	(4, 1)	(4, 2)	(4, 3)	(4, 4)	(4, 5)	(4, 6)
(5, 0)	(5, 1)	(5, 2)	(5, 3)	(5, 4)	(5, 5)	(5, 6)
(6, 0)	(6, 1)	(6, 2)	(6, 3)	(6, 4)	(6, 5)	(6, 6)

1. iterace

revidovaná trojice	původní fronta	nové trojice	nová fronta
(1, 3, 2)	[(1, 3, 2), (1, 2, 3), (2, 1, 3)]	[(3, 1, 2), (3, 2, 1)]	[(1, 2, 3), (2, 1, 3)]

$R_{1,2}$			
(0, 3)	(0, 4)	(0, 5)	(0, 6)
	(1, 4)	(1, 5)	(1, 6)
		(2, 5)	(2, 6)
			(3, 6)

$R_{2,3}$				
(0, 2)	(0, 3)	(0, 4)	(0, 5)	(0, 6)
	(1, 3)	(1, 4)	(1, 5)	(1, 6)
		(2, 4)	(2, 5)	(2, 6)
			(3, 5)	(3, 6)
				(4, 6)

$R_{1,3}$						
(0, 0)	(0, 1)	(0, 2)	(0, 3)	(0, 4)	(0, 5)	(0, 6)
(1, 0)	(1, 1)	(1, 2)	(1, 3)	(1, 4)	(1, 5)	(1, 6)
(2, 0)	(2, 1)	(2, 2)	(2, 3)	(2, 4)	(2, 5)	(2, 6)
(3, 0)	(3, 1)	(3, 2)	(3, 3)	(3, 4)	(3, 5)	(3, 6)
(4, 0)	(4, 1)	(4, 2)	(4, 3)	(4, 4)	(4, 5)	(4, 6)
(5, 0)	(5, 1)	(5, 2)	(5, 3)	(5, 4)	(5, 5)	(5, 6)
(6, 0)	(6, 1)	(6, 2)	(6, 3)	(6, 4)	(6, 5)	(6, 6)

2. iterace

revidovaná trojice	původní fronta	nové trojice	nová fronta
(1, 2, 3)	[(1, 2, 3), (2, 1, 3)]	[(2, 1, 3), (2, 3, 1)]	[(2, 1, 3), (2, 3, 1)]

$R_{1,2}$			
(0, 3)	(0, 4)	(0, 5)	(0, 6)
	(1, 4)	(1, 5)	(1, 6)
		(2, 5)	(2, 6)
			(3, 6)

$R_{2,3}$				
(0, 2)	(0, 3)	(0, 4)	(0, 5)	(0, 6)
	(1, 3)	(1, 4)	(1, 5)	(1, 6)
		(2, 4)	(2, 5)	(2, 6)
			(3, 5)	(3, 6)
				(4, 6)

$R_{1,3}$						
(0, 0)	(0, 1)	(0, 2)	(0, 3)	(0, 4)	(0, 5)	(0, 6)
(1, 0)	(1, 1)	(1, 2)	(1, 3)	(1, 4)	(1, 5)	(1, 6)
(2, 0)	(2, 1)	(2, 2)	(2, 3)	(2, 4)	(2, 5)	(2, 6)
(3, 0)	(3, 1)	(3, 2)	(3, 3)	(3, 4)	(3, 5)	(3, 6)
(4, 0)	(4, 1)	(4, 2)	(4, 3)	(4, 4)	(4, 5)	(4, 6)
(5, 0)	(5, 1)	(5, 2)	(5, 3)	(5, 4)	(5, 5)	(5, 6)
(6, 0)	(6, 1)	(6, 2)	(6, 3)	(6, 4)	(6, 5)	(6, 6)

3. iterace

revidovaná trojice	původní fronta	nové trojice	nová fronta
(2, 1, 3)	[(2, 1, 3), (2, 3, 1)]	[(1, 2, 3), (1, 3, 2)]	[(2, 3, 1), (1, 2, 3)]

$R_{1,2}$			
(0, 3)	(0, 4)	(0, 5)	(0, 6)
	(1, 4)	(1, 5)	(1, 6)
		(2, 5)	(2, 6)
			(3, 6)

$R_{2,3}$				
(0, 2)	(0, 3)	(0, 4)	(0, 5)	(0, 6)
	(1, 3)	(1, 4)	(1, 5)	(1, 6)
		(2, 4)	(2, 5)	(2, 6)
			(3, 5)	(3, 6)
				(4, 6)

$R_{1,3}$						
(0, 0)	(0, 1)	(0, 2)	(0, 3)	(0, 4)	(0, 5)	(0, 6)
(1, 0)	(1, 1)	(1, 2)	(1, 3)	(1, 4)	(1, 5)	(1, 6)
(2, 0)	(2, 1)	(2, 2)	(2, 3)	(2, 4)	(2, 5)	(2, 6)
(3, 0)	(3, 1)	(3, 2)	(3, 3)	(3, 4)	(3, 5)	(3, 6)
(4, 0)	(4, 1)	(4, 2)	(4, 3)	(4, 4)	(4, 5)	(4, 6)
(5, 0)	(5, 1)	(5, 2)	(5, 3)	(5, 4)	(5, 5)	(5, 6)
(6, 0)	(6, 1)	(6, 2)	(6, 3)	(6, 4)	(6, 5)	(6, 6)

4. iterace

revidovaná trojice	původní fronta	nové trojice	nová fronta
(2, 3, 1)	[(2, 3, 1), (1, 2, 3)]	∅	[(1, 2, 3)]

Žádná další změna už nenastala.

5. iterace

revidovaná trojice	původní fronta	nové trojice	nová fronta
(1, 2, 3)	[(1, 2, 3)]	∅	∅

Žádná další změna už nenastala.

Výsledné relace jsou tedy následující:

- $R_{1,2} \in \{(0, 3), (0, 4), (1, 4)\}$,
- $R_{2,3} \in \{(3, 5), (3, 6), (4, 6)\}$,
- $R_{1,3} \in \{(0, 5), (0, 6), (1, 6)\}$.

Výsledné domény proměnných jsou:

- $X_1 \in \{0, 1\}$
- $X_2 \in \{3, 4\}$
- $X_3 \in \{5, 6\}$

Je nové přidáno omezení určené relací $R_{1,3} \in \{(0, 5), (0, 6), (1, 6)\}$.

3.příklad

```
/****** * .dat soubor ******/
number = 5;

duration = [1,2,1,3,1];
earliest = [8,8,9,10,10];
latest = [16,16,16,20,20];

precedences = {<2,1>, <3,2>, <5,4>};

/****** * .mod soubor ******/
using CP;

int number = ...;

range Lectures = 1..number;

int duration[Lectures] = ...;
int earliest[Lectures] = ...;
int latest[Lectures] = ...;

tuple Prec {
    int Before;
    int After;}

{Prec} precedences = ...;

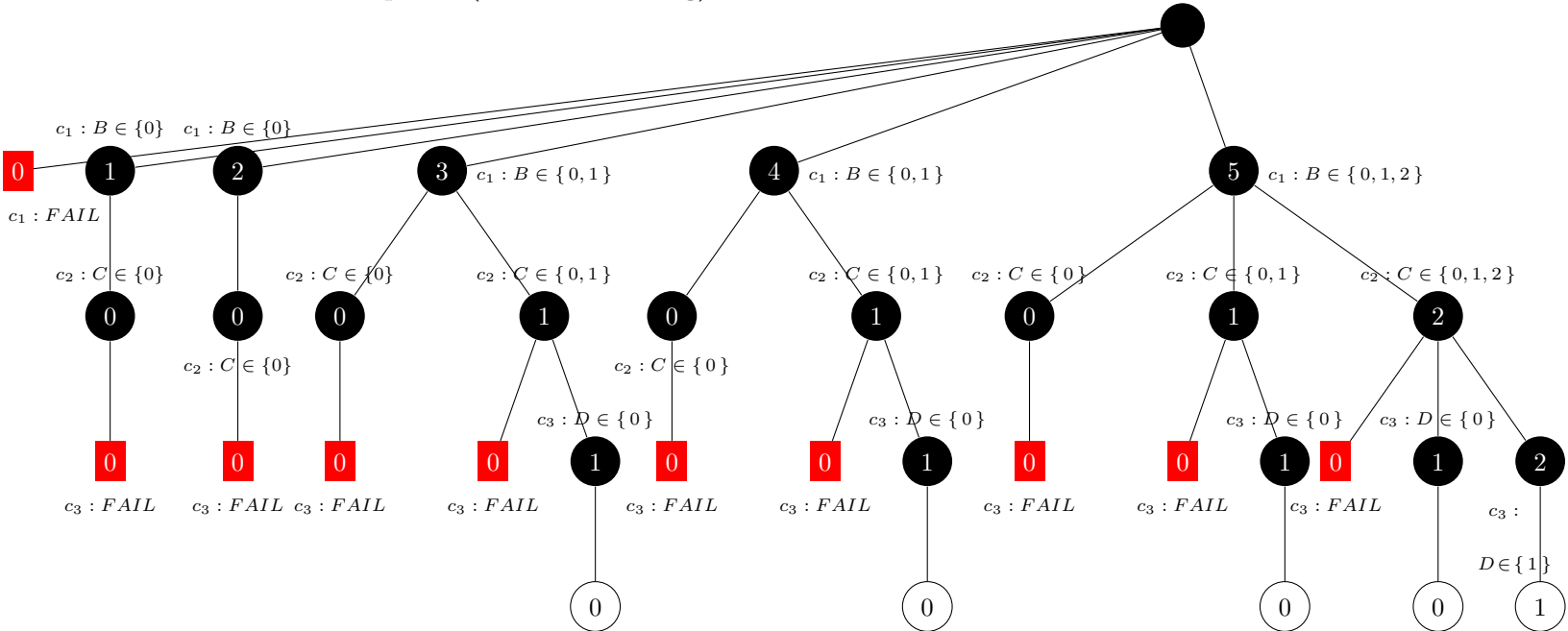
dvar interval start[l in Lectures] in earliest[l]..latest[l] size duration[l];
dvar sequence seq in start;

minimize max(l in Lectures) endOf(start[l]);

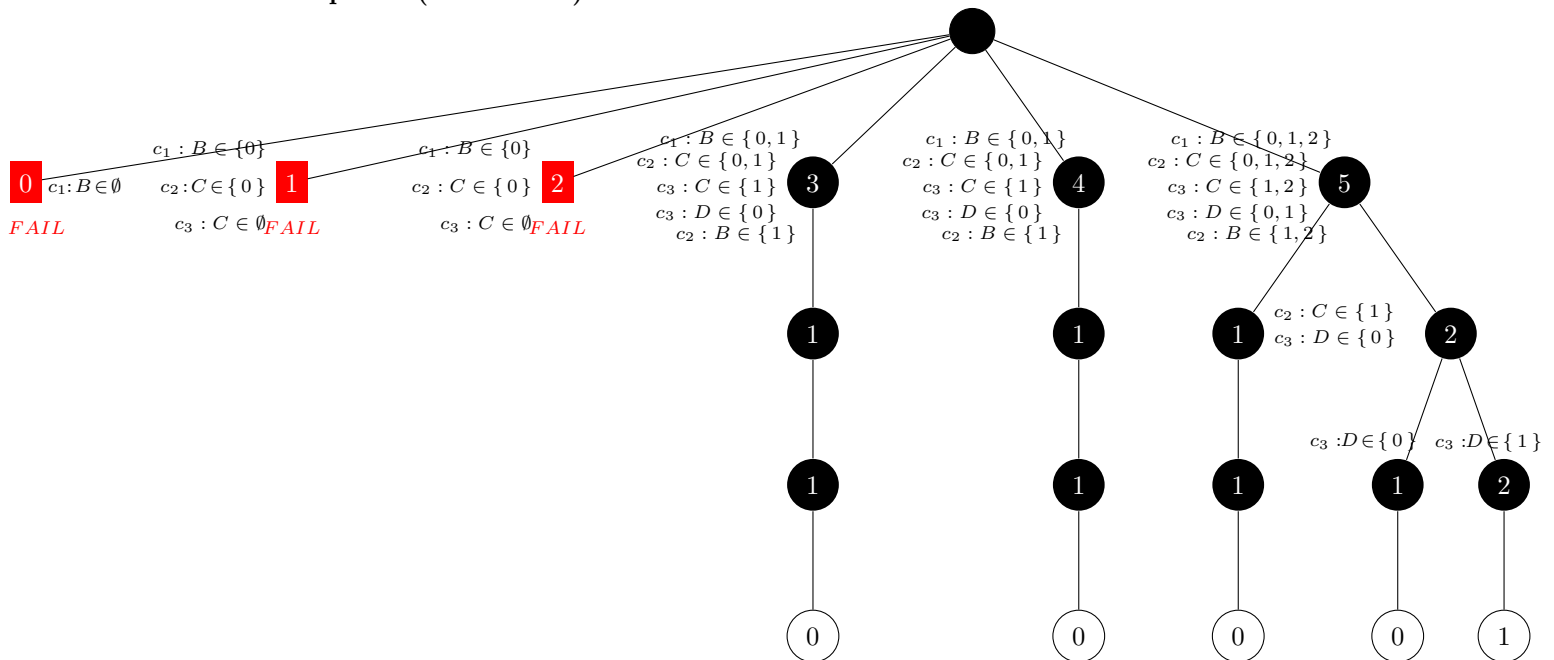
subject to {
    noOverlap(seq);
    forall (p in precedences) endBeforeStart(start[p.Before],start[p.After]);
}
```

4.příklad

Kontrola dopředu (forward checking)



Pohled dopředu (look ahead) bez iniciální konzistence



5.příklad

```
/****** *.dat soubor * * *****/
pocetDnu = 5;
pocetZamestnancu = 9;
pocetUkolu = 4;
trvani = 8;

zamestnanci = [<1,6>,<2,6><3,14>,<4,14>,<5,6>,<6,14>,<7,6>,<8,14>,<9,6>];

ukoly = [<1,1,8,1>,<2,2,16,2>,<3,1,8,3>,<4,2,16,4>];
```

```

/***** *.mod soubor * *****/

using CP;

int pocetZamestnancu = ...;
int pocetDny = ...;
int pocetUkolu = ...;
int trvani = ...;

range Zamestnanci = 1..pocetZamestnancu;
range Dny = 1..pocetDny;
range Ukoly = 1..pocetUkolu;

tuple Ukol {
    key int id;
    int pocet;
    int hodin;
    int preference;
}

tuple Zamestnanec {
    key int id;
    int od;
}

Ukol ukoly[Ukoly] = ...;
Zamestnanec zamestnanci[Zamestnanci] = ...;

dvar interval casy[z in Zamestnanci][d in Dny]
    in zamestnanci[z].od .. (zamestnanci[z].od+trvani) size trvani;

dvar interval prirazeni[z in Zamestnanci][d in Dny][u in Ukoly] optional;

cumulFunction pocetUkol[u in Ukoly][d in Dny] =
    sum (z in Zamestnanci) pulse(prirazeni[z][d][u],1);

cumulFunction hodinUkol[u in Ukoly][z in Zamestnanci] =
    sum (d in Dny) pulse(prirazeni[z][d][u],trvani);

minimize sum (z in Zamestnanci, d in Dny, u in Ukoly)
    presenceOf(prirazeni[z][d][u])*ukoly[u].preference;

subject to {

    forall (z in Zamestnanci, d in Dny)
        alternative(casy[z][d], all (u in Ukoly) prirazeni[z][d][u]);

    forall (u in Ukoly, d in Dny)
        pocetUkol[u][d] <= ukoly[u].pocet;

    forall (z in Zamestnanci, u in Ukoly)
        hodinUkol[u][z] <= ukoly[u].hodin;
}

```


Příklady 10. – zadání

- Uved'te všechny podpory hodnot pro následující problém
 - proměnné A, B, C, D mohou nabývat hodnot 0, 1, 2
 - $A = B + 1, B \leq C - 1, C = D$.
- Uved'te jednotlivé kroky algoritmu DAC na následujícím příkladu při uspořádání proměnných A, B, C, D, E . Proměnné a omezení: $A, B, C, D, E \in \{0, 1, 2, 3\}$

$$\begin{aligned} c_1 : A &= B \\ c_2 : C &\leq A \\ c_3 : D &= B + 1 \\ c_4 : E &> D \end{aligned}$$

Je zaručeno, že najdeme řešení bez navracení (zdůvodněte Vaši odpověď)? Pokud ano, uveďte postup, jakým způsobem najdeme jednotlivá řešení bez navracení (všechna tato řešení v uvedeném postupu napište).

- Napište řešení následujícího příkladu v OPL.

Máme zadány obchody, ze kterých můžeme odebírat zboží, a několik poboček, kde každá z nich bude odebírat zboží právě z jednoho obchodu. U každé pobočky máme přitom stanoveno, jaká je cena za odběr z daného obchodu. Dále má každý obchod stanoven maximální počet poboček, které může obsluhovat (tj. kapacita obchodu). Určete pro každou pobočku jeden obchod, který ji bude obsluhovat tak, aby byla minimalizována cena za odběr zboží z obchodů všemi pobočkami.

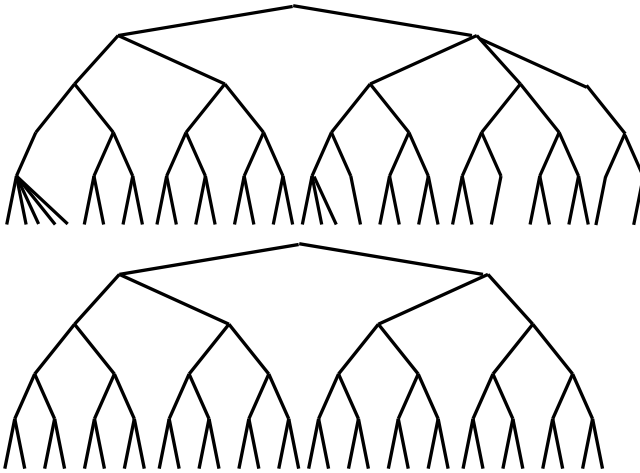
Řešení otestujte na následujícím problému (tj. musíte použít odpovídající datovou sadu, kterou odevzdejte jako součást řešení). Data pro tento problém jsou také k dispozici v souboru `priklady_10_obchody_vstup.txt`.

obchody	Tesco	Albert	Penny	Billa	Lidl
kapacita	1	4	2	1	3
pobočka 1	20	24	11	25	30
pobočka 2	28	27	82	83	74
pobočka 3	74	97	71	96	70
pobočka 4	2	55	73	69	61
pobočka 5	46	96	59	83	4
pobočka 6	42	22	29	67	59
pobočka 7	1	5	73	59	56
pobočka 8	10	73	13	43	96
pobočka 9	93	35	63	85	46
pobočka 10	47	65	55	71	95

Příklad neoptimálního řešení – pobočka/obchod: 1/Tesco, 2/Albert, 3/Albert, 4/Albert, 5/Albert, 6/Penny, 7/Penny, 8/Billa, 9/Lidl, 10/Lidl;

cena tohoto řešení: $20+27+97+55+96+29+73+43+46+95=581$

- Na prvním stromě ukažte, které větve projde kombinovaný algoritmus *omezení počtu návratů* (*bounded-backtrack search BBS*) a *omezení hloubky* (*depth-bounded search DBS*) ve variantě DBS(2, BBS(3)). Na druhém stromě uveďte průchod stromu pro DDS(4), tj. prohledávání s hloubkou omezenými diskrepancemi (*depth-bounded discrepancy search, DDS*). Uveďte, v jakém pořadí jsou prozkoumány jednotlivé větve při výběru hodnot zleva.



5. Jaká je *úroveň konzistence* pro následující *fuzzy CSP* problém P s omezeními c_1 , c_2 a c_3 a jaké dostaneme řešení? Jaká je *úroveň splnění* pro problém $P_1 = (\{c_1, c_2\}, \{A\})$, tj. pro $(c_1 \oplus c_2) \Downarrow_{\{A\}}$? Uveďte postup řešení.

$$P = (\{c_1, c_2, c_3\}, \{A, B\})$$

$$\text{dom}(A) = \text{dom}(B) = \{x, y\}$$

$$\text{con}(c_1) = \{A, B\}, \quad \text{def}(x, x) = 0.1, \quad \text{def}(x, y) = 0.2, \quad \text{def}(y, x) = 0.4, \quad \text{def}(y, y) = 0.8$$

$$\text{con}(c_2) = \{A\}, \quad \text{def}(x) = 0.7, \quad \text{def}(y) = 0.9$$

$$\text{con}(c_3) = \{B\}, \quad \text{def}(x) = 0.5, \quad \text{def}(y) = 0.6$$

6. Napište řešení následujícího příkladu v OPL. V tovární hale se montují výrobky. Každý výrobek se postupně montuje na všech strojích ve stejném pořadí od prvního do posledního, tj. na každém stroji je prováděna právě jedna operace pro každý výrobek a operace jednoho výrobku jsou prováděny postupně jedna za druhou bez překrytí. Doba trvání každé operace je zadána. Na každém stroji se v daném čase může provádět nejvýše jedna operace. Každá operace vyžaduje ke svému provádění jednoho robota, přičemž má předem zadánu množinu robotů, které může použít. Každý robot může v daném čase pracovat na nejvýše jedné operaci.

Každá operace vyžaduje ke svému provádění zadaný počet zaměstnanců. Každý zaměstnanec může pracovat nejvýše na jedné operaci a k dispozici je omezený počet zaměstnanců. Určete *pro každou operaci její startovní čas a robota*, který na ní bude pracovat tak, aby byl minimalizován čas dokončení celé výroby (není nutné určit, kteří zaměstnanci budou na dané operaci pracovat, stačí, aby byl zajištěn jejich dostatečný počet).

Řešení otestujte na následujícím problému (tj. musíte použít odpovídající datovou sadu, kterou odevzdejte jako součást řešení). Pro první (jednodušší) variantu použijte první tři výrobky, pro druhou variantu použijte všech pět výrobků.

operace	výrobek	stroj	doba trvání	možní roboti	počet zaměstnanců
11	1	1	1	1	3
12	1	2	2	1	2
13	1	3	3	1	2
21	2	1	4	2	2
22	2	2	5	1,2	1
23	2	3	6	2	2
31	3	1	6	1	4
32	3	2	5	1	1
33	3	3	4	1	5
41	4	1	2	1,2	5
42	4	2	2	1,2	5
43	4	3	2	1,2	5
51	5	1	4	2	3
52	5	2	7	1,2	3
53	5	3	6	2	3

pocetStroju = 3; pocetRobotu = 2; pocetZamestnancu = 5;
 varianta 1: pocetVyrobku = 3; varianta 2: pocetVyrobku = 5;

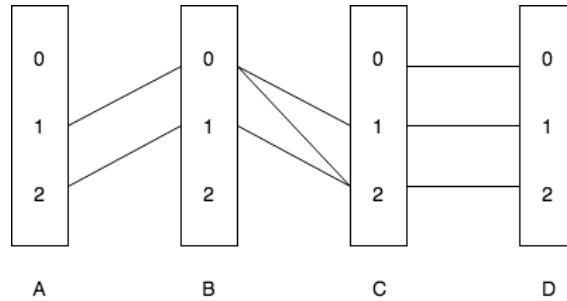
Příklad řešení pro menší problém (operace: start, robot) s časem dokončení 22

11: 0,1; 12: 1,1; 13: 3,1 21: 1,2; 22: 5,2; 23: 12,2 31: 6,1; 32: 12,1; 33: 18,1

Příklady 10. – řešení

1.příklad

Diagram podpôr jednotlivých hodnôt:



Podpory hodnôt:

- A
 - podpory pre hodnotu $0 \in A$: nemá podporu
 - podpory pre hodnotu $1 \in A$: $\langle B, 0 \rangle$
 - podpory pre hodnotu $2 \in A$: $\langle B, 1 \rangle$
- B
 - podpory pre hodnotu $0 \in B$: $\langle A, 1 \rangle, \langle C, 1 \rangle, \langle C, 2 \rangle$
 - podpory pre hodnotu $1 \in B$: $\langle A, 2 \rangle, \langle C, 2 \rangle$
 - podpory pre hodnotu $2 \in B$: nemá podporu
- C
 - podpory pre hodnotu $0 \in C$: $\langle D, 0 \rangle$
 - podpory pre hodnotu $1 \in C$: $\langle B, 0 \rangle, \langle D, 1 \rangle$
 - podpory pre hodnotu $2 \in C$: $\langle B, 0 \rangle, \langle B, 1 \rangle, \langle D, 2 \rangle$
- D
 - podpory pre hodnotu $0 \in D$: $\langle C, 0 \rangle$
 - podpory pre hodnotu $1 \in D$: $\langle C, 1 \rangle$
 - podpory pre hodnotu $2 \in D$: $\langle C, 2 \rangle$

2.příklad

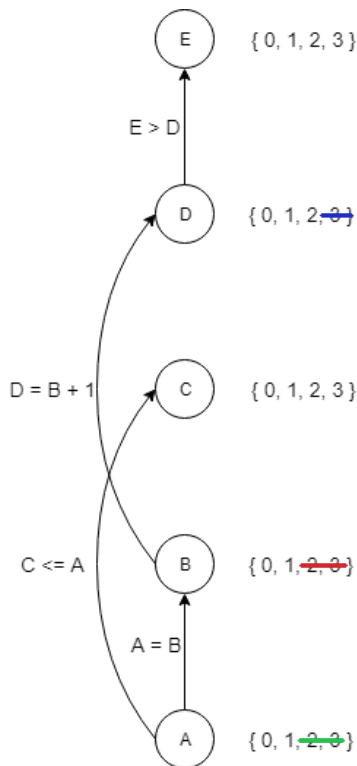
Algoritmus DAC pre dané usporiadanie premenných x_1 až x_n postupne prechádza premenné x_i v poradí x_n až x_1 a pre každú premennú reviduje hrany, ktoré vedú do vrchola reprezentujúceho x_i .

Kroky algoritmu DAC pre zadaný príklad a usporiadanie premenných A, B, C, D, E:

1. Vzhľadom na usporiadanie premenných algoritmus začína vo vrchole E. Do vrcholu E vedie jediná hrana, po jej zrevidovaní je z domény premennej D odstránená hodnota 3. ($c_4 : E > D$)
2. Nasleduje vrchol D, do ktorého vedie jediná hrana a to z vrcholu B. Zrevidovaním tejto hrany sa z domény premennej B odstránia hodnoty 2 a 3. ($c_3 : D = B + 1$)
3. Nasleduje vrchol C, do ktorého vedie hrana z vrcholu A, jej zrevidovaním sa doména premennej A nijako nezmení. ($c_2 : C \leq A$)
4. Nasleduje vrchol B, do ktorého vedie jediná hrana z vrcholu A. Jej zrevidovaním sú z domény premennej A odstránené hodnoty 2 a 3. ($c_1 : A = B$)
5. Do vrcholu A nevedie žiadna hrana, algoritmus končí.

Výsledné domény:

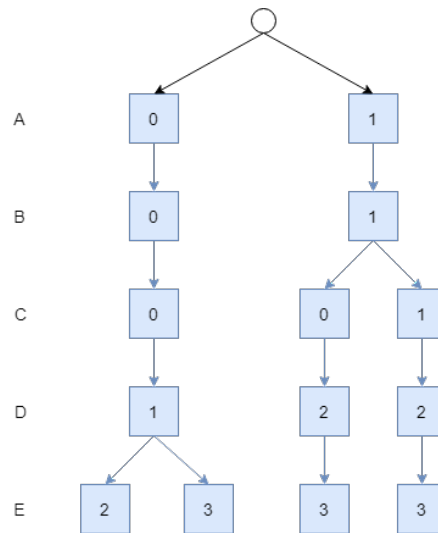
$A \in \{0, 1\}, B \in \{0, 1\}, C \in \{0, 1, 2, 3\}, D \in \{0, 1, 2\}, E \in \{0, 1, 2, 3\}$



Je zaručené, že najdeme riešenie bez navrátenia. Zdôvodnenie:

- Po aplikácii DAC je daný CSP smerovo hranovo konzistentný.
- Pre dané usporiadanie je šírka grafu, ktorým je usporiadanie reprezentované 1.
- Pretože graf má šírku 1, je to strom.
- Teda máme usporiadanie stromového grafu podmienok, ktorý je smerovo hranovo konzistentný vzhľadom k tomuto usporiadaniu. Preto má daný CSP riešenie bez navrátenia.

Nájdienie riešení bez navrátenia (v grafe stavového priestoru nie sú žiadne listy slepej vetvy):



Výsledné riešenia:

$(A, B, C, D, E) \in \{(0, 0, 0, 1, 2), (0, 0, 0, 1, 3), (1, 1, 0, 2, 3), (1, 1, 1, 2, 3)\}$

3.příklad

```

/***** *.dat soubor * *****/

Shops = {Tesco, Albert, Penny, Billa, Lidl};
NbBranches = 10;
Capacity = [1,4,2,1,3];

SupplyCost = [
    [ 20, 24, 11, 25, 30 ],
    [ 28, 27, 82, 83, 74 ],
    [ 74, 97, 71, 96, 70 ],
    [  2, 55, 73, 69, 61 ],
    [ 46, 96, 59, 83,  4 ],
    [ 42, 22, 29, 67, 59 ],
    [  1,  5, 73, 59, 56 ],
    [ 10, 73, 13, 43, 96 ],
    [ 93, 35, 63, 85, 46 ],
    [ 47, 65, 55, 71, 95 ] ];

/***** *.mod soubor * *****/

using CP;

{string} Shops = ...;
int NbBranches = ...;
range Branches = 0..NbBranches-1;

int Capacity[Shops] = ...;           // maximum number branches assigned
to each shop
int SupplyCost[Branches][Shops] = ...; // supply cost between each branchee
and each shop

dvar boolean Supply[Branches][Shops]; // 1 if store supplied by shop, 0
otherwise

dvar int+ TotalCost; // additional domain variable for the optimization cost

minimize TotalCost;

subject to {

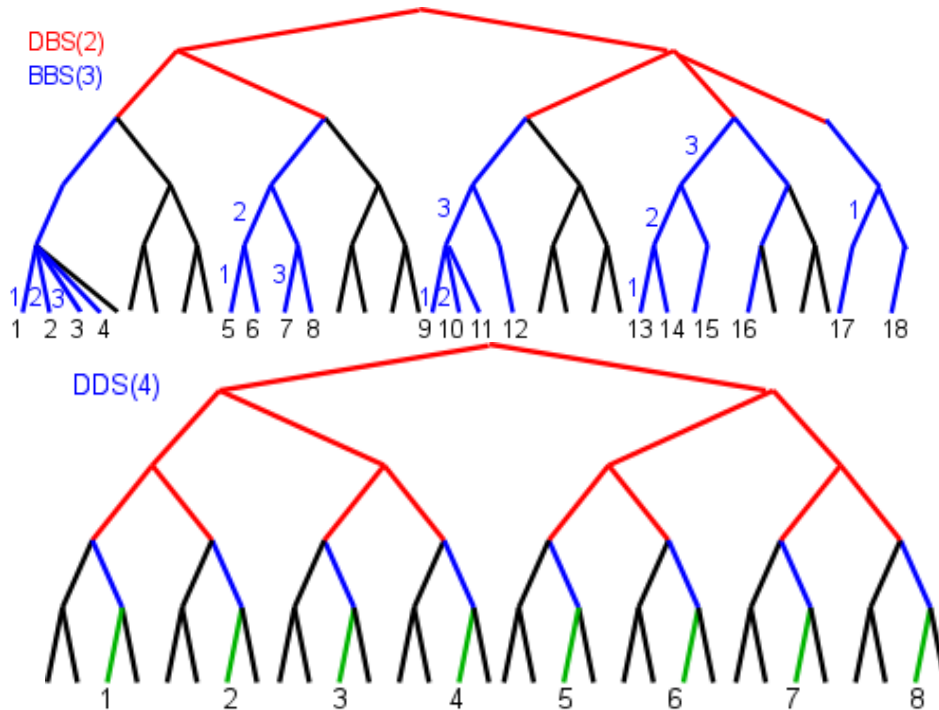
    forall( s in Branches )
        sum( w in Shops ) Supply[s][w] == 1;

    forall( w in Shops )
        sum( s in Branches ) Supply[s][w] <= Capacity[w];

    TotalCost == sum( w in Shops , s in Branches ) SupplyCost[s][w] * Supply[s][w];
}

```

4.příklad



5.příklad

Úroveň konzistence problému P spočítáme jako projekci kombinace všech omezení na prázdnou množinu proměnných $\oplus C \downarrow_{\emptyset}$ (kde $\oplus C = c_1 \oplus c_2 \oplus c_3$):

$$\begin{aligned} \oplus C \downarrow_{\emptyset} &= \max\{\mu_{\oplus C}(x, x), \mu_{\oplus C}(x, y), \mu_{\oplus C}(y, x), \mu_{\oplus C}(y, y)\} \\ \mu_{\oplus C}(x, x) &= \min\{\mu_{c_1}(x, x), \mu_{c_2}(x), \mu_{c_3}(x)\} = \min\{0.1, 0.7, 0.5\} = 0.1 \\ \mu_{\oplus C}(x, y) &= \min\{\mu_{c_1}(x, y), \mu_{c_2}(x), \mu_{c_3}(y)\} = \min\{0.2, 0.7, 0.6\} = 0.2 \\ \mu_{\oplus C}(y, x) &= \min\{\mu_{c_1}(y, x), \mu_{c_2}(y), \mu_{c_3}(x)\} = \min\{0.4, 0.9, 0.5\} = 0.4 \\ \mu_{\oplus C}(y, y) &= \min\{\mu_{c_1}(y, y), \mu_{c_2}(y), \mu_{c_3}(y)\} = \min\{0.8, 0.9, 0.6\} = 0.6 \\ \oplus C \downarrow_{\emptyset} &= \max\{0.1, 0.2, 0.4, 0.6\} = 0.6 \end{aligned}$$

Úroveň konzistence problému P je tedy 0.6 a dostaneme řešení $\{A/y, B/y\}$.

Úroveň splnění pro problém P_1 spočítáme jako $\oplus C' \downarrow_{\{A\}}$ (kde $\oplus C' = c_1 \oplus c_2$):

$$\begin{aligned} \oplus C' \downarrow_{\{A\}}(x) &= \max\{\mu_{\oplus C'}(x, x), \mu_{\oplus C'}(x, y)\} \\ \oplus C' \downarrow_{\{A\}}(y) &= \max\{\mu_{\oplus C'}(y, x), \mu_{\oplus C'}(y, y)\} \\ \mu_{\oplus C'}(x, x) &= \min\{\mu_{c_1}(x, x), \mu_{c_2}(x)\} = \min\{0.1, 0.7\} = 0.1 \\ \mu_{\oplus C'}(x, y) &= \min\{\mu_{c_1}(x, y), \mu_{c_2}(x)\} = \min\{0.2, 0.7\} = 0.2 \\ \mu_{\oplus C'}(y, x) &= \min\{\mu_{c_1}(y, x), \mu_{c_2}(y)\} = \min\{0.4, 0.9\} = 0.4 \\ \mu_{\oplus C'}(y, y) &= \min\{\mu_{c_1}(y, y), \mu_{c_2}(y)\} = \min\{0.8, 0.9\} = 0.8 \\ \oplus C' \downarrow_{\{A\}}(x) &= \max\{0.1, 0.2\} = 0.2 \\ \oplus C' \downarrow_{\{A\}}(y) &= \max\{0.4, 0.8\} = 0.8 \end{aligned}$$

6.příklad

```

/***** *.dat soubor * * *****/

pocetStroju = 3;
pocetRobotu = 2;
pocetVyrobku = 5;
pocetZamestnancu = 5;

operace = [
  <1, {1}, 3>, <2, {1}, 2>, <3, {1}, 2>,
  <4, {2}, 2>, <5, {1, 2}, 1>, <6, {2}, 2>,
  <6, {1}, 4>, <5, {1}, 1>, <4, {1}, 5>,
  <2, {1, 2}, 5>, <2, {1, 2}, 5>, <2, {1, 2}, 5>,
  <4, {2}, 3>, <7, {1, 2}, 3>, <6, {2}, 3>,
];

/***** *.mod soubor * * *****/

using CP;

int pocetStroju = ...;
int pocetRobotu = ...;
int pocetVyrobku = ...;
int pocetZamestnancu = ...;

range Stroje = 1..pocetStroju;
range Roboti = 1..pocetRobotu;
range Vyrobky = 1..pocetVyrobku;

tuple Operace {
  int trvani;
  {int} roboti;
  int lidi;
}

Operace operace[Vyrobky][Stroje] = ...;

dvar interval casy[v in Vyrobky][s in Stroje] size operace[v][s].trvani;
dvar sequence stroj[s in Stroje] in all (v in Vyrobky) casy[v][s];

dvar interval prirazeni[v in Vyrobky][s in Stroje][r in Roboti] optional;
dvar sequence robot[r in Roboti] in all (v in Vyrobky, s in Stroje:
  r in operace[v][s].roboti) prirazeni[v][s][r];

cumulFunction zamestnanci = sum (v in Vyrobky, s in Stroje) pulse(casy[v][s],
  operace[v][s].lidi);

minimize max (v in Vyrobky) endOf(casy[v][pocetStroju]);

subject to {

  forall (s in Stroje)
    noOverlap(stroj[s]);
}

```

```
forall (r in Roboti)
  noOverlap(robot[r]);

zamestnanci <= pocetZamestnancu;

forall (v in Vyrobky, s in 1..pocetStroju-1)
  endBeforeStart(casy[v][s], casy[v][s+1]);

forall (v in Vyrobky, s in Stroje)
  alternative(casy[v][s], all (r in operace[v][s].roboti) prirazeni[v][s][r]);

forall (v in Vyrobky, s in Stroje, r in Roboti: r not in operace[v][s].roboti)
  presenceOf(prirazeni[v][s][r]) == 0;

}
```