57. ročník Matematické olympiády – 2007/2008

Úlohy krajského kola kategorie P

Krajské kolo 57. ročníku MO kategorie P se koná v úterý 15. 1. 2008 v dopoledních hodinách. Na řešení úloh máte 4 hodiny čistého času. V krajském kole MO-P se neřeší žádná praktická úloha, pro zajištění rovných podmínek řešitelů ve všech krajích je použití počítačů při soutěži zakázáno.

Řešení každého příkladu musí obsahovat:

- **Popis řešení,** to znamená slovní popis principu zvoleného algoritmu, *argumenty zdůvodňující jeho správnost* (případně důkaz správnosti algoritmu), diskusi o efektivitě vašeho řešení (časová a paměťová složitost). Slovní popis řešení musí být jasný a srozumitelný i bez nahlédnutí do samotného zápisu algoritmu (do programu). Není vhodné odkazovat se na Vaše řešení předchozích kol, opravovatelé je nemají k dispozici; na autorská řešení se odkazovat můžete.
- **Zápis algoritmu.** V úlohách **P-II-1**, **P-II-2** a **P-II-3** je třeba uvést zápis algoritmu, a to buď ve tvaru zdrojového textu nejdůležitějších částí programu v jazyce Pascal nebo C/C++, nebo v nějakém dostatečně srozumitelném pseudokódu. Nemusíte detailně popisovat jednoduché operace jako vstupy, výstupy, implementaci jednoduchých matematických vztahů, vyhledávání v poli, třídění apod. V řešení úlohy **P-II-4** je nutnou součástí řešení přesný popis příslušných překládacích strojů.

Hodnotí se nejen správnost řešení, ale také kvalita jeho popisu a efektivita zvoleného algoritmu.

Vzorová řešení úloh naleznete krátce po soutěži na webových stránkách olympiády na adrese *http://mo.mff.cuni.cz/*. Na stejném místě bude zveřejněn i seznam úspěšných řešitelů postupujících do ústředního kola. Naleznete zde také popis prostředí, v němž budete v ústředním kole řešit praktické úlohy.

P-II-1 Parkování kočárů

Král Kazimír vdává dceru. U takové slávy (a tolika jídla zdarma) nemůže chybět žádný šlechtic z okolí. A jak tak šlechtici jedou ve svých kočárech na svatbu, vůbec netuší, kolik starostí sluhům krále Kazimíra způsobí při řešení následujícího problému.

Všechny kočáry je třeba zaparkovat, a to ne jen tak nahodile. Kočáry musí stát v řadách za sebou a těchto řad musí být co nejméně, aby královi neponičily trávník.

Dvorní etiketa káže, že až se budou hosté rozjíždět ze svatby domů, musí odjíždět seřazeni podle důležitosti, nejdůležitější host jako poslední. To ovšem ještě není všechno. Kočáry, které jsou zaparkovány v jedné řadě, musí samozřejmě odjíždět v pořadí, ve kterém stojí. Aby se předešlo srážkám kočárů, sluhové je navíc chtějí

zaparkovat tak, aby po skončení svatby odjela vždy celá jedna řada kočárů, až potom začala odjíždět další řada atd.

Soutěžní úloha: Sluhové přesně znají pořadí, v němž budou přijíždět hosté na svatbu, a znají také důležitost každého nich. Když parkují kočár, mohou ho zaparkovat *na začátek nebo na konec* libovolné již existující řady kočárů, případně ho mohou postavit do nové řady. Sluhové musí zaparkovat jednotlivé kočáry v tom pořadí, jak hosté přijíždějí.

Vaším úkolem je určit nejmenší počet řad, který stačí k tomu, aby po správném zaparkování mohly kočáry odjíždět ve stanoveném pořadí.

Formát vstupu: Vstup začíná celým číslem N ($1 \leq N \leq 100\,000$), které určuje počet hostů. Následuje N *různých* kladných celých čísel d_i ($1 \leq d_i \leq 10^9$), která určují důležitost hostů v pořadí, v jakém přijíždějí (větší číslo představuje důležitějšího hosta).

Formát výstupu: Výstup je tvořen jediným řádkem, který obsahuje jedno celé číslo představující nejmenší možný počet řad pro zaparkování kočárů.

Příklady:

Vstup:	*Výstup:*
`10` `10 9 11 12 13 8 14 7 6 100`	`1`

V tomto případě stačí držet se pravidla „kočáry méně důležité než 10 jdou na začátek řady, ostatní na konec.“

Vstup:	*Výstup:*
`6` `12 17 9 23 16 14`	`2`

Jedním možným řešením je zaparkovat kočáry 12 a 17 do různých řad, a pak kočár 9 postavit před kočár 12, kočár 23 za kočár 17, kočár 16 před kočár 17 a nakonec kočár 14 před kočár 16.

Vstup:	*Výstup:*
`12` `1 3 2 5 4 7 6 9 8 11 10 12`	`6`

V jediném optimálním řešení budou po zaparkování řady: (1 2), (3 4), (5 6), (7 8), (9 10) a (11 12).

P-II-2 Dluhopisy

Kleofáš nedávno zdědil po své bohaté tetičce Anastázii hromadu peněz. Nevěděl však co s nimi, a proto se rozhodl investovat je do dluhopisů. Od vás si chce nechat poradit, jak by měl svou investici optimálně spravovat.

Pro jednoduchost budeme předpokládat následující skutečnosti:

- Každý typ dluhopisu má svoji pevnou cenu, stejnou při koupi i prodeji.

- Každý typ dluhopisu má pevně daný roční výnos, který se vyplácí vždy na konci roku.
- Je možné nakoupit libovolné množství každého typu dluhopisu.

Uvažujme například následující situaci: Banka nabízí dva typy dluhopisů: Dluhopisy za 4000 korun s ročním výnosem 400 a dluhopisy za 3000 korun s ročním výnosem 250. Má-li Kleofáš 10 000 korun, nejlepší, co s nimi může udělat, je koupit dva dluhopisy po 3000 a jeden za 4000, čímž získá roční výnos 900 korun. Po dvou letech obdrží Kleofáš dvakrát výnosy a bude mít celkově kapitál 11 800 korun. V tomto okamžiku se mu vyplatí jeden dluhopis za 3000 korun prodat a místo něj si koupit dluhopis za 4000. Po třetím roce bude jeho kapitál roven 12 850 korunám.

Soutěžní úloha: Napište program, který přečte ze vstupu Kleofášův počáteční kapitál, ceny a výnosy nabízených dluhopisů a počet roků, na které chce Kleofáš investovat, a spočítá, kolik nejvíce peněz může Kleofáš mít po uplynutí daného počtu roků.

Formát vstupu: Na prvním řádku vstupu je jedno celé číslo K ($1 \leq K \leq 1\,000\,000$), které udává Kleofášův počáteční kapitál. Na druhém řádku je uveden počet typů nabízených dluhopisů D ($1 \leq D \leq 100$). Na třetím řádku je D dvojic celých čísel c_i a v_i, které představují ceny a výnosy jednotlivých dluhopisů ($0 < c_i \leq 10^9$, $0 \leq v_i \leq c_i/10$, c_i je vždy násobkem $T = 1000$). Na posledním řádku vstupu je uveden počet roků R ($1 \leq R \leq 40$).

Časovou složitost svého algoritmu vyjádřete pomocí K, D, T a R. Navrhněte algoritmus, který pro hodnoty K, D, T a R z výše uvedených rozsahů bude co nejrychlejší.

Formát výstupu: Výstupem programu je jediné číslo, které určuje maximální hodnotu Kleofášova kapitálu po R letech obchodování s dluhopisy. Můžete předpokládat, že se tato hodnota vejde do běžné celočíselné proměnné.

Příklady:

Vstup:	*Výstup:*

`10000`
`2`
`4000 400 3000 250`
`4`
	`14050`

Příklad ze zadání úlohy. Ve čtvrtém roce bude Kleofáš vlastnit 3 dluhopisy po 4 000, čímž vydělá dalších 1 200.

Vstup:	*Výstup:*

`100000`
`3`
`103000 9001 47000 7 83000 100`
`31`
	`112001`

Kleofáš koupí jeden dluhopis za 83 000. Tím za 30 let získá 3 000 korun. Na poslední

rok si konečně může koupit první dluhopis za 103 000. V posledním roce tedy vydělá dalších 9 001.

Vstup:	*Výstup:*

`100000`
`3`
`103000 9001 47000 7 83000 100`
`37`
	`166014`

Pokračování z předchozího příkladu. Po roce 36 Kleofáš dokoupí dluhopis za 47 000, takže v posledním roce získá o 7 korun více.

P-II-3 Piškvorkový turnaj

Silvestr se rozhodl, že uspořádá programátorský turnaj v piškvorkách. Požádal tedy své přátele, aby vytvořili programy pro hraní piškvorek, které se turnaje zúčastní. Silvestrovi přátelé na jeho výzvu rychle zareagovali a tak velký turnaj může začít.

Pravidla turnaje určil Silvestr velmi jednoduše: v každém kole se náhodně vylosují dva programy, které vzájemně sehrají partii, a program, který prohraje, z turnaje nadobro vypadne.

Den před turnajem však Silvestra přemohla zvědavost a zkusil pustit některé dvojice programů proti sobě. Poté si však uvědomil, že se tímto svým počínáním připravil o velkou část překvapení spojenou s turnajem. Protože všechny programy jsou deterministické (tj. nepoužívají náhodnost), dopadne souboj každých dvou programů vždy stejně. Silvestr už tedy ví, že některé programy turnaj nemohou vyhrát. Pro jednoduchost předpokládáme, že to, který program v dvojici začne souboj, výsledek souboje těchto dvou programů neovlivní.

Soutěžní úloha: Pro dané výsledky soubojů některých dvojic programů určete, které programy mohou v turnaji ještě zvítězit.

Formát vstupu: Na prvním řádku vstupu je jedno celé číslo N ($1 \leq N \leq 100\,000$), které určuje počet programů přihlášených do turnaje. Programy jsou očíslovány od 1 do N.

Následuje N řádků, které popisují výsledky zápasů, které Silvestr již zná; i-tý z těchto řádků začíná číslem d_i, které určuje počet programů poražených i-tým programem ve vzájemných soubojích. Tento řádek pak obsahuje d_i čísel programů, které i-tý program porazil. Těchto d_i čísel je seřazeno podle velikosti od nejmenšího po největší.

Označme počet zápasů $d_1 + \cdots + d_N$, které Silvestr den před turnajem spustil, jako M. Můžete předpokládat, že platí $0 \leq M \leq 1\,000\,000$. Také můžete předpokládat korektnost vstupu, tedy speciálně, že pokud program x porazil program y, pak program y neporazil program x.

Formát výstupu: Výstupem programu je jediný řádek, na kterém budou uvedena čísla všech programů, které mohou v turnaji zvítězit.

Příklady:

Vstup:	*Výstup:*

`4`
`2 2 3`
`0`
`1 2`
`1 2`
	`1 3 4`

Do turnaje jsou přihlášeny 4 programy. Program 1 v souboji porazí programy 2 a 3, programy 3 a 4 porazí program 2. Program 2 tedy prohraje souboj s libovolným jiným programem, a tak určitě nemůže turnaj vyhrát.

Ostatní programy v turnaji však zvítězit mohou. Jako příklad si předveďme, jak může v turnaji zvítězit program 3: nejdříve program 3 vyřadí program 2, pak program 4 vyřadí program 1 a nakonec program 3 vyřadí program 4.

Vstup:	*Výstup:*

`5`
`2 2 3`
`0`
`1 2`
`1 2`
`0`
	`1 2 3 4 5`

Tentokrát může zvítězit libovolný z programů 1, 2, 3, 4 a 5. Např. program 2 může zvítězit následovně: nejdříve program 3 porazí program 4, pak program 5 postupně vyřadí programy 3 a 1 a nakonec program 2 vyřadí program 5.

P-II-4 Překládací stroje

Studijní text, který je stejný jako v domácím kole, následuje po zadání soutěžní úlohy.

Soutěžní úloha:

a) *(6 bodů)* Nechť M_1 je množina tvořená všemi řetězci písmen a a b, které obsahují stejný počet písmen a a b. Tedy např. $abbbaa \in M_1$, avšak $aabab \notin M_1$.

Nové množiny můžeme sestrojit následujícími operacemi:

- překladem již sestrojené množiny pomocí překládacího stroje (lze použít jiné překládací stroje při různých překladech),
- sjednocením dvou již sestrojených množin, nebo
- průnikem dvou již sestrojených množin.

Pomocí co nejmenšího počtu výše popsaných operaci sestrojte množinu G, která obsahuje právě všechny řetězce písmen a, b a c, které obsahují stejné množství písmen a jako písmen b a také jako písmen c. Tedy například $aabbcc \in G$, $bac \in G$, ale $abcc \notin G$.

b) *(4 body)* Množina X obsahuje zápisy kladných celých čísel v desítkové soustavě, v nichž se vyskytuje stejný počet číslic 1 a 2. Tedy například $1122 \in X$, $21231 \in X$, $47 \in X$, ale $112 \notin X$ a $031221 \notin X$ (zápis kladného čísla nemůže začínat číslicí 0).

Množina Y obsahuje zápisy v desítkové soustavě těch kladných čísel, která jsou dělitelná 7. Tedy například $140 \in Y$, $7707 \in Y$, ale $47 \notin Y$ a $07 \notin Y$.

Sestrojte překládací stroj, který přeloží množinu X na množinu Y, anebo dokažte, že takový překládací stroj neexistuje.

Studijní text:

Překládací stroj přijímá na vstupu řetězec znaků. Tento řetězec postupně čte a podle předem zvolené soustavy pravidel (tedy podle svého programu) občas nějaké znaky zapíše na výstup. Když stroj zpracuje celý vstupní řetězec a úspěšně ukončí svůj výpočet, vezmeme řetězec znaků zapsaný na výstup a nazveme ho *překladem* vstupního řetězce.

Výpočet stroje nemusí být jednoznačně určen. Jinými slovy, soustava pravidel může někdy stroji umožnit, aby se rozhodl o dalším postupu výpočtu. V takovém případě se může stát, že některý řetězec bude mít více různých překladů.

Naopak, může se stát, že v určité situací se podle daných pravidel nebude moci v překladu pokračovat vůbec. V takovém případě se může stát, že některý řetězec nebude mít vůbec žádný překlad.

Formálnější definice překládacího stroje

Každý překládací stroj pracuje nad nějakou předem zvolenou konečnou množinou znaků. Tuto množinu znaků budeme nazývat *abeceda* a značit Σ. V soutěžních úlohách bude vždy Σ známa ze zadání úlohy. Abeceda nebude nikdy obsahovat znak \$, ten budeme používat k označení konce vstupního řetězce.

Stroj si může během překladu řetězce pamatovat informaci konečné velikosti. Formálně tuto skutečnost definujeme tak, že stroj se v každém okamžiku překladu nachází v jednom z *konečně mnoha* stavů. Nutnou součástí programu překládacího stroje je tedy nějaká konečná *množina stavů*, v nichž se stroj může nacházet. Tuto množinu označíme K. Kromě samotné množiny stavů je také třeba uvést, ve kterém stavu se stroj nachází na začátku každého překladu. Tento stav nazveme *počáteční stav*.

Program stroje se skládá z konečného počtu překladových pravidel. Každé pravidlo má tvar čtveřice (p, u, v, q), kde $p, q \in K$ jsou nějaké dva stavy a u, v jsou nějaké dva řetězce znaků z abecedy Σ.

Stavy p a q mohou být i stejné. Řetězec u může být tvořen jediným znakem \$. Řetězce u a v mohou být i stejné. Některý z těchto řetězců může být případně prázdný. Aby se program lépe četl, budeme místo prázdného řetězce psát symbol ε.

Překladové pravidlo má následující význam: „Když je stroj právě ve stavu p a dosud nepřečtená část vstupu začíná řetězcem u, může stroj tento řetězec ze vstupu přečíst, na výstup zapsat řetězec v a změnit svůj stav na q.“ Všimněte si, že pravidlo

tvaru (p, ε, v, q) můžeme použít vždy, když se stroj nachází ve stavu p, bez ohledu na to, jaké znaky ještě zůstávají na vstupu.

Ještě potřebujeme stanovit, kdy překlad úspěšně skončil. V první řadě budeme požadovat, aby překládací stroj přečetl celý vstupný řetězec. Kromě toho umožníme stroji „odpovědět“, zda se mu překlad podařil nebo ne. To zařídíme tak, že některé stavy stroje označíme jako *koncové stavy*. Množinu všech koncových stavů budeme značit F.

Formální definice překládacího stroje

Překládací stroj je uspořádaná pětice (K, Σ, P, q_0, F), kde Σ a K jsou *konečné* množiny, $q_0 \in K$, $F \subseteq K$ a P je *konečná* množina překladových pravidel popsaných výše. Přesněji, nechť Σ^* je množina všech řetězců tvořených znaky ze Σ, potom P je *konečná* podmnožina množiny $K \times (\Sigma^* \cup \{\$\}) \times \Sigma^* \times K$.

(Pro každé $q \in K$ budeme množinu pravidel, jejichž první složkou je q, nazývat „překladová pravidla ze stavu q“.)

Chceme-li definovat konkrétní překládací stroj, musíme uvést všech pět výše uvedených objektů.

Když už máme definován konkrétní stroj $A = (K, \Sigma, P, q_0, F)$, můžeme určit, jak tento stroj překládá konkrétní řetězec. Uvedeme nejprve formální definici a potom ji slovně vysvětlíme.

Množina *platných překladů* řetězce u překládacím strojem A je:

$$A(u) = \Big\{ \quad v \quad \Big| \quad \exists n \geq 0 \; \exists (p_1, u_1, v_1, r_1), \ldots, (p_n, u_n, v_n, r_n) \in P : \\ (\forall i \in \{1, \ldots, n-1\} : r_i = p_{i+1}) \; \wedge \; p_1 = q_0 \; \wedge \; r_n \in F \; \wedge \\ \wedge \; \exists k \geq 0 : u_1 u_2 \ldots u_n = u \underbrace{\$ \ldots \$}_{k} \wedge v_1 v_2 \ldots v_n = v \quad \Big\}.$$

Definice stanoví, kdy je řetězec v překladem řetězce u. Vysvětlíme si slovně význam jednotlivých řádků definice:

- První řádek říká, že aby se dalo u přeložit na v, musí existovat nějaká posloupnost překladových pravidel, kterou při tomto překladu použijeme. Další dva řádky popisují, jak tato posloupnost musí vypadat.
- Druhý řádek zabezpečuje, aby stavy v použitých pravidlech byly správné: První pravidlo musí být pravidlem z počátečního stavu, každé další pravidlo musí být pravidlem z toho stavu, do něhož se stroj dostal použitím předcházejícího pravidla.

  Navíc stav, v němž se bude stroj nacházet po skončení výpočtu, musí být koncový.
- Poslední řádek popisuje řetězce, které stroj při použití dotyčných překladových pravidel čte a zapisuje.

  Řetězec, který při použití těchto pravidel stroj přečte ze vstupu, musí být skutečně zadaným řetězcem u, případné může být zprava doplněn vhodným počtem znaků \$.

  Řetězec, který stroj zapíše na výstup, musí být přesně řetězcem v.

K čemu budeme používat překládací stroje?

Překládací stroje nám budou sloužit k získání překladu jedné množiny řetězců na jinou množinu řetězců. Jestliže A je překládací stroj a $M \subseteq \Sigma^*$ nějaká množina řetězců, potom překlad množiny M strojem A je množina

$$A(M) = \bigcup_{u \in M} A(u).$$

Jinými slovy, výslednou množinu $A(M)$ sestrojíme tak, že vezmeme všechny řetězce z M a pro každý z nich přidáme do $A(M)$ všechny jeho platné překlady.

Příklad 1

Mějme abecedu $\Sigma = \{0, \ldots, 9\}$. Nechť M je množina všech řetězců, které představují zápisy kladných celých čísel v desítkové soustavě. Sestrojíme překládací stroj A, pro který bude platit, že překladem této množiny M bude množina zápisů všech kladných celých čísel, která jsou dělitelná třemi.

Řešení

Nejjednodušší bude prostě vybrat z M ta čísla, která jsou dělitelná třemi. Náš překládací stroj bude kopírovat cifry ze vstupu na výstup, přičemž si bude pomocí stavu pamatovat, jaký zbytek po dělení třemi dává dosud přečtené (a zapsané) číslo. Nachází-li se po dočtení vstupu ve stavu odpovídajícím zbytku 0, přejde do koncového stavu.

Formálně A bude pětice $(K, \Sigma, P, 0, F)$, kde $K = \{0, 1, 2, end\}$, $F = \{end\}$ a překladová pravidla vypadají následovně:

$$P = \Big\{(x, y, y, z) \;\Big|\; x \in \{0,1,2\} \wedge y \in \Sigma \wedge z = (10x + y) \bmod 3\Big\} \cup \Big\{(0, \$, \varepsilon, end)\Big\}.$$

Příklad 2

Mějme abecedu $\Sigma = \{a, e, i, \bullet, \blacksquare\}$. Sestrojíme překládací stroj B, pro který bude platit, že překladem libovolné množiny M, která obsahuje pouze řetězce z písmen a, e a i, bude množina stejných řetězců zapsaných v morseovce (bez oddělovačů mezi znaky). Zápisy našich písmen v morseovce vypadají následovně: a je $\bullet\blacksquare$, e je $\bullet$ a i je $\bullet\bullet$.

Například množinu $M = \{ae, eea, ia\}$ by náš stroj měl přeložit na množinu $\{\bullet\blacksquare\bullet, \bullet\bullet\bullet\blacksquare\}$. (Všimněte si, že řetězce eea a ia mají v morseovce bez oddělovačů stejný zápis.)

Řešení

Překládací stroj B bude číst vstupní řetězec po znacích a vždy zapíše na výstup kód přečteného znaku.

Formálně B bude pětice $(K, \Sigma, P, \heartsuit, F)$, kde $K = \{\heartsuit\}$, $F = \{\heartsuit\}$ a překladová pravidla vypadají takto:

$$P = \{(\heartsuit, a, \bullet\blacksquare, \heartsuit),\ (\heartsuit, e, \bullet, \heartsuit),\ (\heartsuit, i, \bullet\bullet, \heartsuit)\}.$$

Všimněte si, že nepotřebujeme nijak zvlášť kontrolovat, zda jsme na konci vstupu. Během celého překladu je totiž stroj v koncovém stavu, takže jakmile přečte poslední znak ze vstupu, bude vytvořený překlad platný.

Příklad 3

Mějme abecedu $\Sigma = \{a, e, i, \bullet, \blacksquare\}$. Sestrojíme překládací stroj C, pro který bude platit, že překladem libovolné množiny M, která obsahuje pouze řetězce tvořené znaky $\bullet$ a $\blacksquare$, bude množina *všech* řetězců z písmen a, e a i, jejichž zápisy v morseovce (bez oddělovačů mezi znaky) jsou obsaženy v množině M. Například množinu $M = \{\bullet\blacksquare\bullet, \bullet\bullet\bullet\blacksquare\}$ by náš stroj měl přeložit na $\{ae, eea, ia\}$.

Řešení

Našemu překládacímu stroji dáme možnost rozhodnout se v každém okamžiku překladu, že bude číst kód nějakého písmena a zapíše na výstup toto písmeno. Potom každé možnosti, jak lze rozdělit vstupní řetězec na kódy písmen, bude odpovídat jeden platný překlad.

Formálně C bude pětice $(K, \Sigma, P, \diamondsuit, F)$, kde $K = \{\diamondsuit\}$, $F = \{\diamondsuit\}$ a překladová pravidla vypadají následovně:

$$P = \{(\diamondsuit, \bullet\blacksquare, a, \diamondsuit),\ (\diamondsuit, \bullet, e, \diamondsuit),\ (\diamondsuit, \bullet\bullet, i, \diamondsuit)\}.$$

Ukážeme si, jak mohl probíhat překlad řetězců z výše uvedené množiny M. Existují tyto tři možnosti:

$(\diamondsuit, \bullet\blacksquare, a, \diamondsuit), (\diamondsuit, \bullet, e, \diamondsuit)$
$(\diamondsuit, \bullet\bullet, i, \diamondsuit), (\diamondsuit, \bullet\blacksquare, a, \diamondsuit)$
$(\diamondsuit, \bullet, e, \diamondsuit), (\diamondsuit, \bullet, e, \diamondsuit), (\diamondsuit, \bullet\blacksquare, a, \diamondsuit)$

Kdybychom zkusili pro libovolný vstupní řetězec z M použít překladová pravidla v jiném pořadí – např. pro vstup $\bullet\bullet\bullet\blacksquare$ použít třikrát pravidlo $(\diamondsuit, \bullet, e, \diamondsuit)$ – nepodaří se nám dočíst vstupní řetězec až do konce.

Příklad 4

Mějme abecedu $\Sigma = \{a, b, c\}$. Nechť množina X obsahuje právě všechny řetězce, v nichž je obsažen stejný počet znaků a a b. Tedy například $abbccac \in X$, ale $cbaa \notin X$.

Nechť množina Y obsahuje právě všechny řetězce, které neobsahují žádné a, neobsahují podřetězec bc, a písmen c je dvakrát více než písmen b. Tedy například $ccccbb \in Y$, ale $cccbcb \notin Y$ a $acacba \notin Y$.

Sestrojíme překládací stroj D, který přeloží X na Y.

Řešení

Budeme překládat jenom některé vhodné řetězce z množiny X. Budou to ty řetězce, které neobsahují žádné c a všechna a v nich předcházejí všem znakům b. Takovýto řetězec přeložíme tak, že nejprve každé a přepíšeme na cc, a potom zkopírujeme na výstup všechna b. Tedy například překladem slova $aabb$ bude slovo $ccccbb$.

Formálně D bude pětice $(K, \Sigma, P, \textbf{čti-a}, F)$, kde $K = \{\textbf{čti-a}, \textbf{čti-b}\}$, $F = \{\textbf{čti-b}\}$ a překladová pravidla vypadají takto:

$$P = \{(\textbf{čti-a}, a, cc, \textbf{čti-a}),\ (\textbf{čti-a}, \varepsilon, \varepsilon, \textbf{čti-b}),\ (\textbf{čti-b}, b, b, \textbf{čti-b})\}.$$

Proč tento překládací stroj funguje? Když vstupní řetězec obsahuje nějaké písmeno c, při jeho překládání se u prvního výskytu c náš stroj zastaví. Proto takové řetězce nemají žádný platný překlad. Podobně nemají platný překlad řetězce, v nichž není dodrženo pořadí písmen a a b. Po přečtení nějaké posloupnosti písmen a přejde stroj pomocí druhého pravidla do stavu **čti-b**, a pokud se poté ještě objeví na vstupu a, stroj se zastaví.

Platné překlady tedy existují skutečně pouze pro slova výše popsaného tvaru. Je zjevné, že překladem každého z nich získáme nějaký řetězec z Y, takže $D(X) \subseteq Y$. Naopak, vybereme-li si libovolné slovo z Y, snadno najdeme slovo z X, které se na něj přeloží. Proto také $Y \subseteq D(X)$, takže $Y = D(X)$.