Kapitola 4

Třídy složitosti

4.1 Rozhodovací úlohy

4.1.1 Teorie složitosti pracuje zejména s tzv *rozhodovacími* úlohami. Rozhodovací úlohy jsou takové úlohy, jejichž „řešením“ je buď odpověď „ANO“ nebo odpověď „NE“.

4.1.2 Příklad. *SAT – splňování Booleovských formulí*: Je dána výroková formule φ v CNF. Rozhodněte, zda je φ splnitelná.

Na danou formuli φ je tedy odpověď (tj. řešení) buď „ANO“ nebo „NE“. Všimněte si, že v tomto případě se neptáme po ohodnocení, ve kterém je formule pravdivá – zajímá nás pouze fakt, zda je splnitelná.

4.1.3 Řada praktických úloh není podobného druhu jako uvedený příklad. Často se jedná o tzv. optimalizační úlohy, tj. úlohy, kde mezi přípustnými řešeními hledáme přípustné řešení v jistém smyslu optimální. Obvykle to bývá tak, že je dána účelová funkce, která každému přípustnému řešení přiřadí číselnou hodnotu, a úkolem je najít přípustné řešení, pro které je hodnota účelové funkce optimální, tj. buď největší nebo naopak nejmenší. V dalším textu se s řadou těchto úloh setkáme. Takovými úlohami jsou například úlohy zmíněné v předminulé přednášce **??** nalezení minimální kostry v ohodnoceném neorientovaném grafu i nalezení nejkratších cest v daném ohodnoceném orientovaném grafu.

Nyní uvedeme další příklad.

4.1.4 Problém obchodního cestujícího – TSP. Jsou dána města 1,2, $\ldots, n$. Pro každou dvojici měst i, j je navíc dáno kladné číslo $d(i,j)$ (tak zvaná vzdálenost měst i,j). *Trasa* je dána permutací π množiny $\{1,2,\ldots,n\}$ do sebe. Délka trasy T odpovídající permutaci π je

$$d(T) = \sum_{i=1}^{n-1} d(\pi(i), \pi(i+1)) + d(\pi(n), \pi(1)).$$

Neformálně, trasa je pořadí měst, ve kterém má obchodní cestující města projít, a to tak, aby každé město navštívil přesně jednou a vrátil se do toho města, ze kterého vyšel. Cena trasy je pak součtem všech vzdáleností, které při své cestě urazil.

4.1.5 Rozhodovací verze.

- Minimální kostra: Je dán neorientovaný graf $G = (V, E)$, ohodnocení $c\colon E \to \mathbb{N}$ a dále číslo K. Existuje minimální kostra, jejíž cena je nejvýše K?
- Je dána matice délek $\mathbf{A} = (a(i,j))$, výchozí vrchol r, cílový vrchol c a číslo K. Existuje cesta z vrcholu r do vrcholu c délky nejvýše K?
- Kromě čísel $d(i,j)$ z 4.1.4 je dáno číslo K. Existuje trasa π délky nejvýše K?

4.1.6 Vyhodnocovací verze.

- Minimální kostra: Je dán neorientovaný graf $G = (V, E)$ a $c: E \to \mathbb{N}$. Najděte cenu minimální kostry ohodnoceného grafu.
- Je dána matice délek $\mathbf{A} = (a(i, j))$, výchozí vrchol r a cílový vrchol c. Najděte délku nejkratší cesty z vrcholu r do vrcholu c.
- Jsou dána čísla $d(i, j)$ a 4.1.4. Najděte cenu optimální trasy, tj. trasy s nejmenší možnou délkou.

4.1.7 Optimalizační verze.

- Minimální kostra: Je dán neorientovaný graf $G = (V, E)$ a $c: E \to \mathbb{N}$. Najděte minimální kostru ohodnoceného grafu.
- Je dána matice délek $\mathbf{A} = (a(i, j))$, výchozí vrchol r, cílový vrchol c. Najděte nejkratší cestu z vrcholu r do vrcholu c.
- Jsou dána čísla $d(i, j)$ a 4.1.4. Najděte optimální trasu, tj. trasu s nejmenší možnou délkou.

4.1.8 Dá se dokázat, že když je kterákoli verze dané úlohy polynomiálně řešitelná, jsou polynomiálně řešitelné všechny tři verze. Ukážeme si to na příkladu obchodního cestujícího.

Předpokládejme, že existuje algoritmus $\mathcal{A}$, který rozhodne, zda pro libovolnou danou instanci TSP a dané K existuje trasa délky nejvýše K.

Uvažujme libovolnou instanci TSP. Označme d největší délku $d(i, j)$; dále označme $A := n \cdot d$, kde n je počet měst. Zavoláme algoritmus $\mathcal{A}$ pro $K := \lceil \frac{A}{2} \rceil$. Jestliže algoritmus $\mathcal{A}$ dá pro K odpověď „ano“, tak jako K volíme střed mezi 0 a K, jestliže algoritmus $\mathcal{A}$ dá pro K odpověď „ne“, tak jako K volíme střed mezi K a $2K$. Takto postupujeme tak dlouho, dokud nemá interval délku nula. Nyní je K hodnota optimální trasy, tj. řešení vyhodnocovací verze úlohy TSP. Uvědomte si, že vzhledem k tomu, že nás zajímají pouze **celočíselná** K, stane se to po maximálně $\lg(A) = \lg(n \cdot d)$ což je $\mathcal{O}(\lg(n))$ opakování.

Ukázali jsme, že po $\mathcal{O}(\lg(n))$ voláních algoritmu $\mathcal{A}$ známe hodnotu optimální trasy, označme ji D_{opt}.

Uvažujme úplný graf G na množině $V = \{1, \ldots, n\}$ ohodnocený délkami $d(i, j)$. Nyní „zorientujeme hrany“ a to tak, že hraně $\{i, j\}$, kde $i < j$, přiřadíme uspořádanou dvojici (i, j),a tyto dvojice uspořádáme lexikograficky. Probíráme dvojice (i, j) v tomto pořadí a pro každou dvojici vytvoříme novou instanci $I_{i,j}$ TSP tak, že z v předchozí instanci změníme pouze délku $d(i, j)$ a to $d(i, j) := n \cdot d$. Zavoláme algoritmus $\mathcal{A}$ na instanci $I_{i,j}$ a $K = D_{opt}$. Jestliže algoritmus $\mathcal{A}$ odpoví „ano“, hraně (i, j) ponecháme tuto novou délku. Jestliže algoritmus $\mathcal{A}$ odpoví „ne“, hraně (i, j) vrátíme původní délku a přejdeme na další dvojici v uspořádání. V okamžiku, kdy máme pouze n hran s původní délkou, těchto n hran tvoří (některou) optimální trasu TSP.

Uvědomte si, že v druhé části jsme použili pouze $\mathcal{O}(n^2)$ volání algoritmu $\mathcal{A}$. Odtud dostáváme: Kdyby existoval polynomiální algoritmus na řešení rozhodovací verze TSP, pak existuje i polynomiální algoritmus na řešení optimalizační verze TSP.

4.2 Třídy $\mathcal{P}$ a $\mathcal{NP}$

4.2.1 Instance úlohy jako slovo nad vhodnou abecedou. Instance libovolné rozhodovací úlohy můžeme zakódovat jako slova nad vhodnou abecedou. Ukažme si to na příkladě problému SAT a úlohy nalezení nejkratší cesty v daném orientovaném ohodnoceném grafu.

- Pro problém SAT (splňování booleovských formulí) je instancí libovolná formule φ v konjunktivním normálním tvaru (CNF). Označme jednotlivé logické proměnné formule φ jako $x_1, x_2, \ldots, x_n$. Pak φ můžeme zakódovat jako slovo nad abecedou $\{x, 0, 1, (,), \vee, \wedge, \neg\}$ takto:

proměnná x_i se zakóduje slovem xw, kde w je binární zápis čísla i, ostatní symboly jsou zachovány.

Například formuli $\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_4)$ odpovídá slovo

$$(x1 \ \vee \ \neg\, x10 \vee x11) \ \wedge \ (\neg\, x1 \ \vee \ x100).$$

- U úlohy nalezení nejkratší cesty z vrcholu r do vrcholu c můžeme postupovat takto: Instanci tvoří matice délek daného orientovaného ohodnoceného grafu, dvojice vrcholů r a c a číslo k. Matici není těžké zakódovat jako slovo, za ní pak následuje pořadové číslo vrcholu r, pořadové číslo vrcholu c a číslo k, vše oddělené např. symbolem #.

4.2.2 Úloha jako jazyk nad abecedou. Protože řešením rozhodovací úlohy je buď „ANO“ nebo „NE“, rozdělíme instance úlohy na tzv. „ANO-instance“ a „NE-instance“. *Jazyk* úlohy $\mathcal{U}$, značíme jej $L_{\mathcal{U}}$, se skládá ze všech slov odpovídajících ANO-instancím úlohy $\mathcal{U}$.

Uvědomte si, že některá slova nad abecedou Σ nemusí odpovídat žádné instanci dané úlohy. Tato slova chápeme jako „NE-instance“. Můžeme proto říci, že množina všech NE instancí tvoří doplněk jazyka $L_{\mathcal{U}}$, tj. je to $\Sigma^\star \setminus L_{\mathcal{U}}$.

4.2.3 Třída $\mathcal{P}$. Řekneme, že rozhodovací úloha $\mathcal{U}$ leží ve třídě $\mathcal{P}$, jestliže existuje deterministický Turingův stroj, který rozhodne jazyk $L_{\mathcal{U}}$ a pracuje v polynomiálním čase; tj. funkce $T(n)$ je $\mathcal{O}(p(n))$ pro nějaký polynom $p(n)$.

4.2.4 Příklady.

- Minimální kostra v grafu. Je dán neorientovaný graf G s ohodnocením hran c. Je dáno číslo k. Existuje kostra grafu ceny menší nebo rovno k?

- Nejkratší cesty v acyklickém grafu. Je dán acyklický graf s ohodnocením hran a. Jsou dány vrcholy r a c. Je dáno číslo k. Existuje orientovaná cesta z vrcholu r do vrcholu c délky menší nebo rovno k?

- Toky v sítích. Je dána síť s horním omezením c, dolním omezením l, se zdrojem z a spotřebičem s. Dále je dáno číslo k. Existuje přípustný tok od z do s velikosti alespoň k?

- Minimální řez. Je dána síť s horním omezením c, dolním omezením l. Dále je dáno číslo k. Existuje řez, který má kapacitu menší nebo rovnu k?

Uvedli jsme všechny úlohy v rozhodovací verzi. Velmi často se mluví i o jejich optimalizačních verzích jako o polynomiálně řešitelných úlohách.

4.2.5 Třída $\mathcal{NP}$. Řekneme, že rozhodovací úloha $\mathcal{U}$ leží ve třídě $\mathcal{NP}$, jestliže existuje nedeterministický Turingův stroj, který rozhodne jazyk $L_{\mathcal{U}}$ a pracuje v polynomiálním čase.

4.2.6 Poznámka. V definici 4.2.3 jsme místo existence Turingova stroje mohli požadovat existenci programu P pro RAM, který řeší $\mathcal{U}$ v polynomiálním čase. Abychom přiblížili, které jazyky (rozhodovací úlohy) leží ve třídě $\mathcal{NP}$, zavedeme pojem nedeterministického algoritmu jako analogii RAM.

4.2.7 Nedeterministický algoritmus pracuje ve dvou fázích,

1. Algoritmus náhodně vygeneruje řetězec s (odpovídá řešení dané úlohy).

2. Deterministický algoritmus (Turingův stroj, program pro RAM) na základě vstupu a řetězce s dá odpověď ANO nebo NEVIM. (Deterministicky a polynomiálně ověří řešení.)

Řekneme, že nedeterministický algoritmus řeší úlohu $\mathcal{U}$, jestliže

1. Pro každou ANO instanci úlohy $\mathcal{U}$ existuje řetězec s, na jehož základě algoritmus dá odpověď ANO.
2. Pro žádnou NE instanci úlohy $\mathcal{U}$ neexistuje řetězec s, na jehož základě algoritmus dá odpověď ANO.

Řekneme, že nedeterministický algoritmus *pracuje v čase* $\mathcal{O}(T(n))$, jestliže každý průchod oběma fázemi 1 a 2 pro instanci velikosti n potřebuje $\mathcal{O}(T(n))$ kroků.

4.2.8 Poznámka. Fakt, že nedeterministický algoritmus pracuje v polynomiálním čase, znamená, že každá z fází vyžaduje polynomiální čas a tudíž i řetězec s musí mít polynomiální délku (vzhledem k velikosti instance).

V definici 4.2.5 jsme místo existence nedeterministického Turingova stroje mohli požadovat existenci nedeterministického algoritmu, který řeší úlohu $\mathcal{U}$ v polynomiálním čase.

4.2.9 Příklady $\mathcal{NP}$ úloh.

- Kliky v grafu. Je dán neorientovaný graf G a číslo k. Existuje klika v grafu G o alespoň k vrcholech?
- Nejkratší cesty v obecném grafu. Je dán orientovaný graf s ohodnocením hran a. Jsou dány vrcholy r a v. Je dáno číslo k. Existuje orientovaná cesta z vrcholu r do vrcholu v délky menší nebo rovno k?
- k-barevnost. Je dán neorientovaný graf G. Je graf G k-barevný?
- Problém batohu. Je dáno n předmětů $1, 2, \ldots, n$. Každý předmět i má cenu c_i a váhu w_i. Dále jsou dána čísla A a B. Je možné vybrat předměty tak, aby celková váha nepřevýšila A a celková cena byla alespoň B? Přesněji, existuje podmnožina předmětů $I \subseteq \{1, 2, \ldots, n\}$ taková, že
$$\sum_{i\in I} w_i \le A \quad \text{a} \quad \sum_{i\in I} c_i \ge B?$$

4.3 Třída $\mathcal{NPC}$

4.3.1 Redukce a polynomiální redukce úloh. Jsou dány dvě rozhodovací úlohy $\mathcal{U}$ a $\mathcal{V}$. Řekneme, že úloha $\mathcal{U}$ se *redukuje* na úlohu $\mathcal{V}$, jestliže existuje algoritmus (program pro RAM, Turingův stroj) M, který pro každou instanci I úlohy $\mathcal{U}$ zkonstruuje instanci I' úlohy $\mathcal{V}$ a to tak, že

$$I \text{ je ANO-instance } \mathcal{U} \text{ právě tehdy, když } I' \text{ je ANO-instance } \mathcal{V}.$$

Fakt, že úloha $\mathcal{U}$ se redukuje na úlohy $\mathcal{V}$ značíme

$$\mathcal{U} \lhd \mathcal{V}.$$

Jestliže navíc algoritmus M pracuje v polynomiálním čase, říkáme, že $\mathcal{U}$ se *polynomiálně* redukuje na $\mathcal{V}$ a značíme

$$\mathcal{U} \lhd_p \mathcal{V}.$$

Fakt, že se úloha $\mathcal{U}$ redukuje na úlohu $\mathcal{V}$ zhruba řečeno znamená, že $\mathcal{U}$ není obtížnější než $\mathcal{V}$.

4.3.2 Tvrzení. Jsou dány tři rozhodovací úlohy $\mathcal{U}$, $\mathcal{V}$ a $\mathcal{W}$. Jestliže platí

$$\mathcal{U} \lhd_p \mathcal{V} \text{ a } \mathcal{V} \lhd_p \mathcal{W}, \quad \text{pak} \quad \mathcal{U} \lhd_p \mathcal{W}.$$

4.3.3 $\mathcal{NP}$ úplné úlohy. Řekneme, že rozhodovací úloha $\mathcal{U}$ je $\mathcal{NP}$ *úplná*, jestliže

1. $\mathcal{U}$ je ve třídě $\mathcal{NP}$;
2. každá $\mathcal{NP}$ úloha se polynomiálně redukuje na $\mathcal{U}$.

Třída všech $\mathcal{NP}$ úplných úloh se značí $\mathcal{NPC}$.

Zhruba řečeno, $\mathcal{NP}$ úplné úlohy jsou ty „nejtěžší“ mezi všemi $\mathcal{NP}$ úlohami.

4.3.4 Tvrzení. Jsou dány dvě $\mathcal{NP}$ úlohy $\mathcal{U}$ a $\mathcal{V}$, pro které platí $\mathcal{U} \lhd_p \mathcal{V}$. Pak

1. jestliže $\mathcal{V}$ je ve třídě $\mathcal{P}$, pak také $\mathcal{U}$ je ve třídě $\mathcal{P}$;
2. jestliže $\mathcal{U}$ je $\mathcal{NP}$ úplná úloha, pak také $\mathcal{V}$ je $\mathcal{NP}$ úplná úloha.

4.3.5 Tvrzení. Kdyby některá $\mathcal{NP}$ úplná úloha patřila do třídy $\mathcal{P}$ (tj. byla by polynomiálně řešitelná), pak $\mathcal{P} = \mathcal{NP}$. Jinými slovy, každá $\mathcal{NP}$ úloha by byla polynomiálně řešitelná.

4.3.6 $\mathcal{NP}$ obtížné úlohy. Jestliže o některé úloze $\mathcal{U}$ pouze víme, že se na ní polynomiálně redukuje některá $\mathcal{NP}$ úplná úloha, pak říkáme, že $\mathcal{U}$ je $\mathcal{NP}$ těžká, nebo též $\mathcal{NP}$ obtížná. Poznamenejme, že to vlastně znamená, že $\mathcal{U}$ je alespoň tak těžká jako všechny $\mathcal{NP}$ úlohy.

4.3.7 Cookova věta. Úloha SAT, splňování formulí v konjunktivním normálním tvaru, je $\mathcal{NP}$ úplná úloha.

4.3.8 Myšlenka důkazu. Není těžké se přesvědčit, že úloha SAT je ve třídě $\mathcal{NP}$. První fáze nedeterministického algoritmu vygeneruje ohodnocení logických proměnných a na základě tohoto ohodnocení jsme schopni v polynomiálním čase ověřit, zda je v tomto ohodnocení formule pravdivá nebo ne.

Druhá část důkazu spočívá v popisu práce Turingova stroje formulí výrokové logiky. Načrtneme základní myšlenku tohoto popisu.

Je dán nedeterministický Turingův stroj M s množinou stavů Q, vstupní abecedou Σ, páskovou abecedou Γ, přechodovou funkcí δ, počátečním stavem q_0 a koncovým stavem q_f. Předpokládejme, že M přijímá slovo w a potřebuje přitom $p(n)$ kroků.

Zavedeme logické proměnné:

$h_{i,j}$, $i = 0, 1, \ldots, p(n)$, $j = 1, 2, \ldots, p(n)$;

fakt, že hodnota proměnné $h_{i,j}$ je rovna 1 znamená, že hlava Turingova stroje v čase i čte j-té pole pásky.

s_i^q, $i = 0, 1, \ldots, p(n)$, $q \in Q$;

fakt, že hodnota proměnné s_i^q je rovna 1 znamená, že Turingův stroj v čase i je ve stavu q.

$t_{i,j}^A$, $i = 0, 1, \ldots, p(n)$, $j = 1, 2, \ldots, p(n)$, $A \in \Gamma$;

fakt, že hodnota proměnné $t_{i,j}^A$ rovna 1 znamená, že v čase i v j-tém poli pásky je páskový symbol A.

Nyní je třeba formulemi popsat následující fakta:

1. V každém okamžiku je Turingův stroj v právě jednom stavu.
2. V každém okamžiku čte hlava Turingova stroje právě jedno pole vstupní pásky.
3. V každém okamžiku je na každém poli pásky Turingova stroje právě jeden páskový symbol.
4. Na začátku práce (tj. v čase 0) je Turingův stroj ve stavu q_0, hlava čte první pole pásky a na pásce je na prvních n polích vstupní slovo, ostatní pole pásky obsahují B.

5. Krok Turingova stroje je určen přechodovou funkcí, tj. stav stroje, obsah čteného pole a poloha hlavy v čase $i+1$ je dána přechodovou funkcí.

6. V polích pásky, které v čase i hlava nečte, je obsah v čase $i+1$ stejný jako v i.

7. Na konci práce Turingova stroje, tj. v čase $p(n)$, je stroj ve stavu q_f.

Ukážeme jak utvořit formule pro jednotlivé body

Bod 1. V okamžiku i je Turingův stroj v aspoň jednom stavu:

$$\bigvee_{q\in Q} s_i^q.$$

V okamžiku i Turingův stroj není ve dvou různých stavech:

$$\bigwedge_{q\neq q'} (\neg s_i^q \vee \neg s_i^{q'}).$$

Nyní fakt, že Turingův stroj je v okamžiku i v právě jednom stavu je konjunkce obou výše uvedených formulí:

$$(\bigvee_{q\in Q} s_i^q) \wedge \bigwedge_{q\neq q'} (\neg s_i^q \vee \neg s_i^{q'}).$$

Bod 2. V okamžiku i je v j-tém poli pásky Turingova stroje aspoň jeden páskový symbol:

$$\bigvee_{A\in\Gamma} t_{i,j}^A.$$

V okamžiku i v j-tém poli pásky Turingova stroje nejsou dva různé páskové symboly:

$$\bigwedge_{A\neq A'} (\neg t_{i,j}^A \vee \neg t_{i,j}^{A'}).$$

Nyní fakt, že Turingův stroj má v okamžiku i v j-tém poli právě jeden páskový symbol je konjunkce obou výše uvedených formulí:

$$(\bigvee_{A\in\Gamma} t_{i,j}^A) \wedge \bigwedge_{A\neq A'} (\neg t_{i,j}^A \vee \neg t_{i,j}^{A'}).$$

Bod 3. V okamžiku i čte hlava Turingova stroje aspoň jedno pole pásky:

$$\bigvee_{1\leq j\leq p(n)} h_{i,j}.$$

V okamžiku i nečte hlava Turingova stroje dvě různá pole:

$$\bigwedge_{j\neq k} (\neg h_{i,j} \vee \neg h_{i,k}).$$

Nyní fakt, že hlava Turingova stroje v okamžiku i čte přesně jedno pole pásky je konjunkce obou výše uvedených formulí:

$$(\bigvee_{1\leq j\leq p(n)} h_{i,j}) \wedge \bigwedge_{j\neq k} (\neg h_{i,j} \vee \neg h_{i,k}).$$

Bod 4. Na začátku práce (tj. v čase 0) je Turingův stroj ve stavu q_0, hlava čte první pole pásky a na pásce je na prvních n polích vstupní slovo $a_1 a_2 \dots a_n$, ostatní pole obsahují B.

$$s_0^{q_0} \wedge h_{0,1} \wedge t_{0,1}^{a_1} \wedge \dots \wedge t_{0,n}^{a_n} \wedge t_{0,n+1}^B \wedge \dots \wedge t_{0,p(n)}^B.$$

Bod 5. Jestliže Turingův stroj je v čase i ve stavu q, hlava je na j-tém poli pásky, hlava čte páskový symbol A a $\delta(q, A)$ se skládá z trojic (p, C, D) (zde $D = 1$ znamená posun hlavy doprava, $D = -1$ znamená posun hlavy doleva), pak formule má tvar:

$$\bigwedge_j \bigwedge_{A \in \Gamma} ((s_i^q \wedge h_{i,j} \wedge t_{i,j}^A) \Rightarrow \bigvee (s_{i+1}^p \wedge t_{i+1,j}^C \wedge h_{i+1,j+D})).$$

Bod 6. Obsah polí kromě j-tého zůstává v čase $i+1$ stejný:

$$\bigwedge_j \bigwedge_{A \in \Gamma} ((\neg h_{i,j} \wedge t_{i,j}^A) \Rightarrow t_{i+1,j}^A).$$

Bod 7. Na konci práce Turingova stroje, tj. v čase $p(n)$ je stroj ve stavu q_f.

$$s_{p(n)}^{q_f}.$$

Výslednou formuli dostaneme jako konjunkci všech dílčích formulí pro všechny časové okamžiky $i = 0, 1, \ldots, p(n)$.

4.4 Převody úloh

4.4.1 Na základě tvrzení 4.3.4 víme: K důkazu, že rozhodovací úloha $\mathcal{U}$ ze třídy $\mathcal{NP}$ je $\mathcal{NP}$ úplná, stačí, abychom ukázali, že se na $\mathcal{U}$ polynomiálně redukuje některá $\mathcal{NP}$ úplná úloha. Zatím jediná $\mathcal{NP}$ úplná úloha, kterou známe, je SAT, splňování booleovských formulí v konjunktivním normálním tvaru. Ukážeme řadu polynomiálních redukcí a tím ukážeme, že i další rozhodovací úlohy jsou $\mathcal{NP}$ úplné.

4.4.2 $3-CNF\ SAT$**.** Úloha: Je dána formule φ v konjunktivním normálním tvaru, kde každá klauzule má 3 literály.
Otázka: Je formule φ splnitelná?

4.4.3 Tvrzení. Platí

$$SAT \lhd_p 3-CNF\,SAT.$$

4.4.4 Nástin převodu SAT na $3-CNF\ SAT$. Je dána formule φ v konjunktivním normálním tvaru. Zkonstruujeme formuli ψ, která

1. je v konjunktivním normálním tvaru, kde každá klauzule obsahuje maximálně 3 literály;
2. je splnitelná právě tehdy, když je splnitelná formule φ.

Označme $C_1, C_2, \ldots, C_k$ všechny klauzule formule φ. Jestliže každá z klauzulí obsahuje nejvýše 3 literály, nemusíme nic konstruovat, v tomto případě je $\psi = \varphi$.

Pro každou klauzuli C, která obsahuje víc než 3 literály, sestrojíme formuli ψ_C takto: Nechť $C = l_1 \vee l_2 \vee \ldots \vee l_s$, kde l_i jsou literály. Zavedeme nové logické proměnné $x_1, x_2, \ldots, x_{s-3}$ a položíme

$$\psi_C = (l_1 \vee l_2 \vee x_1) \wedge (\neg x_1 \vee l_3 \vee x_2) \wedge (\neg x_2 \vee l_4 \vee x_3) \wedge \ldots \wedge (\neg x_{s-3} \vee l_{s-1} \vee l_s).$$

Platí: Formule ψ_C je splnitelná právě tehdy, když C je splnitelná.

Formuli ψ dostaneme jako konjunkci všech klauzulí formule φ, které mají nejvýše 3 literály a formulí ψ_C pro klauzule C o více než 3 literálech.

Předpokládejme, že formule φ má k klauzulí a nejdelší klauzule má s literálů. Pak v konstrukci ψ jsme přidali maximálně $(s-3)k$ nových logických proměnných (rovnost nastává v případě, že každá z klauzulí formule φ obsahuje přesně $s > 3$ literálů). Navíc jsme formuli prodloužili o maximálně o $2(s-3)k$ literálů (každá nová logická proměnná se ve formuli ψ objevuje přesně dvakrát). Tedy délka formule ψ se pouze polynomiálně zvětšila vzhledem k délce formule φ.

4.4.5 Důsledek. Protože úloha $3-CNF\ SAT$ je ve třídě $\mathcal{NP}$, jedná se o $\mathcal{NP}$ úplnou úlohu.

4.4.6 Obarvení vrcholů grafu. Je dán prostý neorientovaný graf bez smyček $G = (V, E)$. *Obarvení vrcholů* grafu G je přiřazení, které každému vrcholu v grafu G přiřazuje jeho barvu $b(v)$, $b(v)$ je prvek množiny (barev) B, pro které platí, že žádné dva vrcholy spojené hranou nemají stejnou barvu. (Jinými slovy, jestliže $\{u, v\}$ je hrana grafu G, pak $b(u) \neq b(v)$.)

Graf G se nazývá k-*barevný*, jestliže jeho vrcholy je možné obarvit k barvami (tj. množina B má k prvků).

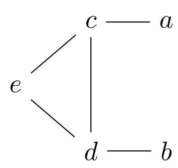
4.4.7 k-barevnost. Úloha: Je dán prostý neorientovaný graf G bez smyček a číslo k.
Otázka: Je graf G k-barevný?

4.4.8 Tvrzení. Platí

$$3 - CNF\ SAT \lhd_p \text{3-barevnost.}$$

4.4.9 Základní myšlenka převodu. Je dána formule φ, která je v CNF a každá klauzule má 2 nebo 3 literály. K důkazu je třeba zkonstruovat prostý neorientovaný graf G bez smyček takový, že φ je splnitelná právě tehdy, když G je 3-barevný.

Konstrukce využívá pomocný graf G_1 o pěti vrcholech $\{a, b, c, d, e\}$ a pěti hranách



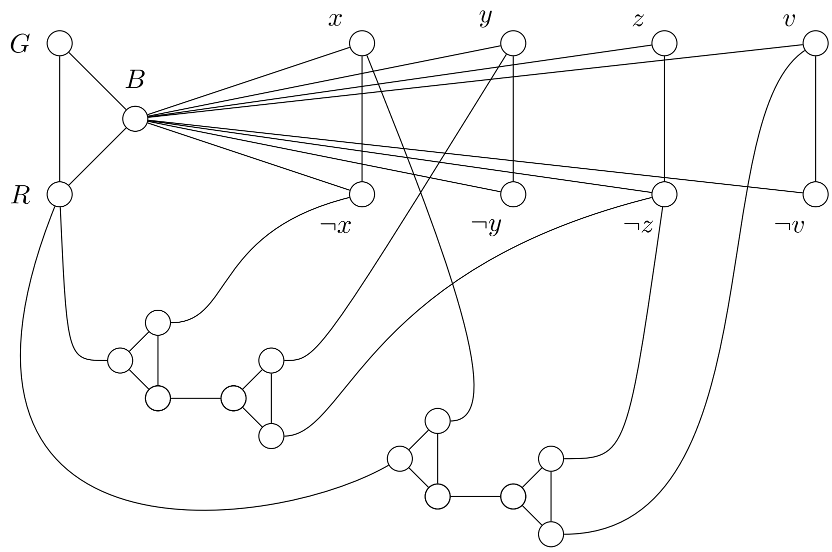
s touto vlastností:

- Jestliže vrcholy a a b mají stejnou barvu, pak tuto barvu musí mít i vrchol e.
- Jestliže jeden z vrcholů a a b má barvu z, pak lze tento graf obarvit tak, aby i vrchol e měl barvu z.

Mějme formuli φ, označme $x_1, x_2, \ldots, x_n$ všechny logické proměnné, které se ve formuli φ vyskytují. Vytvoříme neorientovaný graf $G = (V, E)$, kde

- V obsahuje všechny literály, tj. $x_1, \neg x_1, \ldots, x_n, \neg x_n$, vrcholy R, G, B.
- E obsahuje hrany tak, že R, G, B, $B, x_i, \neg x_i$ pro každé $i = 1, \ldots, n$ tvoří trojúhelník.
- Pro každou klauzuli obsahující literály l_1, l_2, l_3 přidáme do grafu dvě kopie pomocného grafu G_1 a to takto: Literály l_2 a l_3 odpovídají vrcholům a a b první kopie pomocného grafu G_1, vrcholy l_1 a e odpovídají vrcholům a, b a vrchol R je vrchol e druhé kopii grafu G_1,

Příklad grafu G pro dvě klauzule $C_1 = \neg z \vee y \vee \neg x$ a $C_2 = t \vee \neg z \vee x$ (x, y, z, t jsou logické proměnné) je na následujícím obrázku.



Předpokládejme, že formule φ je splnitelná; máme tedy pravdivostní ohodnocení, ve kterém je φ pravdivá. Obarvíme graf G třemi barvami z (zelená),c (červená) a m (modrá) takto:

- Vrcholy R, G, B: $b(R) = c$, $b(G) = z$, $b(B) = m$.
- Vrchol odpovídající literálu l má barvu z právě tehdy, když je l pravdivý, v opačném případě jej obarvíme c.

Protože každá klauzule obsahuje alespoň jeden literál, který je pravdivý, tj. jeho vrchol je obarven barvou z, je možné obarvit i zbývající vrcholy tak, aby G byl tříbarevný.

Předpokládejme, že graf G je tříbarevný. Přejmenujme barvy tak, aby platilo: $b(R) = c$, $b(G) = z$, $b(B) = m$. Nyní definujeme pravdivostní ohodnocení logických proměnných x_1, x_2, $\ldots$, x_n takto:

$$\text{proměnná } x_i \text{ je pravdivá iff } b(x_i) = z \text{ a proměnná } x_i \text{ je nepravdivá iff } b(x_i) = c.$$

Z vlastností pomocného grafu G_1 vyplývá, že v každé klauzuli je alespoň jeden literál, který je obarven barvou z, tudíž je pravdivý.

Není těžké nahlédnout, že počet vrcholů i hran grafu G je polynomiální vůči délce formule φ.

4.4.10 Důsledek. Protože 3-barevnost je ve třídě $\mathcal{NP}$, jedná se o $\mathcal{NP}$ úplnou úlohu.

4.4.11 Tvrzení. Platí

$$\text{3-barevnost } \lhd_p \ ILP.$$

4.4.12 Převod 3-barevnosti na *ILP*. Je dán prostý neorientovaný graf bez smyček $G = (V, E)$. Zkonstruujeme instanci I úlohy celočíselného lineárního programování takovou, že I má přípustné řešení právě tehdy, když graf G je 3-barevný.

Všechny proměnné budou nabývat hodnot 0 nebo 1 (tj. bude se jednat o tzv. 0-1 celočíselné lineární programování).

Proměnné: Pro každý vrchol $v \in V$ zavedeme tři proměnné:

$$x_v^c, x_v^m, x_v^z.$$

Význam: Fakt, že proměnná x_v^b je rovna 1, $b \in \{c, m, z\}$, znamená, že vrchol v má barvu b.

Podmínky:

- Pro každý vrchol $v \in V$ máme rovnici, která zaručuje, že vrchol v má právě jednu barvu – buď c nebo m nebo z:
$$x_v^c + x_v^m + x_v^z = 1.$$
- Pro každou hranu $e = \{u, v\}$ máme tři nerovnosti (pro každou barvu jednu) zaručující, že oba vrcholy u a v nemohou mít stejnou barvu:
$$x_u^c + x_v^c \leq 1, \;\; x_u^m + x_v^m \leq 1, \;\; x_u^z + x_v^z \leq 1.$$

Platí: Graf G je 3-barevný právě tehdy, když I má přípustné řešení.

Instance I má $3\,|V|$ proměnných a $|V| + 3\,|E|$ podmínek. Jedná se tedy o instanci velikosti $\mathcal{O}(n + m)$, kde $n = |V|$ a $m = |E|$.

4.4.13 Důsledek. Protože ILP je ve třídě $\mathcal{NP}$, jedná se o $\mathcal{NP}$ úplnou úlohu.

4.4.14 Problém rozkladu. Úloha: Je dána konečná množina X a systém jejích podmnožin $\mathcal{S}$.

Otázka: Je možné z $\mathcal{S}$ vybrat prvky tak, že tvoří rozklad množiny X? Jinými slovy, existuje $\mathcal{A} \subseteq \mathcal{S}$ tak, že $\mathcal{A}$ je rozklad množiny X?

4.4.15 Tvrzení. Platí
$$\text{3-barevnost } \lhd_p \text{ problém rozkladu.}$$

4.4.16 Převod 3-barevnosti na problém rozkladu. Je dán neorientovaný prostý graf bez smyček $G = (V, E)$. Zkonstruujeme množinu X a systém jejích podmnožin $\mathcal{S}$ tak, že graf G je tříbarevný právě tehdy, když ze systému $\mathcal{S}$ lze vybrat rozklad množiny X.

Množina X:

- Pro každý vrchol $v \in V$ dáme do množiny X prvky
$$v, p_v^c, p_v^m, p_v^z.$$
- Pro každou hranu $e = \{u, v\}$ dáme do množiny X prvky
$$q_{uv}^c, q_{uv}^m, q_{uv}^z, q_{vu}^c, q_{vu}^m, q_{vu}^z.$$

Množina X má $4\,|V| + 6\,|E|$ prvků.

Systém podmnožin $\mathcal{S}$ tvoří tyto množiny:

1. Pro každý vrchol $v \in V$:
$$\{v, p_v^c\}, \{v, p_v^m\}, \{v, p_v^z\}.$$
2. Pro každý vrchol $v \in V$ označme $N(v)$ množinu všech sousedů vrcholu v (tj. $N(v) = \{u \mid \{u, v\} \in E\}$). Do $\mathcal{S}$ dáme množiny:
$$S_v^c = \{p_v^c, q_{vu}^c \mid u \in N(v)\}, S_v^m = \{p_v^m, q_{vu}^m \mid u \in N(v)\}, S_v^z = \{p_v^z, q_{vu}^z \mid u \in N(v)\}.$$
3. Pro každou hranu $e = \{u, v\}$ dáme do $\mathcal{S}$ množiny:
$$\{q_{uv}^c, q_{vu}^m\}, \{q_{uv}^c, q_{vu}^z\}, \{q_{uv}^m, q_{vu}^c\}, \{q_{uv}^m, q_{vu}^z\}, \{q_{uv}^z, q_{vu}^c\}, \{q_{uv}^z, q_{vu}^m\}.$$

Systém $\mathcal{S}$ má $3\,|V|$ množin z 1), $3\,|V|$ množin z 2) a $6\,|E|$ množina z 3).

Je-li graf G 3-barevný, je možné jeho vrcholy obarvit barvami $\{c, m, z\}$. Označme $b(v)$ barvu vrcholu $v \in V$. Z systému $\mathcal{S}$ vybereme $\mathcal{A}$ takto:

$\mathcal{A}$ se skládá z:

1. $\{v, p_v^{b(v)}\}$ pro všechny $v \in V$,
2. $S_v^{b_1}$ a $S_v^{b_2}$, kde b_1 a b_2 jsou zbylé dvě barvy, kterými není obarven vrchol v,
3. $\{q_{uv}^{b(u)}, q_{vu}^{b(v)}\}$ pro každou hranu $e = \{u, v\}$,

Jestliže existuje rozklad $\mathcal{A} \subseteq \mathcal{S}$ množiny X, pak sestrojíme obarvení grafu G takto:

$$b(v) := b, b \in \{c, m, z\} \quad \text{právě tehdy, když} \quad \{v, p_v^b\} \in \mathcal{A}.$$

Není těžké dokázat, že z volby systému $\mathcal{S}$ a $\mathcal{A}$ vyplývá: b je obarvení vrcholů třemi barvami.

4.4.17 Důsledek. Protože problém rozkladu je ve třídě $\mathcal{NP}$, jedná se o $\mathcal{NP}$ úplnou úlohu.

4.4.18 SubsetSum. Úloha: Jsou dána kladná čísla $a_1, a_2, \ldots, a_n$ a číslo K.
Otázka: Lze vybrat podmnožinu čísel $a_1, a_2, \ldots, a_n$ tak, aby jejich součet byl roven číslu K?

Jinými slovy, existuje $J \subseteq \{1, 2, \ldots, n\}$ tak, že

$$\sum_{i \in J} a_i = K.$$

4.4.19 Tvrzení. Platí

$$\text{problém rozkladu} \lhd_p \text{SubsetSum}.$$

4.4.20 Převod problému rozkladu na SubsetSum. Je dána konečná množina X a systém jejích podmnožin $\mathcal{S}$. Přejmenujeme prvky X tak, že $X = \{0, 1, \ldots, n-1\}$ a $\mathcal{S} = \{S_1, S_2, \ldots, S_r\}$.

Zvolíme přirozené číslo p větší než r (počet prvků $\mathcal{S}$). Každé podmnožině S_i přiřadíme kladné číslo a_i takto: Ke každé množině S_i označíme χ_{S_i} její charakteristickou funkci; tj. $\chi_{S_i}(j) = 1$ právě tehdy, když $j \in S_i$. Pak

$$S_i \longrightarrow \sum_{j=0}^{n-1} \chi_{S_i}(j)\, p^j = a_i.$$

Nakonec zvolíme číslo $K = \sum_{i=0}^{n-1} p^i$.

Protože $p > r$, není těžké ukázat, že

$$\sum_{i \in J} a_i = K \ \text{ právě tehdy, když } \ \mathcal{A} = \{S_i \mid i \in J\} \text{ je rozklad } X.$$

4.4.21 Důsledek. Protože SubsetSum je ve třídě $\mathcal{NP}$, jedná se o $\mathcal{NP}$ úplnou úlohu.

4.4.22 Poznámka. Nyní není těžké sestrojit polynomiální redukci problému SubsetSum na problém dělení kořisti nebo na problém batohu. Proto jsou i tyto dvě úlohy $\mathcal{NP}$ úplné.

4.4.23 Problém klik. Úloha: Je dán prostý neorientovaný graf $G = (V, E)$ bez smyček a číslo k.
Otázka: Existuje v grafu G klika o alespoň k vrcholech?

4.4.24 Tvrzení. Platí

$$3 - CNF\ SAT \lhd_p \text{ problém klik.}$$

4.4.25 Nástin převodu $3 - CNF\,SAT$ na problém klik. Je dána formule φ v CNF, s k klauzulemi $C_1, C_2, \ldots, C_k$, kde každá klauzule má 3 literály. Sestrojíme k-partitní neorientovaný graf $G = (V, E)$ takto:

G má pro každou klauzuli jednu stranu; strana odpovídající klauzuli C se skládá ze 3 vrcholů označených literály klauzule C. Hrany grafu G vedou vždy mezi dvěma stranami a to tak, že spojují dva literály, které nejsou komplementární (tj. jeden není negací druhého).

Platí: Formule φ je splnitelná právě tehdy, když v grafu G existuje klika o k vrcholech. (Poznamenejme, že k je počet klauzulí formule φ.)

Jestliže φ je pravdivá v ohodnocení u, vybereme v každé klauzuli formule φ jeden literál, který je v daném ohodnocení pravdivý. Pak množina vrcholů odpovídajících těmto literálům tvoří kliku v G o k vrcholech.

Jestliže v grafu G existuje klika A o k vrcholech, pak A má jeden vrchol v každé straně grafu G. Položme jako pravdivé všechny literály, které se nacházejí v A a hodnoty ostatních logických proměnných zadefinujme libovolně. Pak v tomto ohodnocení je formule φ pravdivá.

Zkonstruovaný graf G má tolik vrcholů jako má formule φ literálů, tj. n vrcholů, kde n je délka formule φ. Vzhledem k tomu, že prostý graf s n vrcholy má $\mathcal{O}(n^2)$ hran, jedná se o polynomiální redukci.

4.4.26 Důsledek. Protože problém klik je ve třídě $\mathcal{NP}$, jedná se o $\mathcal{NP}$ úplnou úlohu.

4.4.27 Nezávislé množiny. Je dán prostý neorientovaný graf $G = (V, E)$ bez smyček. Množina vrcholů $N \subseteq V$ se nazývá *nezávislá množina* v G, jestliže žádná hrana grafu G nemá oba krajní vrcholy v N. Jinými slovy, indukovaný podgraf množinou N je diskrétní graf.

Úloha: Je dán prostý neorientovaný graf G bez smyček a číslo k.
Otázka: Existuje v G nezávislá množina o k vrcholech?

4.4.28 Tvrzení. Platí

$$\text{problém klik} \lhd_p \text{nezávislé množiny.}$$

4.4.29 Převod problému klik na nezávislé množiny. Je dán prostý neorientovaný graf bez smyček $G = (V, E)$. Definujeme opačný graf $G^{op} = (V, E^{op})$ takto:

$$\{u, v\} \in E^{op} \text{ právě tehdy, když } u \neq v \text{ a } \{u, v\} \notin E.$$

Platí: Množina $A \subseteq V$ je klika v grafu G právě tehdy, když je maximální nezávislou množinou v grafu G^{op}. (Jinými slovy, A je nezávislá množina a přidáním libovolného vrcholu už nebude nezávislá.)

To, že se jedná o polynomiální redukci vyplývá z faktu, že všech hran v grafu G i doplňkovém grafu G^{op} je $\frac{n(n-1)}{2}$, kde n je počet vrcholů.

4.4.30 Důsledek. Protože úloha o nezávislých množinách je ve třídě $\mathcal{NP}$, jedná se o $\mathcal{NP}$ úplnou úlohu.

4.4.31 Vrcholové pokrytí. Je dán prostý neorientovaný graf bez smyček $G = (V, E)$. Podmnožina vrcholů $B \subseteq V$ se nazývá *vrcholové pokrytí* G, jestliže každá hrana grafu G má alespoň jeden krajní vrchol v množině B.

Poznamenejme, že celá množina vrcholů V je vrcholovým pokrytím, problém je najít vrcholové pokrytí o co nejmenším počtu vrcholů.

Úloha: Je dán prostý neorientovaný graf G bez smyček a číslo k.
Otázka: Existuje v grafu G vrcholové pokrytí o k vrcholech?

4.4.32 Tvrzení. Platí

$$\text{nezávislé množiny} \lhd_p \text{vrcholové pokrytí.}$$

4.4.33 Nástin převodu nezávislých množin na vrcholové pokrytí. Platí: Je-li množina N nezávislá množina grafu G, pak množina $V \setminus N$ je vrcholovým pokrytím grafu G. A naopak, je-li B vrcholové pokrytí grafu G, pak množina $V \setminus B$ je nezávislá množina v G.

Proto: Je dán prostý neorientovaný graf G bez smyček a číslo k. Pak v G existuje nezávislá množina o k vrcholech právě tehdy, když v G existuje vrcholové pokrytí o $n - k$ vrcholech, kde $n = |V|$ je počet vrcholů grafu G.

4.4.34 Důsledek. Protože problém vrcholového pokrytí je ve třídě $\mathcal{NP}$, jedná se o $\mathcal{NP}$ úplnou úlohu.

4.4.35 Existence hamiltonovského cyklu. Je dán orientovaný graf G.

Otázka: Existuje v grafu G hamiltonovský cyklus? (Jinými slovy, existuje v grafu G cyklus procházející všemi vrcholy?)

4.4.36 Tvrzení. Platí

$$\text{vrcholové pokrytí} \lhd_p \text{existence hamiltonovského cyklu.}$$

4.4.37 Základní myšlenka převodu. Převod je založen na využití speciálního grafu H o 4 vrcholech a 6 orientovaných hranách. Graf H má tuto vlastnost: Má-li být graf součástí hamiltonovského cyklu, pak jsou jen dva základní způsoby průchodu grafem H, buď se projdou všechny vrcholy za sebou, nebo při dvojím průchodu vždy dva a dva.

Předpokládejme, že je dán neorientovaný prostý graf $G = (V, E)$ bez smyček a číslo k. Je možno vytvořit orientovaný graf G' takový, že v G existuje vrcholové pokrytí o k vrcholech právě tehdy, když v G' existuje hamiltonovský cyklus.

Graf G' se, zhruba řečeno, vytvoří takto: Za každou hranu grafu G do G' dáme kopii grafu H. Kromě takto získaných vrcholů přidáme ještě vrcholy $1, 2, \ldots, k$. Celkově tedy počet vrcholů grafu G' je $4\,|E| + k$. Hrany grafu G' jsou jednak hrany všech kopií grafu H, jednak hrany vedoucí mezi nimi a dále hrany do a z vrcholů $1, 2, \ldots, k$. Celkově je hran grafu G' také úměrně počtu hran grafu G plus k-násobek počtu vrcholů grafu G. To znamená, že redukce je polynomiální.

4.4.38 Důsledek. Protože problém existence hamiltonovského cyklu je ve třídě $\mathcal{NP}$, jedná se o $\mathcal{NP}$ úplnou úlohu.

4.4.39 Tvrzení. Platí

$$\text{existence hamiltonovské kružnice} \lhd_p \text{problém obchodního cestujícího.}$$

Převod zmíněný v tvrzení je velmi jednoduchý a je ponechán studentům jako domácí úkol.

4.4.40 Důsledek. Protože problém obchodního cestujícího je ve třídě $\mathcal{NP}$, jedná se o $\mathcal{NP}$ úplnou úlohu.

4.4.41 Tvrzení. Platí

$$\text{existence orientované hamiltonovské cesty} \ \lhd_p \ \text{nejdelší cesty v orientovaném grafu.}$$

Převod zmíněný v tvrzení je velmi jednoduchý.

4.4.42 Důsledek. Protože problém nejdelších cest v orientovaném grafu je ve třídě $\mathcal{NP}$, jedná se o $\mathcal{NP}$ úplnou úlohu.

4.4.43 Tvrzení. Platí

$$\text{nejdelší cesty v orientovaném grafu} \ \lhd_p \ \text{nejkratší cesty v orientovaném grafu.}$$

Převod zmíněný v tvrzení je velmi jednoduchý.

4.4.44 Důsledek. Protože problém nejkratších cest v orientovaném grafu je ve třídě $\mathcal{NP}$, jedná se o $\mathcal{NP}$ úplnou úlohu.

4.4.45 Heuristiky. Jestliže je třeba řešit problém, který je $\mathcal{NP}$ úplný, musíme pro větší instance opustit myšlenku přesného nebo optimálního řešení a smířit se s tím, že získáme „dostatečně přesné“ nebo „dostatečně kvalitní“ řešení. K tomu se používají heuristické algoritmy pracující v polynomiálním čase. Algoritmům, kde umíme zaručit „jak daleko“ je nalezené řešení od optimálního, se také říká aproximační algoritmy.

4.4.46 Heuristika pro vrcholové pokrytí — 1. Uvažujme následující heuristický algoritmus, který pro daný neorientovaný graf najde jeho vrcholové pokrytí. Algoritmus je založen na „hladovém postupu“.

Vstup: neorientovaný graf $G = (V, E)$.

Výstup: vrcholové pokryti C grafu G.

```
begin
    C := ∅
        while E ≠ ∅ do
            vyber vrchol v s největším stupněm
            C := C ∪ {v}
            odstraň v spolu s hranami s ním incidentními
end
return C
```

Přestože algoritmus „vypadá rozumně“, v některých případech najde vrcholové pokrytí, které má podstatně víc vrcholů než nejméně početné pokrytí. Zde tím „podstatně“ rozumíme toto: existuje graf G, který má vrcholové pokrytí o k vrcholech, ale výše uvedený algoritmus najde vrcholové pokrytí o $\Theta(k \lg k)$ vrcholech.

4.4.47 Heuristika pro vrcholové pokrytí — 2. Uvažujme ještě jeden heuristický algoritmus, který pro daný neorientovaný graf najde jeho vrcholové pokrytí.

Vstup: neorientovaný graf $G = (V, E)$.

Výstup: vrcholové pokryti C grafu G.

```
begin
    C := ∅
        while E ≠ ∅ do
            vyber hranu {u, v}
            C := C ∪ {u, v}
            odstraň vrcholy u, v spolu se všemi hranami s nimi incidentními
end
return C
```

Dá se dokázat, že druhý algoritmus vždy najde vrcholové pokrytí, které obsahuje maximálně dvakrát tolik vrcholů než je počet nejméně početného vrcholového pokrytí.

4.4.48 Tvrzení. Označme C_{min} nejméně početné vrcholové pokrytí grafu G. Pak druhá heuristika najde vrcholové pokrytí C takové, že

$$|C| \leq 2\,|C_{min}|.$$

Důkaz. Označme F množinu všech hran, která byla vybrána algoritmem. Pak $|C| = 2\,|F|$; ano, za každou vybranou hranu jsme do množiny C vložili dva vrcholy — krajní vrcholy této hrany. Navíc, žádné dvě hrany v množině F nemají společný vrchol; tudíž je jejich pokrytí je třeba $|F|$ vrcholů. Proto $|C_{min}| \geq |F|$ a $|C| = 2\,|F| \leq 2\,|C_{min}|$.

4.4.49 Aproximační algoritmus. Definice. Uvažujme optimalizační problém $\mathcal{U}$. Polynomiální algoritmus $\mathcal{A}$ se nazývá R *aproximační algoritmus*, jestliže existuje reálné číslo R takové, že pro každou instanci algoritmus $\mathcal{A}$ najde přípustné řešení ne horší než R krát hodnota optimálního řešení.

To znamená, že pro minimalizační úlohu najde řešení, které nemá hodnotu účelové funkce větší než R krát hodnotu optimálního řešení; pro maximalizační úlohu najde řešení, které nemá hodnotu účelové funkce menší než R krát hodnotu optimálního řešení.

Druhá heuristika pro nalezení vrcholového pokrytí je tedy 2 aproximační algoritmus pro problém vrcholového pokrytí.

Ne pro všechny úlohy, jejichž rozhodovací verze jsou NP úplné, aproximační algoritmy existují. Příkladem je problém obchodního cestujícího, jak ukazuje následující věta.

4.4.50 Tvrzení. Kdyby existovala konstanta r a polynomiální algoritmus $\mathcal{A}$ takový, že pro každou instanci obchodního cestujícího I najde trasu délky $D \leq r\,OPT(I)$, kde $OPT(I)$ je délka optimální trasy instance I, pak

$$\mathcal{P} = \mathcal{NP}.$$

4.4.51 Zdůvodnění tvrzení 4.4.50. Za předpokladu tvrzení 4.4.50 bychom uměli polynomiálně vyřešit problém existence hamiltonovské kružnice. Naznačíme odpovídající převod.

Je dán neorientovaný graf $G = (V, E)$, $V = \{1, 2, \ldots, n\}$, a ptáme se, zda v něm existuje hamiltonovská kružnice. Zkonstruujeme instanci obchodního cestujícího takto: Pro města $\{1, 2, \ldots, n\}$ položíme

$$d(i,j) = \begin{cases} 1, & \{i,j\} \in E \\ r\,n+1, & \{i,j\} \notin E \end{cases}$$

Trasa v instanci popsané výše může mít délku n, jestliže je tvořena všemi hranami délky 1. V tomto případě jsou všechny hrany hranami grafu G a trasa představuje hamiltonovskou kružnici. Nebo musí trasa mít délku alespoň $n - 1 + nr + 1 = nr + n$. To je v případě, že aspoň jedna spojnice v trase není tvořena hranou grafu G.

Tedy jestliže algoritmus $\mathcal{A}$ najde trasu délky jiné než n, pak v grafu G neexistuje hamiltonovská kružnice. Takto bychom polynomiálním algoritmem byli schopni rozhodnout existenci hamiltonovské kružnice. Protože existence hamiltonovské kružnice je $\mathcal{NP}$ úplný problém, platilo by $\mathcal{P} = \mathcal{NP}$.

4.4.52 Trojúhelníková nerovnost. Řekneme, že instance obchodního cestujícího splňuje trojúhelníkovou nerovnost, jestliže pro každá tři města i, j, k platí:

$$d(i,j) \leq d(i,k) + d(k,j).$$

4.4.53 Tvrzení. Jestliže instance I obchodního cestujícího splňuje trojúhelníkovou nerovnost, pak existuje polynomiální algoritmus $\mathcal{A}$, který pro I najde trasu délky D, kde $D \leq 2\,OPT(I)$ ($OPT(I)$ je délka optimální trasy v I).

4.4.54 Slovní popis algoritmu z tvrzení 4.4.53. Instanci I považujeme za úplný graf G s množinou vrcholů $V = \{1, 2, \ldots, n\}$ a ohodnocením d.

1. V grafu G najdeme minimální kostru (V, K).
2. Kostru (V, K) prohledáme do hloubky z libovolného vrcholu.
3. Trasu T vytvoříme tak, že vrcholy procházíme ve stejném pořadí jako při prvním navštívení během prohledávání grafu. T je výstup em algoritmu.

Zřejmě platí, že délka kostry K je menší než $OPT(I)$. Ano, vynecháme-li z optimální trasy některou hranu, dostaneme kostru grafu G. Protože K je minimální kostra, musí být délka K menší než $OPT(I)$ (předpokládáme, že vzdálenosti měst jsou kladné). Vzhledem k platnosti trojúhelníkové nerovnosti, je délka T menší nebo rovna dvojnásobku délky kostry K.

4.4.55 Christofidesův algoritmus. Jestliže instance I obchodního cestujícího splňuje trojúhelníkovou nerovnost, pak následující algoritmus najde trasu T délky D takovou, že $D \leq \frac{3}{2}\,OPT(I)$.

Instanci I považujeme ze úplný graf G s množinou vrcholů $V = \{1, 2, \ldots, n\}$ a ohodnocením d.

1. V grafu G najdeme minimální kostru (V, K).
2. Vytvoříme úplný graf H na množině všech vrcholů, které v kostře (V, K) mají lichý stupeň.
3. V grafu H najdeme nejlevnější perfektní párování P.
4. Hrany P přidáme k hranám K minimální kostry. Graf $(V, P \cup K)$ je eulerovský graf. V grafu $(V, P \cup K)$ sestrojíme uzavřený eulerovský tah.
5. Trasu T získáme z eulerovského tahu tak, že vrcholy navštívíme v pořadí, ve kterém jsme do nich poprvé vstoupili při tvorbě eulerovského tahu.

Platí, že délka takto vzniklé trasy je maximálně $\frac{3}{2}$ krát větší než délka optimální trasy.

4.4.56 Poznámka. Odhad délky trasy, kterou jsme získali v 4.4.54, i odhad pro trasu získanou Christofidesovým algoritmem není možné zlepšit.

4.5 Třída co-$\mathcal{NP}$

4.5.1 Pozorování. Je-li jazyk L ve třídě $\mathcal{P}$, pak i jeho doplněk $\overline{L}$ patří do třídy $\mathcal{P}$. Obdobné tvrzení se pro jazyky třídy $\mathcal{NP}$ neumí dokázat.

4.5.2 Definice. Jazyk L patří do třídy co-$\mathcal{NP}$, jestliže jeho doplněk patří do třídy $\mathcal{NP}$.

4.5.3 Příklady.

- Jazyk $USAT$, který je doplňkem jazyka SAT splnitelných booleovských formulí, leží ve třídě co-$\mathcal{NP}$. (Jazyk $USAT$ se skládá ze všech nesplnitelných booleovských formulí a ze všech slov, které neodpovídají booleovské formuli.)
- Jazyk $TAUT$, který se skládá ze všech slov odpovídajících tautologii výrokové logiky, patří do třídy co-$\mathcal{NP}$.

4.5.4 Otázka, zda co-$\mathcal{NP} = \mathcal{NP}$, je otevřená.

4.5.5 Lemma. Mějme dva jazyky L_1 a L_2, pro které platí $L_1 \lhd_p L_2$. Pak platí také $\overline{L_1} \lhd_p \overline{L_2}$, (kde $\overline{L}$ je doplněk jazyka L).

Zdůvodnění. Jestliže $L_1 \lhd_p L_2$, $L_1 \subseteq \Sigma^\star$, $L_2 \subseteq \Phi^\star$, pak existuje polynomiální algoritmus M, který pro každé slovo $w \in \Sigma^\star$ zkonstruuje slovo $M(w) \in \Phi^\star$ a to tak, že

$$w \in L_1 \quad \text{právě tehdy, když} \quad M(w) \in L_2.$$

To ale znamená, že

$$w \notin L_1 \quad \text{právě tehdy, když} \quad M(w) \notin L_2,$$

a tedy $\overline{L_1} \lhd_p \overline{L_2}$.

4.5.6 Tvrzení. Platí co-$\mathcal{NP} = \mathcal{NP}$ právě tehdy, když existuje $\mathcal{NP}$ úplný jazyk, jehož doplněk je ve třídě $\mathcal{NP}$.

Důkaz. Jestliže co-$\mathcal{NP} = \mathcal{NP}$, pak každý doplněk nějakého $\mathcal{NP}$ úplného jazyka leží ve třídě $\mathcal{NP}$.

Předpokládejme, že existuje $\mathcal{NP}$ úplný jazuk L, jehož doplněk $\overline{L}$ leží ve třídě $\mathcal{NP}$. Ukážeme, že
co-$\mathcal{NP} \subseteq \mathcal{NP}$ a $\mathcal{NP} \subseteq$ co-$\mathcal{NP}$.

Vezměme libovolný jazyk L_1 ze třídy co-$\mathcal{NP}$. Pak $\overline{L_1} \in \mathcal{NP}$. Protože L je $\mathcal{NP}$ úplný jazyk, $\overline{L_1} \lhd_p L$ a podle předchozího lemmatu $L_1 \lhd_p \overline{L} \in \mathcal{NP}$. Odtud $L_1 \in \mathcal{NP}$.

Vezměme libovolný jazyk L_2 takový, že $L_2 \in \mathcal{NP}$. Pak $L_2 \lhd_p L$ a tedy (opět podle předchozího lemmatu) $\overline{L_2} \lhd_p \overline{L} \in \mathcal{NP}$. Proto $L_2 \in$ co-$\mathcal{NP}$.

4.6 Třídy $\mathcal{P}$SPACE a $\mathcal{NP}$SPACE

4.6.1 Je dán Turingův stroj M (deterministický nebo nedeterministický). Připomeňme, že M pracuje s paměťovou složitostí $p(n)$ právě tehdy, když pro každé slovo délky n nepoužije paměťovou buňku větší než $p(n)$. Uvědomte si: jestliže Turingův stroj přijímá jazyk s polynomiální časovou složitostí, pak se musí zastavit na **každém** vstupu (ať už leží v přijímané jazyce, nebo ne); tj. Turingův stroj jazyk rozhoduje. Podobné tvrzení neplatí pro Turingův stroj, který nějaký jazyk přijímá s polynomiální paměťovou složitostí; ano, Turingův stroj se může zacyklit na slově, které nepřijímá.

4.6.2 Třída $\mathcal{P}$SPACE. Jazyk L patří do třídy $\mathcal{P}$SPACE jestliže existuje deterministický Turingův stroj M, který přijímá jazyk L a pracuje s polynomiální paměťovou složitostí.

4.6.3 Tvrzení. Platí

$$\mathcal{P} \subseteq \mathcal{P}\texttt{SPACE}.$$

4.6.4 Třída $\mathcal{NP}$SPACE. Jazyk L patří do třídy $\mathcal{NP}$SPACE jestliže existuje nedeterministický Turingův stroj M, který přijímá jazyk L a pracuje s polynomiální paměťovou složitostí.

4.6.5 Tvrzení. Platí

$$\mathcal{NP} \subseteq \mathcal{NP}\texttt{SPACE}.$$

4.6.6 Věta. Je dán Turingův stroj M (deterministický nebo nedeterministický), který přijímá jazyk L s paměťovou složitostí $p(n)$ (kde p je nějaký polynom). Pak existuje konstanta c taková, že M přijme slovo w délky n po nejvýše $c^{p(n)+1}$ krocích.

4.6.7 Myšlenka důkazu věty 4.6.6. Konstantu c volíme tak, abychom měli zajištěno, že Turingův stroj M má při práci se vstupem délky n méně než $c^{p(n)+1}$ různých situací. Zajímají nás totiž pouze takové výpočty, ve kterých se situace neopakují. Označme t počet páskových symbolů Turingova stroje M a označme s počet stavů M. Pak M má $p(n)\, s\, t^{p(n)}$ různých situací; ano, stroj se může nacházet v s různých stavech, hlava může skenovat jedno z $p(n)$ polí pásky, a páska může mít jeden z $t^{p(n)}$ různých obsahů.

Položme $c = t + s$. Z binomické věty vyplývá, že

$$c^{p(n)+1} = (t+s)^{p(n)+1} = t^{p(n)+1} + (p(n)+1)\, t^{p(n)}\, s + \dots .$$

Odtud $c^{p(n)+1} \geq p(n)\, t^{p(n)}\, s$.

4.6.8 Věta. Je-li jazyk L ve třídě $\mathcal{P}$SPACE ($\mathcal{NP}$SPACE), pak L je rozhodován deterministickým (nedeterministickým) Turingovým strojem M s polynomiální paměťovou složitostí, který se vždy zastaví po nejvýše $c^{q(n)}$ krocích, kde $q(n)$ je vhodný polynom a c konstanta.

4.6.9 Myšlenka důkazy věty 4.6.8. Předpokládejme, že $L \in \mathcal{P}$SPACE. Pak existuje Turingův stroj M_1, který přijímá jazyk L s paměťovou složitostí $p(n)$ ($p(n)$ je vhodný polynom). Víme (z věty 4.6.6), že existuje konstanta c taková, že Turingův stroj M_1 potřebuje nejvýše $c^{p(n)+1}$ kroků.

Vytvoříme Turingův stroj M_2, který bude mít dvě pásky: první páska bude simulovat M_1, druhá bude počítat kroky na první pásce. Jestliže počet kroků překročí $c^{p(n)+1}$, Turingův stroj M_2 se neúspěsně zastaví. Počítání kroků M_2 provádí v c-adické soustavě (tak, aby zabralo jen $\mathcal{O}(p(n))$ polí pásky).

Hledaný Turingův stroj M je Turingův stroj s jednou páskou, který simuluje Turingův stroj M_2. Turingův stroj M pracuje v s časovou složitostí $\mathcal{O}(c^{2p(n)})$, tedy v maximálně $d\, c^{2p(n)}$ krocích. Nyní stačí položit $q(n) = 2p(n) + \log_c d$ nebo jakýkoli polynom větší.

4.6.10 Savitchova věta. Platí

$$\mathcal{P}\mathtt{SPACE} = \mathcal{NP}\mathtt{SPACE}.$$

4.6.11 Nástin myšlenky důkazu Savitchovy věty. Zřejmě $\mathcal{P}$SPACE $\subseteq \mathcal{NP}$SPACE. Důkaz opačné inkluze $\mathcal{NP}$SPACE $\subseteq \mathcal{P}$SPACE spočívá v tom, že jsme schopni nedeterministický Turingův stroj M pracující s paměťovou složitostí $p(n)$ simulovat deterministickým Turingovým strojem N, který pracuje s paměťovou složitostí $\mathcal{O}([p(n)]^2)$ (o časové složitosti nic dokazovat nebudeme). Konstrukce N je založena na následující rekursivní proceduře $DOSTP(I, J; m)$, kde i a J jsou situace NTM M a m je kladné přirozené číslo. $DOSTUP(I, J; m)$ vrátí 1, jestliže $I \vdash^\star J$ v nejvýše m krocích.

$DOSTUP(I, J; m)$

Vstup: Situace I a J NTM M a m je kladné přirozené číslo.

Výstup: TRUE, jestliže J je dostupná z I v nejvýše m krocích, FALSE v opačném případě.

```
begin
   if m = 1 then
         if I = J nebo I ⊢ J then return TRUE
         else return FALSE
   end
   else (induktivní část]
         for každou možnou situaci K do
         if DOSTUP(I, K; ⌊m/2⌋) a DOSTUP(K, J; ⌈m/2⌉ then
              return TRUE
         return FALSE
```

```
   end
end
```

Uvědomte si, že rekurzivní procedura $DOSTUP(I, J; m)$ má vždy na zásobníku jen jednu trojici $(I_1, J_1; m)$, nejvýše jednu trojici $(I_2, J_2; \frac{m}{2})$, nejvýše jednu $(I_3, J_3; \frac{m}{4})$, atd. Tedy současně nemá na zásobníku víc než $\lg m$ různých trojic.

Je dán nedeterministický Turingův stroj M, který přijímá jazyk L s polynomiální paměťovou složitostí $p(n)$. Pro vstup w voláme proceduru $DOSTUP(I_0, J; m)$, kde I_0 je počáteční situace M, J je některá přijímající situace M a $m = c^{p(n)+1}$ (c je konstanta z 4.6.6). Dá se dokázat, že pro vykonání procedury $DOSTUP(I, J; m)$ deterministickým Turingovým strojem stačí paměťová složitost $\mathcal{O}([p(n)]^2)$. To vyplývá z toho, že $DOSTUP(I_0, J; m)$ má na zásobníku maximálně $\lg c^{p(n)+1} = d\,p(n)$ trojic $(I, J; r)$ a každá z trojic má nejvýše délku $\mathcal{O}(p(n))$. (Uvědomte si, že nám nezáleží na tom, jak dlouho deterministický Turingův stroj pracuje, zajímáme se pouze o paměťové nároky.)

4.6.12 Důsledek. Platí

$$\mathcal{P} \subseteq \mathcal{NP} \subseteq \mathcal{P}\mathtt{SPACE}.$$

4.7 Testování prvočíselnosti

4.7.1 Jazyky L_p a L_s. Jazyk L_p obsahuje všechna prvočísla, jazyk L_s obsahuje všechna složená čísla; přesněji:

$$L_p = \{w \mid w \text{ je binární zápis prvočísla}\}$$

$$L_s = \{w \mid w \text{ je binární zápis složeného čísla}\}.$$

Jazyk L_s je (až na číslo 1) doplňkem jazyka L_p; přidáme-li 1 do jazyka L_s, pak dostáváme

$$L_s = \overline{L_p}, \quad L_p = \overline{L_s}.$$

4.7.2 Tvrzení. Jazyk L_s leží ve třídě $\mathcal{NP}$.

Zdůvodněni: Jestliže číslo n je složené, znamená to, že má dělitele r, pro nějž platí $1 < r < n$. Známe-li některého (tzv. vlastního) dělitele r, jsme schopni dělením čísla n číslem r zjistit, že n je opravdu složené číslo. Pro prvočíslo žádný takový vlastní dělitel neexistuje.

Nyní si stačí uvědomit, že vlastní dělitel je hledaný certifikát s polynomiální velkostí. Ano, délka binárního slova odpovídajícího n, je $k = \lg n$, délka dělitele r je $\mathcal{O}(k)$ a celočíselné dělení dvou binárních čísel délky k lze provést v polynomiálním čase vzhledem k délce binárního zápisu čísel.

4.7.3 Důsledek. Jazyk L_p je ve třídě co-$\mathcal{NP}$.

4.7.4 Tvrzení. Jazyk L_p je ve třídě $\mathcal{NP}$.

Najít polynomiální certifikát pro jazyk obsahující prvočísla je podstatně těžší než pro jazyk obsahující složená čísla. V tomto případě se jedná např. o generátor grupy $(\mathbb{Z}_p \setminus \{0\}, \odot, 1)$ (p prvočíslo); tj primitivní prvek konečného tělesa $(\mathbb{Z}_p, \oplus, \odot, 0, 1)$.

4.7.5 Důsledek. Jazyky L_p a L_s patří do průniku tříd $\mathcal{NP}$ a co-$\mathcal{NP}$.

4.7.6 V dalším ukážeme, že existuje pravděpodobnostní algoritmus — Millerův test prvočíselnosti, který pro dané velké liché číslo n s pravděpodobností aspoň $\frac{1}{2}$ rozhodne, zda n je prvočíslo. Dříve než algoritmus uvedeme, připomeneme několik faktů z algebry, které budeme potřebovat.

- Množina $\mathbb{Z}_n$ tzv. zbytkových tříd modulo n je

$$\mathbb{Z}_n = \{0, 1, \ldots, n-1\}.$$

- Na množině $\mathbb{Z}_n$ jsou definovány operace $\oplus$ a $\odot$ takto

$$a \oplus b = c, \quad \text{kde } c \text{ je zbytek při dělení čísla } a + b \text{ číslem } n,$$

$$a \odot b = c, \quad \text{kde } c \text{ je zbytek při dělení čísla } a.b \text{ číslem } n.$$

- $(\mathbb{Z}_n, \oplus, 0)$ je komutativní grupa, $(\mathbb{Z}_n, \odot, 1)$ je komutativní monoid a platí distributivní zákony

  Navíc, prvek $a \in \mathbb{Z}_n$ má inverzní prvek (vzhledem k operaci $\odot$) právě tehdy, když a a n jsou nesoudělná čísla.

  Proto $(\mathbb{Z}_n, \oplus, \odot, 0, 1)$ pro n prvočíslo je těleso; pro složená n, tělesem není.

- Podle malé Fermatovy věty pro a nesoudělné s prvočíslem p platí

$$a^{p-1} \equiv 1 \pmod{n}.$$

- Je-li H podgrupa konečné grupy G, pak počet prvků podgrupy H dělí počet prvků grupy G.

- Operace sčítání, násobení, umocňování a dělení v $\mathbb{Z}_n$ je možné provést v polynomiálním čase vzhledem k velikosti čísel, se kterými se operace provádějí.

4.7.7 Millerův test prvočíselnosti.

Vstup: velké liché přirozené číslo n.

Výstup: „prvočíslo“ nebo „složené“.

1. Spočítáme $n-1 = 2^l\, m$, kde m je liché číslo.

2. Náhodně vybereme $a \in \{1, 2, \ldots, n-1\}$.

3. Spočítáme $a^m \pmod n$,
   jestliže $a^m \equiv 1 \pmod n$, stop, výstup „prvočíslo“.

4. Opakovaným umocňováním počítáme
   $a^{2\,m} \pmod n, a^{2^2\,m} \pmod n, \ldots, a^{2^l\,m} \pmod n$.

5. Jestliže $a^{2^l\,m} \not\equiv 1 \pmod n$, stop, výstup „složené“.

6. Vezmeme k takové, že $a^{2^k\,m} \not\equiv 1 \pmod n$ a $a^{2^{k+1}\,m} \equiv 1 \pmod n$.
   Jestliže $a^{2^k\,m} \equiv -1 \pmod n$, stop, výstup „prvočíslo“.
   Jestliže $a^{2^k\,m} \not\equiv -1 \pmod n$, stop, výstup „složené“.

4.7.8 Věta.

1. Jestliže pro vstup n dá Millerův test prvočíselnosti odpověď „složené“, pak je číslo n složené.

2. Jestliže pro vstup n dá Millerův test prvočíselnosti odpověď „prvočíslo“, pak n je prvočíslo s pravděpodobností větší než $\frac{1}{2}$.

Idea důkazu. Add 1. Jestliže je číslo n prvočíslo, tak nemůžeme dostat výstup „složené“. Malá Fermatova věta totiž zaručuje, že nemůžeme skončit v kroku 5 s výstupem „složené“. Dále pro n prvočíslo je $(\mathbb{Z}_n, \oplus, \odot)$ konečné těleso. V tělese existují pouze dva prvky, které umocněné na druhou dávají 1 (tzv. odmocniny z 1) — totiž číslo 1 a -1. Proto nemůžeme skončit ani v kroku 6 výstupem „složené“.

Add 2. Ukázat druhou vlastnost je obtížnější. Důkaz není těžký pro taková složená n, pro která existuje $a \in \mathbb{Z}_n$, a nesoudělné s n, a $a^{n-1} \not\equiv 1 \pmod n$. Pro ostatní složená čísla, tzv. „pseudoprvočísla“, (též „Carmichaelova čísla“), je důkaz dost obtížný.

Ukážeme základní myšlenku důkazu pro složená n: Spočítáme počet takových a vybraných v kroku 2, pro která dostaneme jistě správnou odpověď (tj. nedostaneme odpověď prvočíslo). Protože každé a má stejnou pravděpodobnost být vybráno, stačí, abychom ukázali, že jich je aspoň tolik, kolik jich může dát odpověď špatnou (prvočíslo).

Vybereme-li v kroku 2 neinvertibilní číslo a, určitě dostaneme odpověď složené, protože žádná mocnina neinvertibilního čísla nemůže být rovna 1.

Předpokládejme, že složené číslo n není pseudoprvočíslo, tj. existuje $a \in \mathbb{Z}_n$, a nesoudělné s n, a $a^{n-1} \not\equiv 1 \pmod n$. Označme

$$\mathbb{Z}_n^\star = \{a \in \mathbb{Z}_n \mid a \text{ je invertibilní}\}$$

$$K = \{a \in \mathbb{Z}_n \mid a^{n-1} = 1\}.$$

Víme, že $K \neq \mathbb{Z}_n^\star$, přitom $(K, \odot)$ je podgrupa grupy $(\mathbb{Z}_n^\star, \odot)$. Proto počet prvků K dělí počet prvků $\mathbb{Z}_n^\star$. Odtud počet prvků v množině K je nejvýše dvakrát méně než prvků v množině $\mathbb{Z}_n^\star$; jinými slovy

$$|\mathbb{Z}_n^\star \setminus K| \geq |K|.$$

Vybereme-li $a \in \mathbb{Z}_n^\star \setminus K$, dostaneme správnou odpověď „složené“, protože $a^{n-1} \neq 1$.

Špatnou odpověď můžeme dostat pouze pro $a \in K$ a těch je méně než nebo stejně jako $a \in \mathbb{Z}_n^\star \setminus K$.

Pro pseudoprvočísla platí $|K| = |\mathbb{Z}_n^\star|$ a musíme argumentovat krokem 6, kde se dá ukázat, že počet a, která vedou v kroku 6 na odmocninu z 1 různou od -1 je aspoň tak velký jako počet těch a, která vedou na -1.

4.8 Třídy založené na pravděpodobnostních algoritmech

4.8.1 Randomizovaný Turingův stroj. RTM je, zhruba řečeno, Turingův stroj M se dvěma nebo více páskami, kde první páska má stejnou roli jako u deterministického Turingova stroje, ale druhá páska obsahuje náhodnou posloupnost 0 a 1, tj. na každém políčku se 0 objeví s pravděpodobností $\frac{1}{2}$ a 1 také s pravděpodobností $\frac{1}{2}$.

Na začátku práce:

- stroj M se nachází v počátečním stavu q_0;
- první páska obsahuje vstupní slovo w, zbytek pásky pak blanky B;
- druhá páska obsahuje náhodnou posloupnost 0 a 1;
- případné další pásky obsahují B;
- všechny hlavy jsou nastaveny na prvním políčku dané pásky.

Na základě stavu q, ve kterém se stroj M nachází, a na základě obsahu políček, které jednotlivé hlavy čtou, přechodová funkce δ určuje, zda se M zastaví nebo přejde do nového stavu p, přepíše obsah první pásky (**nikoli ale obsah druhé pásky**) a hlavy posune doprava, doleva nebo zůstanou stát (posuny hlav jsou nezávislé).

Formálně, je-li M ve stavu q, hlava na první pásce čte symbol X, na druhé pásce je číslo a a

$$\delta(q, X, a) = (p, Y, D_1, D_2), \quad q, p \in Q, a \in \{0,1\}, X, Y \in \Gamma, D_1, D_2 \in \{L, R, S\},$$

pak M se přesune do stavu p, na první pásku napíše Y a i-tá hlava se posune doprava pro $D_i = R$, doleva pro $D_i = L$ nebo zůstane na místě pro $D_i = S$.

Jestliže $\delta(q, X, a)$ není definováno, M se zastaví.

M se úspěšně zastaví právě tehdy, když se přesune do koncového (přijímacího) stavu q_f.

4.8.2 Poznámka. Rozdíl mezi RTM a obyčejným TM je v roli druhé pásky. Turingův stroj s dvěma páskami může přepisovat i obsah druhé pásky a to je v případě RTM zakázáno. Navíc při dvou bězích RTM může být průběh práce RTM různý (záleží na náhodně vygenerovaném obsahu druhé pásky). To se u vícepáskového deterministického TM stát nemůže.

Může se zdát, že tento model je nerealistický — nemůžeme před začátkem práce naplnit nekonečnou pásku. Toto je ale „realizováno“ tak, že v okamžiku, kdy druhá hlava čte dosud nenavštívené políčko druhé pásky, náhodně se vygeneruje 0 nebo 1 každé s pravděpodobností $\frac{1}{2}$ a tento symbol už se nikdy během jednoho průběhu práce TM nezmění.

4.8.3 Příklad. Je dán RTM M, kde $Q = \{q_0, q_1, q_2, q_3, q_f\}$, $\Gamma = \{0, 1, B\}$ a přechodová funkce δ je definována tabulkou:

	$0,0$	$1,0$	$0,1$	$1,1$	$B,0$	$B,1$
$\to\ q_0$	$(q_1, 0, R, S)$	$(q_2, 1, R, S)$	$(q_3, 0, S, R)$	$(q_3, 1, S, R)$	–	–
q_1	$(q_1, 0, R, S)$	–	–	–	(q_4, B, S, S)	–
q_2	–	$(q_2, 1, R, S)$	–	–	(q_4, B, S, S)	–
q_3	$(q_3, 0, R, R)$	–	–	$(q_3, 1, R, R)$	(q_4, B, S, S)	(q_4, B, S, S)
$\leftarrow\ q_4$	–	–	–	–	–	–

Předpokládejme, že na vstupu má RTM M slovo w, pak:

- Jestliže první symbol druhé pásky je 0 (tj. náhodně jsme vygenerovali 0), M zkontroluje, zda $w = 0^n$ nebo $w = 1^n$ pro nějaké $n > 0$.
- Jestliže první symbol druhé pásky je 1 (tj. náhodně jsme vygenerovali 1), hlava na druhé pásce se posune doprava a M zkontroluje, zda se obsah druhé pásky od druhého políčka shoduje se vstupem w.

Nenastane-li ani jeden z předchozích případů, M se neúspěšně zastaví.

V případě RTM je třeba spočítat pravděpodobnost s jakou se M pro dané vstupní slovo w úspěšně zastaví, tj. zastaví v „přijímacím“ stavu q_f. V našem příkladě je odpověď tato:

- Jestliže w je prázdné slovo, M se v q_f nikdy nezastaví (tj. pro žádný náhodný obsah druhé pásky).
- Jestliže $w = 0^n$ nebo $w = 1^n$ pro $n > 0$, M se zastaví v q_f s pravděpodobností
$$\frac{1}{2} + \frac{1}{2}\left(\frac{1}{2}\right)^n = \frac{1}{2} + 2^{-(n+1)}.$$
- Jestliže w je jiného tvaru, tj. obsahuje jak 0, tak 1, pak pravděpodobnost, že se M zastaví v q_f je
$$\frac{1}{2}\left(\frac{1}{2}\right)^{|w|} = 2^{-(|w|+1)}.$$

4.8.4 Třída $\mathcal{RP}$. Jazyk L patří do třídy $\mathcal{RP}$ právě tehdy, když existuje RTM M takový, že:

1. Jestliže $w \notin L$, stroj M se ve stavu q_f zastaví s pravděpodobností 0.
2. Jestliže $w \in L$, stroj M se ve stavu q_f zastaví s pravděpodobností, která je alespoň rovna $\frac{1}{2}$.
3. Existuje polynom $p(n)$ takový, že každý běh M (tj. pro jakýkoli obsah druhé pásky) trvá maximálně $p(n)$ kroků, kde n je délka vstupního slova.

Miller-Rabinův test prvočíselnosti je příklad algoritmu, který splňuje všechny tří podmínky (utvoříme-li k němu odpovídající RTM) a proto jazyk L, který se skládá ze všech složených čísel, patří do třídy $\mathcal{RP}$.

4.8.5 Turingův stroj typu Monte-Carlo. RTM splňující podmínky 1 a 2 z předchozí definice 4.8.4, se nazývá RTM typu *Monte-Carlo*.

Uvědomte si, že RTM typu Monte-Carlo obecně nemusí pracovat v polynomiálním čase.

4.8.6 Tvrzení. Je dán jazyk $L \in \mathcal{RP}$, pak pro každou kladnou konstantu $0 < c < \frac{1}{2}$ je možné sestrojit RTM M (algoritmus) s polynomiální složitostí a takový, že:

1. Jestliže $w \notin L$, stroj M se úspěšně zastaví (tj. zastaví se ve stavu q_f) s pravděpodobností 0.
2. Jestliže $w \in L$, stroj M se úspěšně zastaví (tj. zastaví se ve stavu q_f) s pravděpodobností aspoň $1 - c$.

4.8.7 Třída $\mathcal{ZPP}$. Jazyk L patří do třídy $\mathcal{ZPP}$ právě tehdy, když existuje RTM M takový, že:

1. Jestliže $w \notin L$, stroj M se úspěšně zastaví (tj. zastaví se ve stavu q_f) s pravděpodobností 0.
2. Jestliže $w \in L$, stroj M se úspěšně zastaví (tj. zastaví se ve stavu q_f) s pravděpodobností 1.
3. Střední hodnota počtu kroků M v jednom běhu je $p(n)$, kde $p(n)$ je polynom a n je délka vstupního slova.

To znamená: M neudělá chybu, ale nezaručujeme vždy polynomiální počet kroků při jednom běhu, pouze střední hodnota počtu kroků je polynomiální.

4.8.8 Turingův stroj typu Las-Vegas. RTM splňující podmínky z předchozí definice 4.8.7, se nazývá typu *Las-Vegas*.

4.8.9 Tvrzení. Jestliže jazyk L patří do třídy $\mathcal{ZPP}$, pak i jeho doplněk $\overline{L}$ patří do třídy $\mathcal{ZPP}$.

Stejný RTM M typu Las-Vegas slouží „k přijetí“ jak jazyka L, tak i jeho doplňku $\overline{L}$; stačí koncové (přijímající) stavy RTM M prohlásit za nekoncové a ze všech nekoncových stavů M udělat koncové.

4.8.10 Poznámka. Pro jazyky ze třídy $\mathcal{RP}$ se tvrzení obdobné 4.8.9 neumí dokázat. To motivuje následující třídu jazyků.

4.8.11 Třída co-$\mathcal{RP}$. Jazyk L patří do třídy co-$\mathcal{RP}$ právě tehdy, když jeho doplněk $\overline{L}$ patří do třídy $\mathcal{RP}$.

4.8.12 Věta.

$$\mathcal{ZPP} = \mathcal{RP} \cap \text{ co-}\mathcal{RP}.$$

Nástin důkazu. Ukážeme nejprve $\mathcal{RP} \cap \text{ co-}\mathcal{RP} \subseteq \mathcal{ZPP}$.

Předpokládejme, že jazyk L leží v obou třídách $\mathcal{RP}$ i co-$\mathcal{RP}$. Existují proto dva RTM M_1 a M_2 typu Monte Carlo pracující v polynomiálním čase a takové, že

M_1 — pro jazyk L;

M_2 — pro jazyk $\overline{L}$.

Označme $p(n)$ ten větší z polynomů, které určují počet kroků M_1 a M_2. Sestrojíme RTM M typu Las-Vegas pro jazyk L takto: Pro dané vstupní slovo w

1. M nechá pracovat M_1 po dobu $p(n)$ kroků. Jestliže M_1 úspěšně skončí, M také skončí úspěšně.
2. M nechá pracovat M_2 po dobu $p(n)$ kroků. Jestliže M_2 úspěšně skončí, M skončí ale neúspěšně.
3. Jestliže M neskončí ani v kroku 1 ani v kroku 2, M pokračuje opět krokem 1.

Dá se dokázat, že RTM M je typu Las-Vegas.

Nyní ukážeme, že $\mathcal{ZPP} \subseteq \mathcal{RP} \cap \text{ co-}\mathcal{RP}$.

Předpokládejme, že jazyk L leží ve třídě $\mathcal{ZPP}$, existuje tedy pro něj RTM M_1 typu Las-Vegas. Označme $p(n)$ polynom, který udává střední hodnotu počtu kroků RTM M_1 pro vstupní slovo délky n. Vytvoříme RTM M typu Monte Carlo pracující polynomiálním čase pro jazyk L.

M nechá na vstupu w pracovat RTM M_1 po dobu $2p(n)$. Jestliže M_1 úspěšně skončí, M úspěšně skončí; ve všech ostatních případech RTM M skončí neúspěšně.

Dá se dokázat, že M splňuje všechny podmínky pro RTM typu Monte Carlo. Protože pracuje v čase $2p(n)$, jedná se o polynomiální RTM typu Monte Carlo. Proto je jazyk L ve třídě $\mathcal{RP}$.

Protože třída $\mathcal{ZPP}$ je uzavřena na doplňky, je každý jazyk ze třídy $\mathcal{ZPP}$ také ve třídě co-$\mathcal{RP}$.

4.8.13 Věta. Platí

$$\mathcal{P} \subseteq \mathcal{ZPP}, \ \ \mathcal{RP} \subseteq \mathcal{NP}, \ \ \text{co-}\mathcal{RP} \subseteq \text{ co-}\mathcal{NP}.$$

První inkluze je zřejmá, každý polynomiální Turingův stroj můžeme považovat za randomizovaný Turingův stroj typu Las-Vegas.

Druhá inkluze je složitější. Její důkaz spočívá v tom, že pro daný polynomiální RTM M typu Monte Carlo pracující v polynomiálním čase zkonstruujeme nedeterministický Turingův stroj, který přijímá jazyk $L(M)$.

Třetí inkluze jednoduše vyplývá z definic tříd co-$\mathcal{RP}$, co-$\mathcal{NP}$ a z druhé inkluze.

4.9 Nerozhodnutelnost

4.9.1 Rekursivní jazyky. Řekneme, že jazyk L je *rekursivní*, jestliže existuje Turingův stroj M, který rozhoduje jazyk L.

Připomeňme, že Turingův stroj M rozhoduje jazyk L znamená, že jej přijímá a na každém vstupu se zastaví (buď úspěšně nebo neúspěšně).

Třída rekursivních jazyků se často značí R.

4.9.2 Rekursivně spočetné jazyky. Řekneme, že jazyk L je *rekursivně spočetný*, jestliže existuje Turingův stroj M, který tento jazyk přijímá.

Jinými slovy, M se pro každé slovo w, které patří do L, úspěšně zastaví a pro slovo w, které nepatří do L se buď zastaví neúspěšně nebo se nezastaví vůbec.

Třída rekursivně spočetných jazyků se často značí RS.

4.9.3 Poznámka. Jazykům, které nejsou rekursivní, také říkáme, že jsou *algoritmicky neřešitelné* nebo *nerozhodnutelné*. Obdobně mluvíme o úlohách, které jsou nerozhodnutelné nebo algoritmicky neřešitelné. První pojem se užívá častěji pro rozhodovací úlohy, druhý i pro úlohy konstrukční či optimalizační.

Každý rekursivní jazyk je též rekursivně spočetný. V dalším textu ukážeme, že naopak to neplatí, tj. existují rekursivně spočetné jazyky, které nejsou rekursivní.

4.9.4 Tvrzení. Jestliže jazyk L je rekursivní, pak je rekursivní i jeho doplněk $\overline{L}$.

4.9.5 Tvrzení. Jestliže jazyk L i jeho doplněk $\overline{L}$ jsou oba rekursivně spočetné, pak L je rekursivní.

4.9.6 Tvrzení. Pro jazyk L může nastat jedna z následujících možností:

1. L i $\overline{L}$ jsou oba rekursivní.
2. Jeden z L a $\overline{L}$ je rekursivně spočetný a druhý není rekursivně spočetný.
3. L i $\overline{L}$ nejsou rekursivně spočetné.

4.9.7 Kód Turingova stroje. Každý Turingův stroj M lze zakódovat jako binární slovo. Mějme Turingův stroj M s množinou stavů $Q = \{q_1, q_2, \ldots, q_n\}$, množinou vstupních symbolů $\Sigma = \{0, 1\}$, množinou páskových symbolů $\Gamma = \{X_1, X_2, \ldots, X_m\}$, kde $X_1 = 0$, $X_2 = 1$ a $X_3 = B$. Dále počáteční stav je stav q_1, koncový stav je q_2. Označme D_1 pohyb hlavy doprava a D_2 pohyb hlavy doleva. (Tj. $D_1 = R$ a $D_2 = L$.)

Jeden přechod stroje M

$$\delta(q_i, X_j) = (q_k, X_l, D_r)$$

zakódujeme slovem

$$w = 0^i 1 0^j 1 0^k 1 0^l 1 0^r.$$

které nazýváme *Kód Turingova stroje* M, značíme jej $\langle M \rangle$, je

$$\langle M \rangle = 111\, w_1\, 11\, w_2\, 11 \ldots 11\, w_p\, 111,$$

Kde $w_1, \ldots, w_p$ jsou slova odpovídající všem přechodům stroje M.

4.9.8 Binární slova můžeme uspořádat do posloupnosti a tudíž je očíslovat. Jedno z možných očíslování je toto: K binárnímu slovu w utvoříme $1w$ a toto chápeme jako binární zápis přirozeného čísla.

Tedy např. ϵ je první slovo, 0 je druhé slovo, 1 je třetí slovo, atd, 100110 je 1100110 = 64 + 32 + 4 + 2 = 102, tj. 100110 je 102-hé slovo. V dalším textu o binárním slovu na místě i mluvíme jako o slovu w_i. Tedy $w_1 = \epsilon$, $w_{102} = 100110$.

Jedná se vlastně o uspořádání slov nejprve podle délky a mezi slovy stejné délky o lexikografické uspořádání.

4.9.9 Diagonální jazyk L_d. Nejprve uděláme následující úmluvu. Jestliže binární slovo w nemá tvar z 4.9.7, považujeme ho za kód Turingova stroje M, který nepřijímá žádné slovo (neudělá nikdy žádný krok). Tj. $L(M) = \emptyset$.

Jazyk L_d se skládá ze všech binárních slov w takových, že Turingův stroj s kódem w nepřijímá slovo w. (Tedy L_d obsahuje i všechna slova w, která neodpovídají kódům nějakého Turingova stroje, ovšem obsahuje i další binární slova.)

4.9.10 Věta. Neexistuje Turingův stroj, který by přijímal jazyk L_d. Jinými slovy, $L_d \neq L(M)$ pro každý Turingův stroj M.

Nástin důkazu. Postupujeme sporem. Kdyby existoval Turingův stroj M takový, že $L_d = L(M)$, pak by tento Turingův stroj měl kód roven nějakému binárnímu slovu, tj. $\langle M\rangle = w_i$ pro nějaké i.

Na otázku, zda toto slovo w_i patří nebo nepatří do jazyka L_d, nemůžeme dát odpověď, která by nevedla ke sporu.

Kdyby $w_i \in L_d$, pak w_i splňuje podmínku: Turingův stroj s kódem w_i nepřijímá slovo w_i. Ale $L_d = L(M)$ kde $w_i = \langle M\rangle$ — spor.

Kdyby $w_i \notin L_d$, pak Turingův stroj s kódem w_i přijímá slovo w_i. Ale to je podmínka pro to, aby slovo w_i patřilo do L_d — spor.

Proto neexistuje Turingův stroj, který by přijímal jazyk L_d.

4.9.11 Univerzální jazyk. *Univerzální jazyk* L_{UN} je množina slov tvaru $\langle M\rangle\, w$, kde $\langle M\rangle$ je kód Turingova stroje a $w \in \{0,1\}^\star$ je binární slovo takové, že $w \in L(M)$.

4.9.12 Univerzální Turingův stroj. Popíšeme, velmi zhruba, Turingův stroj, který přijímá univerzální jazyk L_{UN}. Tomuto Turingovu stroji se také říká *univerzální Turingův stroj* a značíme ho U.

Univerzální Turingův stroj U má 4 pásky. První páska obsahuje vstupní slovo $\langle M\rangle\, w$, druhá páska simuluje pásku Turingova stroje M a třetí páska obsahuje kód stavu, ve kterém se Turingův stroj M nachází. Dále má U ještě čtvrtou, pomocnou pásku.

Na začátku práce Turingova stroje U je na první pásce vstupní slovo $\langle M\rangle\, w$, ostatní pásky obsahují pouze B, blanky. Připomeňme, že kód Turingova stroje získáme takto. Předpokládejme, že Turingův stroj M se skládá z $(Q, \{0,1\}, \{0,1,B\}, \delta, q_1, \{q_2\})$, kde $Q = \{q_1, q_2, \dots, q_n\}$. Označme 0 jako X_1, 1 jako X_2, B jako X_3, pohyb doprava R jako D_1, pohyb doleva L jako D_2. Pak jednotlivé přechody $\delta(q_i, X_j) = (q_k, X_l, D_m)$ kódujeme

$$t = 0^i 1 0^j 1 0^k 1 0^l 1 0^m, \quad \text{kde} \ \ 1 \leq i, k \leq n, 1 \leq j, l \leq 3, 1 \leq m \leq 2.$$

Turingův stroj M má kód

$$111\, t_1\, 11\, t_2\, 11 \dots 11\, t_r\, 111.$$

Turingův stroj U nejprve zkontroluje, že vstup je opravdu kódem Turingova stroje M následovaný binárním slovem. Jestliže není, U se neúspěšně zastaví.

V případě, že vstupní slovo je tvaru kód Turingova stroje M následovaný binárním slovem w, U přepíše slovo w na druhou pásku a na třetí pásku napíše 0. To je proto, že Turingův stroj je na začátku práce ve stavu q_1 kódovaném jako 0.

Nyní Turingův stroj U simuluje kroky Turingova stroje M s tím, že kdykoli se stroj M dostane do stavu q_2 (koncový „přijímací“ stav M), U se úspěšně zastaví. Toto poznáme tak, že na třetí pásce se objeví 00 předcházené a následované B, blanky.

Poznamenejme, že je třeba ještě řada dalších technických detailů. Např. při přepisování slova w na druhou pásku to děláme tak, že za 0 ve vstupním slově w na pásku napíšeme 10, za 1 ve w na druhou pásku zapíšeme 100. Je-li na druhou pásku potřeba (vzhledem k přechodové funkci Turingova stroje M) na druhou pásku napsat B, napíšeme 1000. Čtvrtá páska slouží k tomu, abychom na druhou pásku byli schopni vždy napsat stav pásky TM M.

4.9.13 Důsledek. Univerzální jazyk L_{UN} je rekursivně spočetný.

4.9.14 Tvrzení. Univerzální jazyk L_{UN} není rekursivní.

Kdyby totiž L_{UN} byl rekursivní, existoval by Turingův stroj M, který rozhodne L_{UN}. Tj. M se vždy zastaví; na slovech z jazyka L_{UN} se úspěšně zastaví, na slovech neležících v L_{UN} se neúspěšně zastaví. Na základě tohoto Turingova stroje M bychom byli schopni rozhodnout diagonální jazyk L_d, o kterém víme, že není ani rekursivně spočetný, viz 4.9.10.

4.9.15 Redukce. Připomeňme definici redukce z 4.3.1.

Jsou dány dvě rozhodovací úlohy $\mathcal{U}$ a $\mathcal{V}$. Řekneme, že úloha $\mathcal{U}$ se *redukuje* na úlohu $\mathcal{V}$, jestliže existuje algoritmus (program pro RAM, Turingův stroj) $\mathcal{A}$, který pro každou instanci I úlohy $\mathcal{U}$ zkonstruuje instanci I' úlohy $\mathcal{V}$ a to tak, že

$$I \text{ je ANO instance } \mathcal{U} \text{ iff } I' \text{ je ANO instance } \mathcal{V}.$$

Fakt, že úloha $\mathcal{U}$ se redukuje na úlohu $\mathcal{V}$ značíme

$$\mathcal{U} \lhd \mathcal{V}.$$

Jsou dány dva jazyky $L_1 \subseteq \Sigma^\star$, $L_2 \subseteq \Gamma^\star$. Řekneme, že jazyk L_1 se *redukuje* na jazyk L_2, jestliže existuje algoritmus (program pro RAM, Turingův stroj) $\mathcal{A}$, který pro každé slovo $w \in \Sigma^\star$ zkonstruuje slovo $A(w) \in \Gamma^\star$ a to tak, že

$$w \in L_1 \text{ iff } A(w) \in L_2.$$

Fakt, že jazyk L_1 se redukuje na jazyk L_2 značíme

$$L_1 \lhd L_2.$$

4.9.16 Tvrzení. Jsou dány dvě úlohy $\mathcal{U}$ a $\mathcal{V}$ takové, že $\mathcal{U} \lhd \mathcal{V}$. Pak platí:

1. Jestliže $\mathcal{V}$ je rozhodnutelná, pak i $\mathcal{U}$ je rozhodnutelná.
2. Jestliže $\mathcal{U}$ je nerozhodnutelná, pak i $\mathcal{V}$ je nerozhodnutelná.
3. Jestliže jazyk úlohy $\mathcal{U}$ není rekursivně spočetný, pak i jazyk úlohy $\mathcal{V}$ není rekursivně spočetný.

4.9.17 Tvrzení. Jsou dány jazyky

$$L_e = \{M \mid L(M) = \emptyset\}, \quad L_{ne} = \{M \mid L(M) \neq \emptyset\}.$$

Pak jazyk L_{ne} je rekursivně spočetný, ale ne rekursivní. Jakyk L_e není ani rekursivně spočetný.

4.9.18 Poznámka. Uvědomme si, že jazyk L_e je doplňkem jazyka L_{ne}. Ano, jestliže slovo w není kódem nějakého Turingova stroje, pak ho považujeme za kód stroje, který nepřijímá žádné slovo, tj. patří do jazyka L_e.

Univerzální Turingův stroj U se dá využít i k tomu abychom ukázali, že jazyk L_{ne} je rekursivně spočetný. Z redukce $L_{UN} \lhd L_{ne}$ a 4.9.16 dostáváme, že L_{ne} není rekursivní. Fakt, že L_e není ani rekursivně spočetný pak vyplývá z 4.9.5.

4.9.19 Věta (Rice). Jakákoli netriviální vlastnost rekursivně spočetných jazyků (jazyků přijímaných Turingovým strojem) je nerozhodnutelná.

Poznamenejme, že netriviální vlastností se rozumí každá vlastnost, kterou má aspoň jeden rekursivně spočetný jazyk a nemají ho všechny rekursivně spočetné jazyky.

4.10 Další nerozhodnutelné úlohy

4.10.1 V minulém oddíle jsme uvedli několik nerozhodnutelných jazyků — úloh. Věta (Rice) dokonce říká, že každá netriviální vlastnost rekursivních jazyků je nerozhodnutelná. Na druhou stranu úlohy týkající se rekursivních jazyků se mohou zdát jako značně umělé. V této části ukážeme další úlohy, které jsou nerozhodnutelné. Poznamenejme ještě, že univerzální jazyk L_{UN} hraje pro nerozhodnutelné jazyky/úlohy obdobnou roli jako hrál problém splnitelnosti booleovských formulí pro $\mathcal{NP}$ úplné úlohy.

Označme $\mathcal{UN}$ úlohu odpovídající univerzálnímu jazyku L_{UN}; tj. tuto úlohu: Instance se skládá z TM M a slova w. Jedná se o ano instanci právě tehdy, když $w \in L(M)$.

4.10.2 Postův korespondenční problém (PCP). Jsou dány dva seznamy slov A, B nad danou abecedou Σ.

$$A = (w_1, w_2, \ldots, w_k), \quad B = (x_1, x_2, \ldots, x_k),$$

kde $w_i, x_i \in \Sigma^\star$, $i = 1, 2, \ldots, k$. Řekneme, že dvojice A, B *má řešení*, jestliže existuje posloupnost $i_1, i_2, \ldots, i_r$ indexů, tj $i_j \in \{1, 2, \ldots, k\}$, taková, že

$$w_{i_1} w_{i_2} \ldots w_{i_r} = x_{i_1} x_{i_2} \ldots x_{i_r}.$$

Otázka: Existuje řešení dané instance?

4.10.3 Příklady.

1. Jsou dány seznamy

	1	2	3	4	5
A	011	0	101	1010	010
B	1101	00	01	00	0

   Tato instance má řešení, např. $2, 1, 1, 4, 1, 5$ je

$$w_2 w_1 w_1 w_4 w_1 w_5 = 00110111010011010 = x_2 x_1 x_1 x_4 x_1 x_5.$$

2. Jsou dány seznamy

	1	2	3	4	5
A	11	0	101	1010	010
B	101	00	01	00	0

   Tato instance nemá řešení.

4.10.4 Modifikovaný Postův korespondenční problém (MPCP). Jsou dány dva seznamy slov A, B nad danou abecedou Σ.

$$A = (w_1, w_2, \ldots, w_k), \quad B = (x_1, x_2, \ldots, x_k),$$

kde $w_i, x_i \in \Sigma^\star$, $i = 1, 2, \ldots, k$. Řekneme, že dvojice A, B *má řešení*, jestliže existuje posloupnost $1, i_1, i_2, \ldots, i_r$ indexů, tj $i_j \in \{1, 2, \ldots, k\}$, taková, že

$$w_1 w_{i_1} w_{i_2} \ldots w_{i_r} = x_1 x_{i_1} x_{i_2} \ldots x_{i_r}.$$

Otázka: Existuje řešení dané instance?

4.10.5 Poznámka. Modifikovaný Postův korespondenční problém se od Postova korespondenčního problému liší tím, že v MPCP vyžadujeme, aby hledaná posloupnost indexů vždy začínala jedničkou. Význam MPCP spočívá v tom, že se dá dokázat následující věta.

4.10.6 Věta. Platí
$$\mathcal{UN} \lhd \text{MPCP} \lhd \text{PCP}.$$

Nástin druhé redukce. Máme dánu instanci MPCP
$$A = (w_1, w_2, \ldots, w_k), \quad B = (x_1, x_2, \ldots, x_k),$$
Předpokládejme, že # a $*$ nejsou prvky Σ, Vytvoříme novou instanci PCP
$$C = (y_0, y_1, \ldots, y_k, y_{k+1}), \quad D = (z_0, z_1, \ldots, z_k, z_{k+1}),$$
kde

1. Pro každé $i = 1, \ldots, k$ slovo y_i vzniklo ze slova w_i tím, že jsme **za** každý symbol slova w_i umístili symbol $*$; obdobně z_i vzniklo ze slova x_i přidáním symbolu $*$ **před** každý symbol slova x_i.
2. $y_0 = *y_1$; $z_0 = z_1$.
3. $y_{k+1} = *\#$, $z_{k+1} = \#$.

Není těžké nahlédnout, že A, B má řešení $1, i_1, \ldots, i_r$ právě tehdy, když má řešení C, D a to musí být $0, i_1, \ldots, i_r, k+1$.

4.10.7 Poznámka. První redukce je obtížnější. Jedná se o popis práce Turingova stroje pomocí slov nad vhodnou abecedou. Trik spočívá v tom, že posloupnost pro MPCP musí začínat prvním slovem (to zajistí, že Turingův stroj začne pracovat v počátečním stavu s daným obsahem pásky). Pro seznam A bude slovo vždy „dohánět výpočet podle přechodové funkce Turingova stroje“, který bude odpovídat seznamu B. Bude tedy slovo vytvořené podle seznamu A prefixem slova vytvořeného podle seznamu B. Slova se stanou stejnými teprve v okamžiku, kdy se TM dostaneme do koncového stavu; tj. kdy se ve slově podle seznamu B objeví koncový stav.

4.10.8 Důsledek. Postův korespondenční problém je nerozhodnutelný.

4.10.9 Poznámka. Kdybychom omezili možnou délku hledané posloupnosti $i_1, i_2, \ldots, i_r$, (tj. omezili r), problém by se stal algoritmicky řešitelným — existoval by algoritmus hrubé síly. Také, kdybychom místo seznamů A, B uvažovali množiny slov, problém by byl dokonce polynomiálně řešitelný.

4.10.10 Víceznačnost bezkontextových gramatik. Je dána bezkontextová gramatika $\mathcal{G} = (N, \Sigma, S, P)$, kde N je množina neterminálních symbolů, Σ je množina terminálních symbolů, S je startovací symbol a P je množina pravidel typu $X \to \alpha$ pro $X \in N$, $\alpha \in (N \cup \Sigma)^\star$.

Otázka: Rozhodněte, zda existuje slovo w, které má dva různé derivační stromy.

4.10.11 Věta. Platí
$$\text{PCP} \lhd \text{víceznačnost bezkontextových gramatik.}$$

4.10.12 Nástin redukce pro důkaz věty 4.10.11. Je dána instance PCP, tj. seznamy slov $A = (w_1, w_2, \ldots, w_k)$ a $B = (x_1, x_2, \ldots, x_k)$. Sestrojíme bezkontextovou gramatiku $\mathcal{G} = (\{S, A, B\}, \Sigma \cup \{a_1, a_2, \ldots, a_k\}, S, P)$, kde P obsahuje tato pravidla
$$S \to A \mid B,$$
$$A \to w_1\, A\, a_1 \mid w_2\, A\, a_2 \mid \ldots \mid w_k\, A\, a_k,$$
$$A \to w_1\, a_1 \mid w_2\, a_2 \mid \ldots \mid w_k\, a_k,$$
$$B \to x_1\, B\, a_1 \mid x_2\, B\, a_2 \mid \ldots \mid x_k\, B\, a_k,$$

$$B \to x_1\, a_1 \mid x_2\, a_2 \mid \ldots | x_k\, a_k,$$

Pak gramatika $\mathcal{G}$ je víceznačná právě tehdy, když nějaké slovo $wa_{i_1}a_{i_2}\ldots a_{i_r}$, $w \in \Sigma^\star$, má dvě různá odvození. Tato situace nastává právě tehdy, když instance PCP má řešení. (Uvědomte si, že dvě různá odvození jsou možná jen, můžeme-li stejné slovo $wa_{i_1}a_{i_2}\ldots a_{i_r}$ odvodit při použití pravidla $S \to A$ i $S \to B$, tedy w vytvořit ze seznamu A i ze seznamu B při použití slov se stejným indexem.)

4.10.13 Věta. Jsou dány bezkontextové gramatiky $\mathcal{G}_1$ a $\mathcal{G}_2$. Označme $L(\mathcal{G}_1)$ a $L(\mathcal{G}_2)$ jazyky generované gramatikami $\mathcal{G}_1$ a $\mathcal{G}_2$. Následující úlohy jsou nerozhodnutelné.

1. $L(\mathcal{G}_1) \cap L(\mathcal{G}_2) = \emptyset$.
2. $L(\mathcal{G}_1) = L(\mathcal{G}_2)$.
3. $L(\mathcal{G}_1) \subseteq L(\mathcal{G}_2)$.
4. $L(\mathcal{G}_1) = \Sigma^\star$.

4.10.14 Tiling problém. Jsou dány čtvercové dlaždičky velikosti 1 cm^2 několika typů. Každá dlaždička má barevné okraje. Máme neomezený počet dlaždiček každého typu.

Otázka: Je možné dlaždičkami vydláždit každou plochu daného typu tak, aby se dlaždičky dotýkaly hranami stejné barvy, za předpokladu, že dlaždičky nesmíme rotovat?

4.10.15 Věta. Tiling problém je nerozhodnutelný.

Tedy speciálně je nerozhodnutelné, zda každou neomezenou plochu je možné vydláždit předem danou sadou dlaždiček.