Kapitola 1

Úvod

1.1 Základní pojmy

1.1.1 Algoritmus. *Algoritmem* rozumíme dobře definovaný proces, tj. posloupnost výpočetních kroků, který přijímá hodnoty (zadání, vstup) a vytváří hodnoty (řešení, výstup).

1.1.2 Problém, úloha. *Úloha*, též *problém*, je obecná specifikace vztahu zadání/řešení. *Instancí* problému, úlohy $\mathcal{U}$ rozumíme konkrétní zadání všech parametrů, které daná úloha (problém) obsahuje. Jinými slovy, instance úlohy je správný příklad zadání.

1.1.3 Řekneme, že algoritmus $\mathcal{A}$ *řeší* úlohu $\mathcal{U}$, jestliže pro každý vstup (každou instanci problému $\mathcal{U}$) vydá správné řešení.

Poznamenejme, že předchozí věta znamená, že každý algoritmus, který řeší nějakou úlohu, se vždy zastaví. To znamená, že algoritmus, který se na nějakém vstupu nezastaví, nemůže řešit žádnou úlohu.

1.1.4 Analýza časové složitosti algoritmu. Existují dva základní způsoby měření časové náročnosti algoritmů.

1. Analýza nejhoršího případu. Jedná se o asymptotický odhad $T(n)$ času potřebného pro vyřešení každé instance velikosti n.
2. Průměrná složitost. Jedná se o asymptotický odhad $T_{aver}(n)$ průměrného času, který je potřeba pro vyřešení instance velikosti n.

1.2 Asymptotický růst funkcí

1.2.1 Symbol $\mathcal{O}$. Je dána nezáporná funkce $g(n)$. Řekneme, že nezáporná funkce $f(n)$ *je* $\mathcal{O}(g(n))$, jestliže existuje kladná konstanta c a přirozené číslo n_0 tak, že

$$f(n) \leq c\, g(n) \quad \text{pro všechny } n \geq n_0.$$

$\mathcal{O}(g(n))$ můžeme též chápat jako třídu všech nezáporných funkcí $f(n)$:

$$\mathcal{O}(g(n)) = \{f(n) \mid \exists c > 0, n_0 \text{ tak, že } f(n) \leq c\, g(n) \;\; \forall n \geq n_0\}.$$

1.2.2 Symbol Ω. Je dána nezáporná funkce $g(n)$. Řekneme, že nezáporná funkce $f(n)$ *je* $\Omega(g(n))$, jestliže existuje kladná konstanta c a přirozené číslo n_0 tak, že

$$f(n) \geq c\, g(n) \quad \text{pro všechny } n \geq n_0.$$

$\Omega(g(n))$ můžeme též chápat jako třídu všech nezáporných funkcí $f(n)$:

$$\Omega(g(n)) = \{f(n) \mid \exists c > 0, n_0 \text{ tak, že } f(n) \geq c\, g(n) \ \ \forall\, n \geq n_0\}.$$

1.2.3 Poznámka. Fakt, že funkce $f(n)$ je $\Omega(g(n))$ je ekvivalentní faktu, že funkce $g(n)$ je $\mathcal{O}(f(n))$.

1.2.4 Symbol Θ. Je dána nezáporná funkce $g(n)$. Řekneme, že nezáporná funkce $f(n)$ *je* $\Theta(g(n))$, jestliže existují kladné konstanty c_1, c_2 a přirozené číslo n_0 tak, že

$$c_1\, g(n) \leq f(n) \leq c_2\, g(n) \quad \text{pro všechny } n \geq n_0.$$

$\Theta(g(n))$ můžeme též chápat jako třídu všech nezáporných funkcí $f(n)$:

$$\Theta(g(n)) = \{f(n) \mid \exists c_1, c_2 > 0, n_0 \text{ tak, že } c_1\, g(n) \leq f(n) \leq c_2\, g(n) \ \ \forall\, n \geq n_0\}.$$

1.2.5 Poznámka. Platí $f(n)$ je $\Theta(g(n))$ právě tehdy, když $f(n)$ je zároveň $\mathcal{O}(g(n))$ a $\Omega(g(n))$.

1.2.6 Symbol malé o. Je dána nezáporná funkce $g(n)$. Řekneme, že nezáporná funkce $f(n)$ *je* $o(g(n))$, jestliže pro každou kladnou konstantu c existuje přirozené číslo n_0 tak, že

$$0 \leq f(n) < c\, g(n) \quad \text{pro všechny } n \geq n_0.$$

$o(g(n))$ můžeme též chápat jako třídu všech nezáporných funkcí $f(n)$:

$$o(g(n)) = \{f(n) \mid \forall\, c > 0\ \exists\, n_0 \text{ tak, že } 0 \leq f(n) < c\, g(n) \ \ \forall\, n > n_0\}.$$

1.2.7 Poznámka. Fakt, že nezáporná funkce $f(n)$ je $\mathcal{O}(g(n))$, zhruba řečeno znamená, že funkce $f(n)$ neroste asymptoticky více než funkce $g(n)$. Naproti tomu fakt, že nezáporná funkce $f(n)$ je $o(g(n))$, znamená, že funkce $f(n)$ roste asymptoticky méně než funkce $g(n)$.

1.2.8 Symbol malé ω. Je dána nezáporná funkce $g(n)$. Řekneme, že nezáporná funkce $f(n)$ *je* $\omega(g(n))$, jestliže pro každou kladnou konstantu c existuje přirozené číslo n_0 tak, že

$$0 \leq c\, g(n) < f(n) \quad \text{pro všechny } n \geq n_0.$$

$\omega(g(n))$ můžeme též chápat jako třídu všech nezáporných funkcí $f(n)$:

$$\omega(g(n)) = \{f(n) \mid \forall\, c > 0\ \exists\, n_0 \text{ tak, že } 0 \leq c\, g(n) < f(n) \ \ \forall\, n > n_0\}.$$

1.2.9 Poznámka. Fakt, že nezáporná funkce $f(n)$ je $\Omega(g(n))$, zhruba řečeno znamená, že funkce $f(n)$ roste asymptoticky alespoň tak, jako funkce $g(n)$. Naproti tomu fakt, že nezáporná funkce $f(n)$ je $\omega(g(n))$, znamená, že funkce $f(n)$ roste asymptoticky více než funkce $g(n)$.

1.2.10 Značení. Přestože symboly $\mathcal{O}, \Omega, \Theta$ představují množiny funkcí, budeme v dalším textu nepřesně psát $f(n) = \mathcal{O}(g(n))$ místo přesnějšího $f(n) \in \in \mathcal{O}(g(n))$. Je ovšem třeba mít na paměti, že znak rovnosti v takovém zápise $f(n) = \mathcal{O}(g(n))$ nemá stejné vlastnosti jako klasická rovnost. Obdobně pro ostatní symboly.

1.2.11 Tvrzení. Jsou dány dvě nezáporné funkce $f(n)$ a $g(n)$. Existuje-li $\lim_{n\to\infty} \frac{f(n)}{g(n)}$, pak

- $f(n) = o(g(n))$, jestliže $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$;
- $f(n) = \omega(g(n))$, jestliže $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$.

1.2.12 Tranzitivita. Máme dány tři nezáporné funkce $f(n)$, $g(n)$ a $h(n)$.

1. Jestliže $f(n) = \mathcal{O}(g(n))$ a $g(n) = \mathcal{O}(h(n))$, pak $f(n) = \mathcal{O}(h(n))$.
2. Jestliže $f(n) = \Omega(g(n))$ a $g(n) = \Omega(h(n))$, pak $f(n) = \Omega(h(n))$.
3. Jestliže $f(n) = \Theta(g(n))$ a $g(n) = \Theta(h(n))$, pak $f(n) = \Theta(h(n))$.

1.2.13 Reflexivita. Pro všechny nezáporné funkce $f(n)$ platí: $f(n) = \mathcal{O}(f(n))$, $f(n) = \Omega(f(n))$ a $f(n) = \Theta(f(n))$.

1.2.14 Tvrzení. $f(n) = \Theta(g(n))$ právě tehdy, když $g(n) = \Theta(f(n))$.

1.2.15 Příklady.

1. Pro každé $a > 1$ a $b > 1$ platí
$$\log_a(n) = \Theta(\log_b(n)).$$
2. Platí
$$\lg n! = \Theta(n \lg n).$$

1.3 Řešení rekursivních vztahů

1.3.1 Věta. Jsou dána čísla $a > 0$, $c > 0$ a přirozené číslo b. Předpokládejme, že funkce $T\colon \mathbb{N} \to \mathbb{N}$ splňuje

1. $T(1) = c$;
2. pro $n = b^k$ platí
$$T(n) = a\,T(\frac{n}{b}) + c\,n;$$
3. pro každé n, pro které $b^k \le n \le b^{k+1}$, platí $T(b^k) \le T(n) \le T^(b^{k+1})$.

Potom

1. je-li $a < b$, pak $T(n) = \mathcal{O}(n)$;
2. je-li $a = b$, pak $T(n) = \mathcal{O}(n \lg n)$;
3. je-li $a > b$, pak $T(n) = \mathcal{O}(n^{\log_b a})$.

1.3.2 Věta — „Master Theorem“. Jsou dána přirozená čísla $a \geq 1$, $b > 1$ a funkce $f(n)$. Předpokládejme, že funkce $T(n)$ je dána na přirozených číslech rekurentním vztahem

$$T(n) = a\,T(\frac{n}{b}) + f(n),$$

kde $\frac{n}{b}$ znamená buď $\lfloor \frac{n}{b} \rfloor$ nebo $\lceil \frac{n}{b} \rceil$.

1. Jestliže $f(n) = \mathcal{O}(n^{\log_b a - \epsilon})$ pro nějakou konstantu $\epsilon > 0$, pak $T(n) = \Theta(n^{\log_b a})$.
2. Jestliže $f(n) = \Theta(n^{\log_b a})$, pak $T(n) = \Theta(n^{\log_b a} \lg n)$.
3. Jestliže $f(n) = \Omega(n^{\log_b a + \epsilon})$ pro nějakou konstantu $\epsilon > 0$ a jestliže $a\,f(\frac{n}{b}) \leq c\,f(n)$ pro nějakou konstantu $c < 1$ pro všechna dostatečně velká n, pak $T(n) = \Theta(f(n))$.

1.3.3 Poznámka. Věta 1.3.2 nepokrývá všechny možné případy. Případy, které nejsou pokryty:

1. Funkce $f(n) = \mathcal{O}(n^{\log_b a})$, ale $f(n) \neq \mathcal{O}(n^{\log_b a - \epsilon})$ pro žádné $\epsilon > 0$. Jinými slovy, $f(n)$ není polynomiálně menší než $\mathcal{O}(n^{\log_b a})$.
2. Funkce $f(n) = \Omega(n^{\log_b a})$, ale $f(n) \neq \mathcal{O}(n^{\log_b a + \epsilon})$ pro žádné $\epsilon > 0$ (jinými slovy, $f(n)$ není polynomiálně větší než $\mathcal{O}(n^{\log_b a})$) nebo neplatí $a\,f(\frac{n}{b}) \leq \leq c\,f(n)$.

1.3.4 Tvrzení. Jestliže $f(n) = \Theta(n^{\log_b a} \lg^k n)$ pro $k \geq 0$, pak pro funkci $T(n)$ danou rovnicí

$$T(n) = a\,T(\frac{n}{b}) + f(n),$$

platí: $T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$.

1.4 Správnost algoritmů

1.4.1 K ověření správnosti algoritmu je třeba ověřit dvě věci

1. algoritmus se na každém vstupu zastaví,
2. algoritmus po zastavení vydá správný výstup – řešení.

Použití obou kroků si nejprve ukážeme na velmi dobře známých algoritmech.

1.4.2 Bublinkové třídění.

Vstup: posloupnost přirozených čísel $a[1], a[2], \ldots, a[n]$.

Výstup: setříděná do neklesající posloupnosti.

```
begin
     for k = n step -1 untill 2 do
          for j = 1 step 1 untill k − 1 do
              if a[j] > a[j + 1] then
                  zaměň a[j] a a[j + 1]
end
```

1.4.3 Fakt, že se algoritmus 1.4.2 zastaví, je zaručen tím, že vnější cyklus se opakuje $n - 1$-krát.

1.4.4 Tvrzení. Pro $k = n - i$ (tj. po i-tém proběhnutí vnějšího cyklu) jsou $a[n-i+1], a[n-i+2], \ldots, a[n]$ největší z čísel $a[1], a[2], \ldots, a[n]$ a navíc $a[n - - i + 1] \leq a[n - i + 2] \leq \ldots \leq a[n]$.

1.4.5 Eukleidův algoritmus.

Vstup: čísla a, b, $b \neq 0$.

Výstup: $d = gcd(a, b)$.

1. (Inicializace.)
   `if` $a \geq b$ `then` $r := a$, $t := b$;
   `else` $r := b$, $t := a$.

2. (Výpočet částečného podílu a zbytku.)
   repeat until $z = 0$
   `do` $r = p \cdot t + z$;
   $r := t$, $t := z$.

3. (Největší společný dělitel.)
   $d := t$.

1.4.6 Protože pro zbytky při dělení čísla r číslem t se stále zmenšují a jsou to přirozená čísla, musí jednou nastat případ, kdy zbytek je nula.

1.4.7 Tvrzení. Dvojice čísel r, t a dvojice čísel t, z z Eukleidova algoritmu 1.4.5 mají stejné společné dělitele.

1.4.8 Variant. Pro důkaz faktu, že se algoritmus na každém vstupu zastaví, je založen na nalezení tzv. *variantu*. Variant je hodnota udaná přirozeným číslem, která se během práce algoritmu snižuje až nabude nejmenší možnou hodnotu (a tím zaručuje ukončení algoritmu po konečně mnoha krocích).

V příkladu 1.4.2 se jednalo o číslo k, v příkladu 1.4.5 se jednalo o zbytek z při dělení čísla r číslem t.

1.4.9 Invariant. *Invariant*, též *podmíněná správnost algoritmu*, je tvrzení, které

- platí před vykonáním prvního cyklu algoritmu, nebo po prvním vykonání cyklu,
- platí-li před vykonáním cyklu, platí i po jeho vykonání,
- při ukončení práce algoritmu zaručuje správnost řešení.

Pro algoritmus pro bublinkové třídění je invariantem tvrzení 1.4.4, pro Euikleidův algoritmus tvrzení 1.4.7.

1.4.10 Minimální kostra. Je dán prostý neorientovaný graf $G = (V, E)$ s množinou vrcholů V a množinou hran E. Dále je dáno ohodnoceni c hran, tj zobrazení $c\colon E \to \mathbb{N}$. Úkolem je najít kostru K grafu G takovou, že

$$\sum_{e \in K} c(e) \quad \text{je nejmenší.}$$

Ukážeme správnost jakéhokoli algoritmu založeného na následujícím schematu.

1.4.11 Obecné schema.

Vstup: neorientovný graf $G = (V, E)$ a ohodnocení hran c.

Výstup: hrany minimální kostry K.

1. (Inicializace)
   $K := \emptyset$, $\mathcal{S} = \{\{v\} \mid v \in V\}$;
2. (Výběr hrany.)
   Dokud $\mathcal{S}$ není jednoprvková
   vybereme hranu $e \in E \setminus K$ takovou, že
   vede mezi dvěma množinami C_1 a C_2 z $\mathcal{S}$ a
   aspoň pro jednu z C_1, C_2 je nejlevnější hrana.
3. (Úpravy.)
   $K := K \cup \{e\}$;

$$\mathcal{S} := (\mathcal{S} \setminus \{C_1, C_2\}) \cup \{C_1 \cup C_2\}.$$

1.4.12 Ukončeni schematu pro minimální kostru (variant). Při zpracování každého výběru hrany v kroku 2 se zmenší počet množin v systému $\mathcal{S}$ o jednu. Protože $\mathcal{S}$ má na začátku práce schematu n množin, po $n-1$ krocích 3 bude $\mathcal{S}$ jednoprvková a schema skončí.

1.4.13 Tvrzení (invariant). Jestliže množina hran K před vykonáním kroku 2 je částí některé minimální kostry a vybereme-li hranu e podle schematu 1.4.11, pak množina hran $K \cup \{e\}$ je také částí některé minimální kostry.

1.4.14 Pozorování. Jak Kruskalův algorismus, tak Primův algoritmus jsou zvláštní případy obecného schematu 1.4.11.

1.4.15 Nejkratší cesty. Je dán prostý orientovaný graf $G = (V, E)$ a ohodnocení hran a, tj. zobrazení $a\colon E \to \mathbb{Z}$.

1.4.16 Matice délek A. *Matice délek* je čtvercová matice $\mathbf{A} = (a(i,j))$ řádu n, n je počet vrcholů grafu G, kde

$$a(i,j) = \begin{cases} 0, & \text{pro } i = j \\ a(e), & \text{pro } e = (i,j) \in E \\ \infty, & \text{pro } (i,j) \notin E \end{cases}$$

1.4.17 Matice vzdáleností U. *Matice vzdáleností* je čtvercová matice $\mathbf{U} = (u(i,j))$ řádu n, n je počet vrcholů grafu G, kde

$$u(i,j) = \begin{cases} 0, & \text{pro } i = j, \\ \text{délka nejkratší cesty z } i \text{ do } j, & \text{jestliže existuje cesta z } i \text{ do } j \\ \infty, & \text{jestliže neexistuje cesta z } i \text{ do } j \end{cases}$$

1.4.18 Pozorování. Předpokládejme, že vrchol y je orientovaně dostupný z vrcholu x v grafu G. Pak platí:

1. Jestliže graf G obsahuje pouze cykly kladné délky (tj. neobsahuje ani cykly záporné délky ani nulové délky), pak nejkratší sled z vrcholu x do vrcholu y existuje a je současně nejkratší cestou z x do y..
2. Jestliže v grafu G neexistuje cyklus záporné délky, pak nejkratší sled z x do y má stejnou délku jako nejkratší cesta z x do y.
3. Jestliže v grafu G neexistuje cyklus záporné délky, pak pro každý sled C z x do y existuje cesta z x do y, která je kratší nebo stejně dlouhá jako sled C.

1.4.19 Trojúhelníková nerovnost. Jestliže v grafu G neexistuje cyklus záporné délky, pak pro každé tři vrcholy x, y, z platí

$$u(x,y) \leq u(x,z) + u(z,y).$$

1.4.20 Bellmanův princip optimality. Jestliže v grafu G neexistuje cyklus záporné délky, pak pro každé tři vrcholy x, y, z platí

$$u(x,y) = \min_{z \neq y}(u(x,z) + a(z,y)).$$

1.4.21 Nejkratší cesty z výchozího vrcholu r. Úloha: Najděte délky nejkratších cest z výchozího vrcholu r.

1.4.22 Obecné schema.

Vstup: orientovný graf $G = (V, E)$ a ohodnocení hran a.

Výstup: hodnoty $U(v)$ rovné $u(r, v)$.

1. (Inicializace.)
   $U(r) := 0$, $U(v) := \infty$ pro $v \neq r$;
2. (Zpracování hran.)
   Existuje-li hrana $e = (v, w)$ taková, že
   $U(w) > U(v) + a(e)$
   položíme $U(w) := U(v) + a(e)$.
3. (Ukončení.)
   Jestliže $U(w) \leq U(v) + a(e)$ pro každou hranu $e = (v, w)$
   stop.
   Jinak pokračuj krokem 2.

1.4.23 Tvrzení. Jestliže v grafu G neexistuje cyklus záporné délky a hodnota $U(v) \neq \infty$, pak $U(v)$ je délka některé cesty z vrcholu r do vrcholu v.

1.4.24 Věta. Jestliže graf G neobsahuje cyklus záporné délky a hodnoty $U(v)$ byly získány podle schematu 1.4.22, pak $U(v) = u(r, v)$.

1.4.25 Nejkratší cesty mezi všemi dvojicemi vrcholů. Úkolem je najít celou matici vzdáleností (a ne jen jeden její řádek).

1.4.26 Označme množinu vrcholů $V(G) = \{1, 2, \ldots, n\}$. Floydův algoritmus (v literatuře též nazývaný Floyd-Warshallův algoritmus) je založen na konstrukci matic $\mathbf{U}_k = (u_k(i,j))$ řádu n pro $k = 0, 1, \ldots, n$ s následující vlastnosti:

$u_k(i,j)$ je délka nejkratší cesty z i do j, která prochází pouze vrcholy $1, 2, \ldots, k$.

1.4.27 Tvrzení. Platí

1. $\mathbf{U}_0$ je matice délek $\mathbf{A}$.
2. $\mathbf{U}_n$ je matice vzdáleností $\mathbf{U}$.
3. Matici $\mathbf{U}_{k+1}$ získáme z matice $\mathbf{U}_k$ takto:

$$u_{k+1}(i,j) = \min\{u_k(i,j), u_k(i,k) + u_k(k,j)\}.$$

1.4.28 Floydův algoritmus.

Vstup: matice délek $\mathbf{A}$.

Výstup: matice vzdáleností $\mathbf{M} = \mathbf{U}$.

```
1. [Inicializace]
       M := A
2.     begin
           for k = 1,2,...,n do
               for i = 1,2,...,n do
                   for j = 1,2,...,n do
                       begin
                           if M(i,j) > M(i,k) + M(k,j) then
                               M(i,j) = M(i,k) + M(k,j)
                       end
       end
```

1.4.29 Ukončení Floydova algoritmu je zaručeno tím, že vnější cyklus se provádí n-krát, tj. variant je k, které se roste od 1 do n.

Invariantem je 1.4.26 a vlastnost 3 z 1.4.27.

1.5 Modely výpočtu

1.5.1 Počítač s libovolným přístupem – RAM se skládá z programové jednotky, aritmetické jednotky, paměti a vstupní a výstupní jednotky.

1.5.2 Programová jednotka obsahuje programový register a vlastní program (programový registr ukazuje na instrukci, která má být provedena).

1.5.3 Aritmetická jednotka provádí aritmetické operace sčítání, odčítání, násobení a celočíselné dělení.

1.5.4 Paměť je rozdělena na paměťové buňky, každá buňka může obsahovat celé číslo. Předpokádáme neomezený počet paměťových buněk a neomezenou velikost čísel uložených v paměťových buňkách. Pořadové číslo paměťové buňky je *adresa* této buňky.

Buňka s adresou 0 je *pracovní registr*, s adresou 1 je *indexový registr*.

1.5.5 Vstupní jednotka je tvořena vstupní páskou a hlavou. Vstupní páska je rozdělena na pole (v každém poli může být celé číslo). Hlava snímá v každém okamžiku jedno pole. Po přečtení pole se hlava posune o jedno pole doprava.

1.5.6 Výstupní jednotka je tvořena výstupní páskou a hlavou. Obdobně jako v případě vstupní jednotky je páska rozdělena na pole. Výstupní hlava zapíše číslo do pole výstupní pásky a posune se o jedno pole doprava.

1.5.7 Konfigurace počítače s libovolným přístupem je přiřazení, které každému poli vstupní i výstupní pásky, každé paměťové buňce a programovému registru přiřazuje celé číslo. *Počáteční konfigurace* je konfigurace pro kterou existuje přirozené číslo n s následující vlastností: kromě prvních n vstupních polí obsahují všechna pole, paměťové buňky i programový registr číslo 0 a prvních n polí obsahuje vstup počítače.

1.5.8 Výpočet počítače s libovolným přístupem je posloupnost konfigurací, taková, že začíná počáteční konfigurací a každá následující konfigurace je určena programem počítače.

1.5.9 Program počítače s libovolným přístupem používá následující příkazy:

- příkazy přesunu: LOAD operand, STORE operand,
- aritmetické příkazy: ADD operand, SUBTRACT operand, MULTIPLY operand, DIVIDE operand,
- vstupní a výstupní příkazy: READ, WRITE,
- příkazy skoku: JUMP návěští, JZERO návěští, JGE návěští,
- příkazy zastavení: STOP, ACCEPT, REJECT.

1.5.10 Operand je buď číslo j, zapisujeme $= j$, nebo obsah j-té paměťové buňky, zapisujeme j, nebo obsah paměťové buňky s adresou $i + j$, kde i je obsah indexového registru, zapisujeme $*j$.

1.5.11 Návěští je přirozené číslo, které udává pořadové číslo instrukce, která bude prováděna, dojde-li ke skoku.

1.5.12 Časová složitost. Řekneme, že program P pro RAM pracuje s časovou složitostí $\mathcal{O}(f(n))$, jestliže pro každý vstup délky n je počet kroků počítače $T(n)$ roven $\mathcal{O}(f(n))$.

1.5.13 Paměťová složitost. Řekneme, že program P pro RAM pracuje s pamětí velikosti m, jestliže během výpočtu nebyl proveden žádný příkaz, který by měl adresu operandu větší než m a byl proveden příkaz s adresou m.

1.5.14 Poznámka. Jestliže se na nějakém vstupu program pro RAM nezastaví, není definována ani časová ani paměťová složitost.

1.5.15 Věta. Ke každému programu P pro RAM, který pracuje v čase $\mathcal{O}(f(n))$, existuje program P', který má časovou složitost $\mathcal{O}([f(n)]^2)$ a paměťovou složitost $\mathcal{O}(f(n))$.

1.5.16 Turingův stroj. Turingův stroj si můžeme představit takto: skládá se

- z řídící jednotky, která se může nacházet v jednom z konečně mnoha stavů,
- potenciálně nekonečné pásky rozdělené na jednotlivá pole a
- hlavy, která umožňuje číst obsah polí a přepisovat obsah polí pásky.

Na základě informace X, která je přečtena na pásce, a na základě stavu q, ve kterém se nachází řídící jednotka Turingova stroje, se řídící jednotka přesune do stavu p, pole pásky přepíše na Y a hlava se přesune buď doprava nebo doleva (tato akce je popsána tzv. přechodovou funkcí).

1.5.17 Formální definice. Turingův stroj je sedmice $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$, kde

- Q je konečná množina stavů,
- Σ je konečná množina vstupních symbolů,
- Γ je konečná množina páskových symbolů, přítom $\Sigma \subset \Gamma$,
- B je prázdný symbol (též nazývaný *blank*), jedná se o páskový symbol, který není vstupním symbolem, (tj. $B \in \Gamma \setminus \Sigma$),
- δ je přechodová funkce, tj. parciální zobrazení z množiny $Q \times \Gamma$ do množiny $Q \times \Gamma \times \{L, R\}$, (zde L znamená pohyb hlavy o jedno pole doleva, R znamená pohyb hlavy o jedno pole doprava),
- $q_0 \in Q$ je počáteční stav a
- $F \subseteq Q$ je množina koncových stavů.

1.5.18 Situace. Situaci Turingova stroje plně popisuje páska, pozice hlavy na pásce a stav, ve kterém se nachází řídící jednotka. Jestliže na pásce jsou v prvních k polích symboly $X_1\, X_2\, \ldots\, X_k$, všechna pole s větším číslem již obsahují pouze B, řídící jednotka je ve stavu q a hlava čte symbol X_i, tak danou situaci zapisujeme

$$X_1\, X_2\, \ldots\, X_{i-1}\, q\, X_i\, X_{i+1}\, \ldots\, X_k.$$

1.5.19 Počáteční situace. Na začátku práce se Turingův stroj nachází v počátečním stavu q_0, na pásce má na prvních n polích vstupní slovo $a_1\, a_2\, \ldots\, a_n$ ($a_i \in \Sigma$), ostatní pole obsahují blank B a hlava čte první pole pásky, tj. symbol a_1. Tedy formálně je počáteční situace $q_0\, a_1\, \ldots\, a_n$.

1.5.20 Krok Turingova stroje. Předpokládejme, že se Turingův stroj nachází v situaci $X_1\, X_2\, \ldots\, X_{i-1}\, q\, X_i\, \ldots\, X_k$. Pak v jednom kroku udělá následující:

Jestliže $\delta(q, X_i) = (p, Y, R)$, stroj se přesune do stavu p, na pásku napíše symbol Y a hlavu posune o jedno pole doprava. Formálně to zapisujeme:

$$X_1\, X_2\, \ldots\, X_{i-1}\, q\, X_i\, \ldots\, X_k \vdash X_1\, X_2\, \ldots\, X_{i-1}\, Y\, p\, X_{i+1}\, \ldots\, X_k.$$

Jestliže $\delta(q, X_i) = (p, Y, L)$, a hlava nečte nejlevější pole, stroj napíše na pásku Y (místo X_i) a posune hlavu o jedno pole doleva. Formálně to zapisujeme:

$$X_1\, X_2\, \ldots\, X_{i-1}\, q\, X_i\, \ldots\, X_k \vdash X_1\, X_2\, \ldots\, X_{i-2}\, p\, X_{i-1}\, Y\, X_{i+1}\, \ldots\, X_k.$$

Jestliže $\delta(q, X_i) = (p, Y, L)$, a hlava čte nejlevější pole, nebo jestliže $\delta(q, X_i)$ není definováno, stroj se neúspěšně zastaví.

1.5.21 Výpočet Turingova stroje je posloupnost jeho kroků, která začíná v počátečním stavu. Tedy jedná se o reflexivní s tranzitivní uzávěr $\vdash^\star$ relace $\vdash$ (na množině všech situací daného Turingova stroje).

Jestliže existuje výpočet Turingův stroj takový, že se stroj dostane do jednoho z koncových stavů $q' \in F$, stroj se úspěšně zastaví. Obsah pásky při úspěšném zastavení je *výstupem*, který Turingův stroj vydá na vstup $a_1\, a_2 \ldots\, a_n$.

1.5.22 Chování Turingova stroje je zobrazení, které každému vstupu na němž se Turingův stroj úspěšně zastaví, přiřadí obsah pásky v okamžiku zastavení, tj. výstup Turingova stroje.

1.5.23 Časová složitost Turingova stroje je parciální zobrazení $T(n)$ z množiny všech přirozených čísel do sebe definované:

Jestliže pro nějaký vstup délky n se Turingův stroj nezastaví, $T(n)$ není definováno. V opačném případě je $T(n)$ rovno maximálnímu počtu kroků, po nichž dojde k zastavení Turingova stroje, kde maximum se bere přes všechny vstupy délky n.

1.5.24 Jazyk přijímaný Turingovým strojem. Turingův stroj se používá také jako akceptor. V tomto případě záleží pouze na tom, zda se Turingův stroj zastaví úspěšně. Vstupní slovo $w \in \Sigma^\star$ je *přijato* Turingovým strojem právě tehdy, když se Turingův stroj na slově w uspěšně zastaví. Množinu slov $w \in \Sigma^\star$, které Turingův stroj přijímá, se nazývá *jazyk přijímaný* Turingovým strojem M a značíme ho $L(M)$.

1.5.25 Poznámka. Základní model Turingova stroje, tak jak jsme ho uvedli v minulých odstavcích, není jediným modelem. Jiná varianta Turingva stroje pracuje s oboustranně nekonečnou páskou, tj. páskou, která je nekonečná „na obě strany". Oba tyto modely jsou ekvivalentní v tom smyslu, že pro každý Turingův stroj M_1 jednoho typu existuje Turingův stroj M_2 druhého typu tak, že oba stroje mají stejné chování. Existují však i další typy Turingova stroje. Uvedeme je v následujícím textu.

1.5.26 Turingův stroj s k páskami. Turingův stroj s k páskami se skládá z řídící jednotky, která se nachází v jednom z konečně mnoha stavů $q \in Q$, množiny vstupních symbolů Σ, množiny páskových symbolů Γ, přechodové funkce δ, počátečního stavu q_0, páskového symbolu B a množiny koncových stavů F. Dále je dáno k pásek a k hlav; i-tá hlava vždy čte jedno pole i-té pásky. Přechodová funkce δ je parciální zobrazení, které reaguje na stav, ve kterém se Turingův stroj nachází a na k-tici páskových symbolů, kterou jednotlivé hlavy snímají. (Formálně je δ parciální zobrazení, $\delta\colon Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R\}^k$).

Na začátku:

1. Vstupní slovo je na první pásce s tím, že první symbol je na prvním poli.
2. Všechna ostatní pole první pásky a všechna pole ostatních pásek obsahují B.
3. Řídící jednotka je v počátečním stavu q_0.
4. Všechny hlavy čtou první pole odpovídající pásky.

1.5.27 Krok Turingova stroje s k páskami je udán přechodovou funkcí. Jestliže přechodová funkce je definována a nedojde na žádné pásce k překročení levého okraje, pak (na základě přechodové funkce):

1. Řídící jednotka se přesune do nového stavu.
2. Každá hlava přepíše obsah pole, které čte (může i stejným symbolem).
3. Každá hlava se posune doprava nebo doleva.

1.5.28 Chování Turingova stroje. Turingův stroj se úspěšně zastaví, jestliže řídící jednotka vstoupila do koncového stavu. Jestliže Turingův stroj nemá definován následující krok a není v koncovém stavu, říkáme, že se Turingův stroj neúspěšně zastavil.

1.5.29 Poznámka. Na každý Turingův stroj s jednou páskou se můžeme dívat jako na Turingův stroj s k páskami, kde $k = 1$. Proto Turingův stroj s jednou páskou je zvláštní případ Turingova stroje s k páskami.

1.5.30 Věta. Ke každému Turingovu stroji M_1 s k páskami existuje Turingův stroj s jednou páskou M_2, který má stejně chování jako M_1.

Navíc, jestliže M_1 potřeboval k úspěšnému zastavení n kroků, pak M_2 potřebuje $\mathcal{O}(n^2)$ kroků.

1.5.31 Tvrzení. Ke každému Turingovu stroji M existuje program P pro RAM takový, že oba mají stejné chování. Navíc, jestliže M potřeboval n kroků, P má časovou složitost $\mathcal{O}(n^2)$.

1.5.32 Věta. Pro každý program P pro RAM existuje Turingův stroj M s pěti páskami takový, že P i M mají stejné chování.

1.5.33 Věta. Jestliže program P pro RAM splňuje následující podmínky:

- program obsahuje pouze instrukce, které zvětšují délku binárně zapsaného čísla maximálně o jednu;
- program obsahuje pouze instrukce, které Turingův stroj s více páskami provede na slovech délky k v $\mathcal{O}(k^2)$ krocích,

pak Turingův stroj z věty 1.5.32 simuluje n kroků programu P pomocí $\mathcal{O}(n^3)$ svých kroků.

1.5.34 Důsledek. Předpokládejme, že program P pro RAM splňuje podmínky z věty 1.5.33. Pak existuje Turingův stroj s jednou páskou, který má stejné chování jako P a n kroků programu P simuluje pomocí $\mathcal{O}(n^6)$ svých kroků.

1.6 Rozhodovací úlohy

1.6.1 Teorie složitosti pracuje zejména s tzv *rozhodovacími* úlohami. Rozhodovací úlohy jsou takové úlohy, jejichž „řešením“ je buď ano nebo ne.

1.6.2 SAT – splňování Booleovských formulí. Je dána výroková formule φ v CNF. Rozhodněte, zda je φ splnitelná.

Na danou formuli φ je tedy odpověď = 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.

1.6.3 Řada praktických úloh není podobného druhu jako uvedené příklady. Jedná se 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 nějvětší nebo naopak nejmenší. V dalším textu se s řadou těchto úloh setkáme. Nyní uvedeme jeden příklad.

1.6.4 Problém obchodního cestujícího – TSP. Je dáno n měst $1, 2, \ dots, n$. Pro každou dvojici měst i, j je dáno kladné číslo (vzdálenost měst i, j) $d(i, j)$. *Trasa* je dána permutací π množiny $\{1, 2, \ldots, n\}$ do sebe. Cena trasy s permutaci π je

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

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 vzdálenosti, které při své cestě urazil.

1.6.5 TSP – rozhodovací verze. Kromě čísel $d(i, j)$ z 1.6.4 je dáno číslo C. Existuje trasa π ceny nejvýše C?

1.6.6 TSP – vyhodnocovací verze. Jsou dána čísla $d(i, j)$ a 1.6.4. Najděte cenu optimální trasy, tj. trasy s nejmenší možnou cenou.

1.6.7 TSP – optimalizační verze. Jsou dána čísla $d(i, j)$ a 1.6.4. Najděte optimální trasu, tj. trasu s nejmenší možnou cenou.

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

1.7.1 Instance úlohy jako slovo nad abecedou. Instance rozhodovací úlohy můžeme zakódovat jako slova nad vhodnou abecedou.

- Pro problém SAT (splňování booleovských formulí) je instancí formule v CNF. Zakódování takové formule ukážeme na příkladu. Je dána formule $\varphi = (x \vee \neg y \vee z) \wedge (\neg x \vee t)$. Logické proměnné formule očíslujeme, tj. položíme $x = x_1$, $y = x_2$, $z = x_3$ a $t = x_4$. Pak formule φ je rovna $\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_4)$. Formuli nyní zakódujeme jako slovo nad abecedou $\{x, 0, 1, (,), \vee, \wedge, \neg\}$ takto

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

  To znamená, za x dáme vždy binární zápis pořadového čísla dané proměnné.

- 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.

1.7.2 Úloha jako jazyk nad abecedou. Protože řešením rozhodovací úlohy je buď „ANO“ nebo „NE“, rozdělíme instance 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 $\mathcal{U}$.

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

1.7.4 Příklady.

- Minimální kostra v grafu. Je dán neorinetovaný 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.

1.7.5 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.

1.7.6 Poznámka. V definice 1.7.3 jsme místo existence Turingova stroje mohli požadovat existenci programu P pro RAM, který řeší $\mathcal{U}$ v polynomiálním čase. Analogií nedeterministického Turingova stroje je nedeterministický algoritmus.

1.7.7 Nedeterministický algoritmus pracuje ve dvou fázích,

1. Algoritmus náhodně vygeneruje řetězec s.

2. Deterministický algoritmus (program pro RAM) na základě vstupu a řetězce s dá odpověď ANO nebo NEVIM.

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.

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

1.7.9 V definice 1.7.5 jsme místo existence nedeterministického Turingova stroje mohli požadovat existenci nedeterministického algoritmu, který řeší úlohu $\mathcal{U}$ v polynomiálním čase.

1.7.10 Příklady.

- 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

$$\sum_{i \in I} w_i \leq A \quad \text{a} \quad \sum_{i \in I} c_i \geq B?$$

1.8 Třída $\mathcal{NPC}$

1.8.1 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) $\mathcal{M}$, který pro každou instanci I úlohy $\mathcal{U}$ zkonstruuje instanci I' úlohy $\mathcal{V}$ a to tak, že

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

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

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

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

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

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

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

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

1.8.4 $\mathcal{NP}$ 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.

1.8.5 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.

1.8.6 Tvrzení. Jestliže by 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.

1.8.7 $\mathcal{NP}$ 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.

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

1.8.9 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 $\mathcal{M}$ s množinou stavů Q, vstupní abecedou Σ, páskovou abecedou Γ, přechodovou funkcí δ, počátečním stavem q_0 a koncovým stavem q_f. Předpokládejme, že $\mathcal{M}$ 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 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 body 1, 4, 5, 6 a 7.

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 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 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, 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:

$$\bigwedge_j \bigwedge_{A\in\Gamma} ((s_i^q \wedge h_{i,j} \wedge t_{i,j}^A) \Rightarrow \bigvee (s_{i1}^p \wedge t_{i+1,j}^C \wedge h_{i+1,j+D})).$$

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 $i = 0, 1, \dots, p(n)$.

1.9 Převody úloh

1.9.1 Na základě tvrzení 1.8.5 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 další rozhodovací úlohy jsou $\mathcal{NP}$ úplné.

1.9.2 $3-CNF\ SAT$**.** Úloha: Je dána formule φ v konjunktivním normálním tvaru, kde každá klasusule má 3 literály.
Otázka: Je formule φ splnitelná?

1.9.3 Tvrzení. Platí

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

1.9.4 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 klausule obsahuje maximálně 3 literály;

2. je splnitelná právě tehdy, když je splnitelná formule φ.

Označme $C_1, C_2, \ldots, C_k$ všechny klausule formule φ. Jestliže každá z klausulí obsahuje nejvýše 3 literály, nemusíme nic konstruovat – $\psi = \varphi$.

Pro každou klausuli C, která obsahuje víc než 3 literály, sestrojíme formuli ψ_C takto: Nechť $C = l_1 \vee l_2 \vee \ldots \vee l_s$. 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 iff C je splnitelná.

Formuli ψ dostaneme jako konjunkci všech formulí ψ_C pro klausule C o více než 3 literálech.

Předpokládejme, že formule φ má k klausulí a nejdelší klausule má s literálů. Pak v konstrukci ψ jsme přidali maximálně $(s-3)$ nových logických proměnných (rovnost nastává v případě, že každá z klausulí 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.

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

1.9.6 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?

1.9.7 Tvrzení. Platí

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

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

G má pro každou klausuli jednu stranu; strana odpovídající klausuli C se skládá ze 3 vrcholů označených literály klasusule 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 klausulí formule φ.

Jestliže φ je pravdivá v ohodnocení u, vybereme v každé klausuli 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.

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

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

1.9.10 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?

1.9.11 Tvrzení. Platí

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

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

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

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.)

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

1.9.14 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?

1.9.15 Tvrzení. Platí

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

1.9.16 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.

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

1.9.18 Vrcholové barvení. Je dán prostý neorientovaný graf bez smyček $G = (V, E)$. *Vrcholové obarvení* grafu G je přiřazení, které každému vrcholu v grafu G přiřazuje jeho barvu $b(v)$, $b(v)$ je prvek množiny (barev) B, a pro přiřazení b 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).

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

1.9.20 Tvrzení. Platí

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

1.9.21 Základní myšlenka převodu. Je dána formule φ, která je v CNF a každá klausule má 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 o pěti vcholech $\{1, 2, 3, 4, 5\}$ a pěti hranách s touto vlastností:

- Jestliže vrcholy 1 a 2 mají stejnou barvu, pak tuto barvu musí mít i vrchol 5.
- Jestliže jeden z vrcholů 1 a 2 má barvu z, pak lze tento graf obarvit tak, aby i vrchol 5 měl barvu z.

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

1.9.23 Tvrzení. Platí

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

1.9.24 Nástin převodu 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: Proměnná $x_v^b = 1$, $b \in \{c, m, z\}$, znamená, že vrchol v má barvu b.

Podmínky:

- Pro každý vrchol $v \in V$ máme rovnici, která říká, 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:

$$x_u^c + x_v^c \leq 1, \quad x_u^m + x_v^m \leq 1, \quad x_u^z + x_v^z \leq 1.$$

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 $m = |E|$.

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

1.9.26 Rozklad množiny. Je dána konečná množina X a systém jejích podmnožin $\mathcal{A}$. Řekneme, že $\mathcal{A}$ je rozklad množiny X, jestliže jsou splněny následující dvě podmínky

1. každý prvek $x \in X$ leží v některé podmnožině $B \in \mathcal{A}$, (tj. $\bigcup\{B \mid B \in \in \mathcal{A}\} = X$;
2. žádné dvě různé podmnožiny z $\mathcal{A}$ nemají společný prvek, tj. jsou po dvou disjunktní.

1.9.27 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 $\mathcal{S}' \subseteq \mathcal{S}$ tak, že $\mathcal{S}'$ je rozklad množiny X?

1.9.28 Tvrzení. Platí

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

1.9.29 Nástin převodu vrcholového pokrytí na problém rozkladu. Je dán neorientovaný prostý graf bez smyček $G = (V, E)$. Zkonstruujeme množinu X a systém jejích podmnožin $\mathcal{S}$ tak, že z 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. $N(v) = \{u \,|\, \{u, v\} \in E\}$. Do $\mathcal{S}$) dáme množiny:

$$S_v^c = \{p_v^c, q_{vu}^c \,|\, u \in N(v)\}, S_v^m = \{p_v^m, q_{vu}^m \,|\, u \in N(v)\}, S_v^z = \{p_v^z, q_{vu}^z \,|\, u \in N(v)\}.$$

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 $\mathcal{S}'$ takto:

$\mathcal{S}'$ 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 $\mathcal{S}'$ množiny X, pak sestrojíme obarvení grafu G takto:

$$b(v) := b, b \in \{c, m, z\} \quad \text{iff} \quad \{v, p_v^b\} \in \mathcal{S}'.$$

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

1.9.31 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?)

1.9.32 Tvrzení. Platí

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

1.9.33 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 uměrně počtu hran grafu G plus dvojnásobek počtu vrcholů grafu G.

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

1.9.35 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.

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

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

1.9.37 Zdůvodnění tvrzení 1.9.36. Za předpokladu tvrzení 1.9.36 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 exituje 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 $n - \\ - 1 + n\,r + 1 = n\,r + n$. 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 $\mathcal{P} = \mathcal{NP}$.

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

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

1.9.39 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).

1.9.40 Slovní popis algoritmu z tvrzení 1.9.39. 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 výstupem 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 trojúhelnikové nerovnosti, je délka T menší nebo rovna dvojnásobku délky kostry K.

1.9.41 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.

1.9.42 Poznámka. Odhad délky trasy, kterou jsme získali v 1.9.40, i odhad pro trasu získanou Christofidesovým algoritmem není možné zlepšit.

1.10 Třída $co - \mathcal{NP}$

1.10.1 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.

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

1.10.3 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 $co - \mathcal{NP}$.

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

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

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

1.11.1 Je dán Turingův stroj M (determininstický nebo nedeterministický). Připomeňme, že M pracuje s paměťovou složitostí $p(n)$ právě tehdy, když pro každé slovo délky n nepoužije paměťovou buňku větší než $p(n)$.

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

1.11.3 Tvrzení. Platí

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

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

1.11.5 Tvrzení. Platí

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

1.11.6 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.

1.11.7 Myšlenka důkazu věty 1.11.6. 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 konfigurací. Zajímají nás totiž pouze takové výpočty, ve kterých se konfigurace neopakují. Označme t počet páskových symbolů Turingova stroje M a ozna4me s počet stavů M. Pak M má $p(n)\, s\, t^{p(n)}$ různých konfigurací.

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

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

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

1.11.8 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.

1.11.9 Myšlenka důkazy věty 1.11.8. 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 1.11.6), 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.

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.

1.11.10 Savitchova věta. Platí

$$\mathcal{PSPACE} = \mathcal{NPSPACE}.$$

1.11.11 Nástin myšlenky důkazu Savitchovy věty. Zřejmě $\mathcal{PSPACE} \subseteq \mathcal{NPSPACE}$. Důkaz opačné inkluze $\mathcal{NPSPACE} \subseteq \mathcal{PSPACE}$ spočívá v tom, že jsme schopni nedeterministický Turingův stroj pracující s paměťovou složitostí $p(n)$ simulovat deterministickým Turingovým strojem, který pracuje s paměťovou složitostí $\mathcal{O}([p(n)]^2)$.

Je dán nedeterministický Turingův stroj M, který přijímá jazyk L s polynomiální paměťovou složitostí $p(n)$. Konstrukce deterministického Turingova stroje přijímajícího stejný jazyk jako M s polynomiální paměťovou složitostí je založena na rekursivní proceduře $dostup(I, J, m)$, kde I a J jsou konfigurace a m je číslo. Výstup procedury $dostup(I, J, m)$ je buď 1, jestliže Turingův stroj se z konfigurace I do konfigurace J dostane v nejvýše m krocích, 0 v opačném případě. Procedura $dostup(I, J, m)$ pro každou konfiguraci K rekursivně zavolá procedury $dostup(I, K, m/2)$ a $dostup(K, J, m/2)$.

Pro vstup w voláme proceduru $dostup(I_0, J, m)$, kde I_0 je počáteční konfigurace M, J je přijímající konfigurace M a $m = \log_2 c^{p(n)+1}$ (c je konstanta z 1.11.6). Dá se dokázat, že pro vykonání procedury $dostup(I, J, m)$ deterministickým Turingovým strojem stačí paměťová složitost $\mathcal{O}([p(n)]^2)$. (Uvědomte si, že nám nezáleží na tom, jak dlouho deterministický Turingův stroj pracuje, zajímáme se pouze o paměťové nároky.)

1.11.12 Důsledek. Platí

$$\mathcal{P} \subseteq \mathcal{NP} \subseteq \mathcal{PSPACE}.$$

1.12 Pravděpodobnostní algoritmy

1.12.1 Nejprve uvedeme příklad Millerova testu prvočíselnosti. Jedná se o pravděpodobnostní algoritmus, který pro dané velké liché přirozené číslo odpovídá na otázku, zda je číslo složené.

1.12.2 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“.

1.12.3 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 $\frac{1}{2}$.

1.12.4 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 $(\mathbb{Z}_n, +, \cdot)$ konečné těleso. V tělese existují pouze dva prvky, které umocněné na druhou dávají 1 — totiž číslo 1 a -1. Proto nemůžeme skončit v kroku 6 výstupem „složené“.

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“, je důkaz dost obtížný.

1.12.5 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.

1.12.6 Poznámka. Rozdíl mezi RTM a obyčejným TM je v roli druhé pásky. Dvoupáskový TM 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.

1.12.7 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:

$$\begin{array}{ll}
\delta(q_0, 0, 0) = (q_1, 0, R, S), & \delta(q_0, 1, 0) = (q_2, 1, R, S), \\
\delta(q_1, 0, 0) = (q_1, 0, R, S), & \delta(q_1, B, 0) = (q_f, B, S, S), \\
\delta(q_2, 1, 0) = (q_2, 1, R, S), & \delta(q_2, B, 0) = (q_f, B, S, S), \\
\delta(q_0, a, 1) = (q_3, a, S, R), & \delta(q_3, a, a) = (q_3, a, R, R), \\
\delta(q_3, B, a) = (q_f, B, S, S), & \text{pro } a \in \{0, 1\}.
\end{array}$$

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 pravděpodbnost, že se M zastaví v q_f je
$$\frac{1}{2}\left(\frac{1}{2}\right)^n = 2^{-(n+1)}.$$

1.12.8 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. Každý běh M (tj. pro jakýkoli obsah druhé pásky) trvá maximálně $p(n)$ kroků, kde $p(n)$ je polynom a n je délka vstupního slova.

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

1.12.9 Turingův stroj typu Monte-Carlo. RTM splňující podmínky 1 a 2 z předchozí definice 1.12.8, se nazývá TM typu *Monte-Carlo*. (Uvědomte si, že RTM typu Monte-Carlo nemusí obecně pracovat v polynomiálním čase.)

1.12.10 Příklad. Jazyk L se skládá ze všech slov odpovídajících těm neorientovaným grafům, které obsahují trojúhleník (tj. úplný graf na třech vrcholech) jako podgraf.

Náš algoritmus v jednom kroku náhodně vybere jednu hranu, označme ji $\{x, y\}$, náhodně vybere jiný vrchol z a zkontroluje, zda podgraf indukovaný $\{x, y, z\}$ je úplný. Jestliže ano, úspěšně skončí, jestliže ne, opakuje postup znovu (tj. opět vybere hranu a k ní jeden vrchol a provede kontrolu). Výběr hrany, vrcholu a kontrolu provádí k-krát. Jestliže ani po k-tém opakování nenajde trojúhelník, neúspěšně skončí. Jedná se o algoritmus typu Monte-Carlo? A pro jaké k?

1.12.11 Po k výběrech a testech platí:

- Jestliže graf neobsahuje trojúhelnik, algoritmus se úspěšně zastaví s pravděpodobností 0.
- Jestliže graf obsahuje trojúhelnik, algoritmus se úspěšně zastaví s pravděpodobností
$$1 - \left(1 - \frac{3}{m(n-2)}\right)^k.$$
kde n je počet vrcholů grafu a m je počet hran.
- Jeden běh algoritmu trvá $\mathcal{O}(k\,n\,m)$.

Abychom dostali algoritmus typu Monte-Carlo, musí být $k \leq \frac{m(n-2)}{3}$.

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

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

1.12.13 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.

1.12.14 Turingův stroj typu Las-Vegas. RTM splňující podmínky z předchozí definice 1.12.13, se nazývá typu *Las-Vegas*.

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

1.12.16 Poznámka. Pro jazyky ze třídy $\mathcal{RP}$ se tvrzení obdobné 1.12.15 neumí dokázat. To motivuje následující třídu jazyků.

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

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

1.12.19 Věta. Platí

$$\mathcal{P} \subseteq \mathcal{ZPP}, \;\; \mathcal{RP} \subseteq \mathcal{NP}.$$

1.13 Nerozhodnutelnost

1.13.1 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

$$w = 0^i 10^j 10^k 10^l 10^r.$$

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.

1.13.2 Binární slova můžeme uspořádat do posloupnosti a tudíž je očíslovat. 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 $w_1 = 0$, $w_{102} = 100110$.

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

1.13.3 Diagonální jazyk L_d. Nejprve uděláme následující úmluvu. Jestliže binární slovo w nemá tvar z 1.13.1, považujeme ho za kód Turingova stroje M, který nepřijímá žádné slovo. 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.)

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

1.13.5 Zdůvodnění věty 1.13.4. 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 nepřijímá 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.

1.13.6 Rekursivní jazyky. Řekneme, že jazyk L je *rekursivní*, jestliže existuje Turingův stroj M, který přijímá jazyk L a na každém vstupu se zastaví.

V takovém případě také říkáme, že Turingův stroj M *rozhoduje jazyk* L.

1.13.7 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.

1.13.8 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. Ukážeme, že naopak to neplatí, tj. existují rekursivně spočetné jazyky, které nejsou rekursivní.

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

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

1.13.11 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.

1.13.12 Univerzální jazyk. *Univerzální jazyk* L_U 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 $w \in L(M)$.

1.13.13 Univerzální Turingův stroj. Popíšeme, velmi zhruba, Turingův stroj, který přijímá univerzální jazyk L_U. Tomuto Turingovu stroji se říká *univrzální Turingův stroj* a značíme ho U.

Univerzální Turingův stroj U má 3 pásky nebo více. 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 U může mít další, pomocné pásky.

Na začátku práce Turingova stroje M je na první pásce vstupní slovo $\langle M\rangle\#w$, ostatní pásky obsahují pouze B, blanky.

Turingův stroj U neprve 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 ako 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 následované B, blanky.)

1.13.14 Důsledek. Univerzální jazyk L_U je rekursivně spočetný.

1.13.15 Tvrzení. Univerzální jazyk L_U není rekursivní.

Kdyby totiž L_U byl rekursivní, existoval by Turingův stroj M, který rozhodne L_U. Tj. M se vždy zastaví a na slovech z jazyka L_U se úspěšně zastaví, na slovech neležících v L_U se neúspěšně zastaví. Na základě tohtoto Turingova stroje M bychom byli schopni rozhodnout diagonální jazyk L_d, o kterém víme, že není ani rekursivně spočetný, viz 1.13.4.

1.13.16 Redukce. Připomeňme definici redukce.

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 úlohy $\mathcal{V}$ značíme

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

Tato definice má význam i pro jazyky. Rozhodovací úlohu chápeme jako jazyk obsahující ta slova, která odpovídají ANO instancím.

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

1. Jestliže $\mathcal{U}$ je nerozhodnutelná, pak i $\mathcal{V}$ je nerozhodnutelná.
2. Jestliže $\mathcal{U}$ není rekursivně spočetná, pak i $\mathcal{V}$ není rekursivně spočetná.

1.13.18 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.

1.13.19 Poznámka. Uvědomme si, že jazyk L_e je dopňkem 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.

Universální Turingův stroj U se dá využít k tomu abychom ukázali, že jazyk L_{ne} je rekursivně spočetný. Z redukce $L_U \lhd L_{ne}$ a 1.13.17 dostáváme, že L_{ne} není rekursivní. Fakt, že L_e není ani rekursivně spočetný pak vyplývá z 1.13.11.

1.13.20 Věta (Rice). Jakákoli netriviálni vlastnost rekursivně spočetných jazyků (jazyků přijímaných Turingovým strojem) je nerozhodnutelná.

Netriviální vlastností rozumíme každou vlastnost, kterou má aspoň jeden rekursivně spočetný jazyk a nemají ho všechny rekursivně spočetné jazyky.

1.14 Další nerozhodnutelné úlohy

1.14.1 V minulé přednášce 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_U hraje pro nerozhodnutelné jazyky/úlohy obdobnou roli jako hrál problém splnitelnosti booleovských formulí pro $\mathcal{NP}$ úlné úlohy.

1.14.2 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?

1.14.3 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.

1.14.4 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?

1.14.5 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.

1.14.6 Věta. Platí

$$L_U \lhd \text{MPCP} \lhd \text{PCP}.$$

1.14.7 Poznámka. Druhá redukce z věty 1.14.6 je jednodušší. Pomocí rozšíření abecedy jsme schopni zkonstruovat instanci PCP tak, abychom měli zajištěno, že řešení, posloupnost indexů, **musí** začínat 1.

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.

1.14.8 Důsledek. Postův korespondenční problém je nerozhodnutelný.

1.14.9 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.

1.14.10 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.

1.14.11 Věta. Platí

$$\text{PCP} \lhd \text{víceznačnost bezkontextových gramatik.}$$

1.14.12 Nástin redukce pro důkaz věty 1.14.11. 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

$$S \to A \mid B,$$

$$A \to w_1\, A\, a_1 \mid w_2\, A\, a_2 \mid \ldots \mid w_k\, A\, a_k,$$

$$A \to w_1\, a_1 \mid w_2\, a_2 \mid \ldots \mid w_k\, a_k,$$

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

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

Pak gramatika $\mathcal{G}$ je víceznačná právě tehdy, když nějaké slovo $w a_{i_1} a_{i_2} \ldots a_{i_r}$, $w \in \Sigma^\star$, 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 $wa_{i_1}a_{i_2}\dots a_{i_r}$ 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.)

1.14.13 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. $L(\mathcal{G}_1) = \Sigma^\star$.

1.14.14 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 plochy tak, aby se dlaždičky dotýkaly hranami stejné barvy, za předpokladu, že dlaždičky nesmíme rotovat?

1.14.15 Věta. Tiling problém je nerozhodnutelný.