FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY
UNIVERZITA KOMENSKÉHO

Pavol Ďuriš

Výpočtová zložitosť

Máj 2009

Autor: Pavol Ďuriš
Názov: Výpočtová zložitosť
Vydavateľ: Knižničné a edičné centrum FMFI UK
Rok vydania: 2009
Miesto vydania: Bratislava
Vydanie: prvé
Počet strán: 35
Internetová adresa: http://www.fmph.uniba.sk/index.php?id=el_st_m
ISBN: 978-80-89186-51-8

Obsah

Kapitola 1

Terminológia

1.1 Výpočtový model

Ako výpočtový model nám bude slúžiť Turingov stroj s jednou vstupnou páskou a k pracovnými páskami, pričom na vstupnú pásku nemožno zapisovať. Hovoríme, že Turingov stroj pracuje s pamäťou $S(n)$ ak na žiadnej pracovnej páske nepoužije viac ako $S(n)$ políčok.

1.2 Zložitostné triedy

Definícia 1.2.1. $DTIME(T(n))$
- trieda jazykov akceptovateľných deterministickými Turingovými strojmi v čase $T(n)$ (t.j. na maximálne $T(n)$ krokov výpočtu).

Definícia 1.2.2. $NTIME(T(n))$
- trieda jazykov akceptovateľných nedeterministickými Turingovými strojmi v čase $T(n)$.

Definícia 1.2.3. $DSPACE(S(n))$
- trieda jazykov akceptovateľných deterministickými T-strojmi s pamäťou $S(n)$.

Definícia 1.2.4. $NSPACE(S(n))$
- trieda jazykov akceptovateľných nedeterministickými T-strojmi s pamäťou $S(n)$.

Definícia 1.2.5. $P = \bigcup_{k>0} DTIME(n^k)$

Definícia 1.2.6. $NP = \bigcup_{k>0} NTIME(n^k)$

Definícia 1.2.7. $PSPACE = \bigcup_{k>0} DSPACE(n^k)$

Definícia 1.2.8. $NPSPACE = \bigcup_{k>0} NSPACE(n^k)$

Definícia 1.2.9. $EXP = \bigcup_{k>0} DTIME(2^{n^k})$

Definícia 1.2.10. $NEXP = \bigcup_{k>0} NTIME(2^{n^k})$

Kapitola 2

Hierarchie a simulácie

2.1 Označenia

Definícia 2.1.1. Funkcia $S(n)$ je páskovo konštruovateľná, ak existuje deterministický Turingov stroj ktorý na každom vstupe dĺžky n nepoužije na žiadnej pracovnej páske viac ako $S(n)$ políčok a na prvej páske použije práve $S(n)$ políčok.

Definícia 2.1.2. Nech M je deterministický Turingov stroj s jednou pracovnou páskou. Kód stroja M (označujeme $\langle M \rangle$) je binárny reťazec tvaru:

$$1\,1\,1\,kód_1\,1\,1\,kód_2\,1\,1 \cdots 1\,1\,kód_h\,1\,1\,1\,,$$

kde každý $kód_i$ je tvaru:

$$0^j 1\, 0^k 1\, 0^l 1\, 0^m 1\, 0^r 1\, 0^s 1\, 0^t\,,$$

kódujúc tak prechodovú funkciu:

$$\delta(q_j, a_k, a_l) = (q_m, d_r, a_s, d_t)$$

Interpretácia symbolov:

- q_j – stav T-stroja,
- a_k – čítaný symbol na vstupnej páske,
- a_l – čítaný symbol na pracovnej páske,
- q_m – stav T-stroja v nasledujúcom kroku,
- a_s – symbol, ktory nahradí v nasled. kroku symbol a_l na prac. páske,
- d_r – počet políčok, o ktoré sa posunie hlava na vstupnej páske vpravo v nasledujúcom kroku,
- d_t – podobne ako d_r, ale pre pracovnú pasku.

Poznámka 2.1.1. Podobne kódujeme viacpáskové Turingove stroje.

2.2 Veta o pamäťovej hierarchii

Veta [o pamäťovej hierarchii]. [1] *Nech* $\log(n) \leq S_1(n) = o(S_2(n))$, *kde* $S_2(n)$ *je páskovo konštruovateľná. Potom:*

$$DSPACE(S_1(n)) \subsetneq DSPACE(S_2(n))$$

Dôkaz.

[1] Veta hovorí v princípe o tom, že ak máme viac pamäte, rozpoznáme zložitejšie jazyky.

Def. prefixový kód Turingovho stroja M je reťazec tvaru $1\,1\cdots 1\langle M\rangle$ [2]

Nech $\overline{M}$ je deterministický Turingov stroj (DTS), ktorý pracuje nasledovne:

(i) Na vstupe w dĺžky n označí na pracovných páskach pamäť rozsahu $S_2(n)$.
Ak by mal $\overline{M}$ v ďaľšom priebehu výpočtu použiť väčšiu pamäť ako $S_2(n)$, potom výpočet zastaví a vstup neakceptuje.

(ii) Ak vstup w nie je prefixový kód žiadneho Turingovho stroja, potom $\overline{M}$ neakceptuje w.

(iii) Nech w je prefixový kód nejakého Turingovho stroja M. Potom $\overline{M}$ (rešpektujúc *(i)*) simuluje $2^{S_2(n)}$ krokov výpočtu stroja M na w, a $\overline{M}$ akceptuje w iff M neakceptuje w počas $2^{S_2(n)}$ krokov.

Nech M je ľub. deterministický T-stroj s pamäťovou zložitosťou $S_1(n)$. Dokážeme $L(M) \neq L(\overline{M})$ a teda ukážeme, že $L(\overline{M}) \in DSPACE(S_2(n)) - DSPACE(S_1(n))$.

Nech M má k pások, t páskových symbolov a q stavov. Potom sa M na vstupe dĺžky n buď zastaví počas $\leq T(n) = (n+2)q[(S_1(n)+1)t^{S_1(n)}]^k$ krokov, alebo sa dostane do nekonečného cyklu z dôvodu opakovania sa nejakých konfigurácií.

Nech w je dostatočne dlhý prefixový kód stroja M, kde pre $n = |w|$ platí: $\overline{M}$ je na vstupe w schopný s pamäťou $S_2(n)$ simulovať $2^{S_2(n)}$ krokov výpočtu stroja M na vstupe w a $T(n) \leq 2^{S_2(n)}$ [3]; také w existuje, lebo $\log(n) \leq S_1(n) = o(S_2(n))$.

Z *(iii)* $\Rightarrow$ $\overline{M}$ akceptuje w iff M neakceptuje w.

$$\Rightarrow L(M) \neq L(\overline{M})$$

Poznámka 2.2.1. $S_2(n)$ musí byť páskovo konštruovateľná, aby sme si ju vedeli ustrážiť pri $\overline{M}$, $S_1(n)$ nemusí byť.

$\square$

Dôsledok 2.2.1. Keďže $S(n) = 2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}$ (umocnené n-krát) je páskovo konštruovateľná, existuje jazyk na rozpoznanie ktorého stačí pamäť $S(n)$, ale nestačí pamäť (a pochopiteľne ani čas) $S(n)/\log^* n$.

2.3 Veta o časovej hierarchii

Definícia 2.3.1. Funkcia $T(n)$ je časovo konštruovateľná, ak existuje deterministický T-stroj ktorý na každom vstupe dĺžky n vykoná práve $T(n)$ krokov.

Veta [o časovej hierarchii]. *Nech $n \leq T_1(n) = o(T_2(n)/\log T_1(n))$, kde $T_2(n)$ je časovo konštruovateľná. Potom $DTIME(T_1(n)) \subsetneq DTIME(T_2(n))$.*

2.4 Savitchova veta

Veta [Savitch]. *Nech $S(n) \geq \log(n)$ je páskovo konštruovateľná, potom*

$$NSPACE(S(n)) \subseteq DSPACE(S^2(n))\,.$$

[2]Tak vieme získať ľubovoľne dlhý kód, čo je výhodné z technických dôvodov.

[3]pre dostatočne veľké n to postačí, lebo $S_2(n)$ rastie rýchlejšie ako $S_1(n)$ a dá sa využiť $\log(n) \leq S_1(n)$ z predpoladov vety na dôkaz tejto nerovnosti pre veľké n.

Dôkaz. Nech M je ľubovoľný nedeterministický T-stroj (s pamäťou $S(n)$) majúci k-pások, q-stavov a t-páskových symbolov.
Na vstupe dĺžky n sa M môže dostať do najviac $\leq q(n+2)[(S(n)+1)t^{S(n)}]^k \leq c^{S(n)}$ rôznych konfigurácií, pre vhodnú konštantu c (nazvime ich $S(n)$-*ohraničené*). $\Rightarrow$ ak $x \in L(M)$, potom existuje akceptačný výpočet stroja M na x majúci najviac $c^{S(n)}$ krokov.

Aby sme zistili, či sa M dostane z konfigurácie C_1 do konfigurácie C_2 počas najviac i krokov výpočtu, stačí zistiť, či sa M dostane z C_1 do nejakej konfigurácie C_3 počas $\leq \lfloor \frac{i}{2} \rfloor$ krokov, a z C_3 do C_2 počas $\leq \lceil \frac{i}{2} \rceil$ krokov.

Algoritmus pre simuláciu stroja M:
begin nech x je vstup dĺžky n a nech C_0 je príslušná počiatočná konf. stroja M.
  pre každú $S(n)$-*ohraničenú* akceptačnú konfiguráciu C_F
    if `TEST` $(C_0, C_F, c^{S(n)})$ **then** `accept`.
end.

Procedúra `TEST` (C_1, C_2, i) [4]
if $C_1 = C_2$ **alebo** M prejde z C_1 do C_2 v jednom kroku
    then return `áno`.
if $i > 1$ **then**
  begin pre každú $S(n)$-*ohraničenú* konf. C_3 **vykonaj**
    if `TEST` $(C_1, C_3, \lceil \frac{i}{2} \rceil)$ **and** `TEST` $(C_3, C_2, \lfloor \frac{i}{2} \rfloor)$
      then return `áno`.
  end.
return `nie`.
end.

Uvedený algoritmus možno implementovať na deterministickom T-stroj M' s pamäťou $O(S^2(n))$, ktorý jednu z pások používa ako zásobník (kvôli rekurzii), ktorý je zväčšený o 1 blok rozsahu $O(S(n))$ pri každom volaní procedúry `TEST`.
Do bloku sú zapisované tieto hodnoty: C_1, C_2, C_3, i, `TEST` $(C_1, C_3, \lceil \frac{i}{2} \rceil)$, `TEST` $(C_3, C_2, \lfloor \frac{i}{2} \rfloor)$ a adresa návratu (t.j. pre volanie medzi „**if**ä „**and**", alebo medzi „**and**ä „**then**", alebo z hlavného programu).

Aký veľký blok teda vlastne potrebujeme? Stačí blok rozsahu $O(S(n))$ (C_i sú rádovo veľkosti $S(n)$, maximálna hodnota i je $c^{S(n)}$, čo vieme zapísať do rádovo $S(n)$ cifier).

Po vykonaní jedného volania procedúry `TEST` je príslušný blok zo zásobníka odstránený. Koľko najviac blokov na zásobníku budeme mať počas simulácie? Keďže pri volaní procedúry `TEST` je hodnota i (tretieho parametra) zmenšená zhruba na polovicu, hĺbka rekurzie (a teda aj počet blokov v zásobníku) je najviac $\leq 1 + \log_2 c^{S(n)}$, čo je rádovo $O(S(n))$.
$\Rightarrow M'$ je Turingov stroj s páskovou zložitosťou $O(S^2(n))$, ktorý možno prerobiť na iný Turingov stroj s páskovou zložitosťou $S^2(n)$ akceptujúci ten istý jazyk (kompresiou pásky).

□

2.5 Ďaľšie vety

Veta 2.5.1. (a) $DTIME(T(n)) \subseteq NTIME(T(n)) \subseteq DSPACE(T(n))$ *pre každú časovo konštruovateľnú* $T(n)$.
(b) $DSPACE(S(n)) \subseteq NSPACE(S(n)) \subseteq \bigcup_c DTIME(c^{\log n + S(n)})$ *pre každú páskovo konštruovateľnú* $S(n)$.

[4] Procedúra `TEST` zistí, či M prejde z C_1 do C_2 počas $\leq i$ krokov.

Dôkaz (pre $NTIME(T(n)) \subseteq DSPACE(T(n))$). Nech M je ľubovoľný nedeterministický Turingov stroj s časovou zložitosťou $T(n)$. Nech d je taká konštanta, že v každom kroku výpočtu M existuje najviac d možností pre ďaľší krok. Nech M' je deterministický T-stroj s pamäťovou zložitosťou $T(n)$, ktorý na vstupe x dĺžky n na jednej z pások postupne generuje (v lexikografickom poradí) všetky možné reťazce dĺžky najviac $T(n)$ nad nejakou d-písmenkovou abecedou $\{a_1, a_2, ..., a_d\}$. Vždy, keď M' vygeneruje ďaľší reťazec, simuluje potom možný výpočet stroja M na vstupe x s nedeterministickými krokmi reprezentovanými vygenerovaným reťazcom, a ak M v tomto výpočte akceptoval x, potom aj M' akceptuje x. A ak M neakceptoval x v žiadnom výpočte, potom ani M' neakceptuje x. $\Longrightarrow L(M) = L(M')$. $\square$

Dôkaz (pre $NSPACE(S(n)) \subseteq \bigcup_c DTIME(c^{\log n + S(n)})$). Nedeterministický T-stroj M s pamäťovou zložitosťou $S(n)$, ktorý má q stavov, t páskových symbolov a k pások, sa môže dostať počas všetkých možných výpočtov na vstupe x dĺžky n do najviac $(n+2)q[(S(n)+1)t^{S(n)}]^k \leq d^{\log n + S(n)}$ rôznych konfigurácií pre vhodnú konštantu d.

Nech G je orientovaný graf, ktorého vrcholy sú možné konfigurácie stroja M na x, a ktorého hrany sú dvojice konfigurácií (C_i, C_j) takých, že M prejde na x z C_i do C_j v 1 kroku.

$\Longrightarrow M$ akceptuje x s pamäťou $S(|x|)$ iff v G existuje cesta z počiatočnej konfigurácie do niektorej akceptačnej konfigurácie. To sa dá zistiť v polynomiálnom čase (vzhľadom na počet vrcholov grafu G) napríklad *Dijkstrovým algoritmom.* Vhodnou implementáciou takého algoritmu na T-stroji možno zistiť uvedený grafový problém v čase $\leq (d^{\log n + S(n)})^m = c^{\log n + S(n)}$ pre $c = d^m$ a vhodné m. $\square$

2.6 Dôsledok predchádzajúcich viet

Dôsledok 2.6.1.

$$DSPACE(\log n) \subseteq NSPACE(\log n) \subseteq P \subseteq$$
$$\subseteq NP \subseteq PSPACE = NPSPACE \subseteq EXP \subseteq NEXP$$

Poznámka 2.6.1. $NSPACE(\log n) \subseteq P$ vyplýva z časti (b) predchádzajúcej vety.

Poznámka 2.6.2. $NP \subseteq PSPACE$ vyplýva z časti (a) predchádzajúcej vety.

Poznámka 2.6.3. $PSPACE = NPSPACE$ je dôsledkom *Savitch*ovej vety.

Poznámka 2.6.4. $NPSPACE \subseteq EXP$ vyplýva z časti (b) predchádzajúcej vety.

Poznámka 2.6.5. Nie je známe, či $DSPACE(\log n) \subsetneq NP$.

Poznámka 2.6.6. Podobne tiež nie je známe, či $P \subsetneq PSPACE$.

Poznámka 2.6.7. Naproti tomu sa vie, že $NSPACE(\log n) \subsetneq PSPACE$ a $P \subsetneq EXP$. Vyplýva to zo Savitchovej Vety a z Viet o pamäťovej a časovej hierarchii.

2.7 Translačná lema

Tvrdenie 2.7.1. *Ak $NSPACE(n^4) \subseteq NSPACE(n^3)$, potom:*

$$NSPACE(n^{20}) \subseteq NSPACE(n^{15}).$$

Dôkaz. Nech L_1 je ľubovoľný jazyk z triedy $NSPACE(n^{20})$, t.j. existuje nedeterministický T-stroj M_1 akceptujúci L_1 s pamäťou n^{20}.

Nech $L_2 = \{x\#^i \mid x \in L_1 \wedge |x\#^i| = |x|^5\}$. Nech M_2 je nedeterministický T-stroj, ktorý na vstupe $x\#^i$ robí nasledovné:

- najprv overí, či $|x\#^i| = |x|^5$.

- ak „áno", potom sa na vstupe $x\#^i$ správa rovnako, ako M_1 na vstupe x.

$\Rightarrow M_2$ akceptuje L_2 s pamäťou n^4, lebo $|x\#^i| = (|x|^5)^4 = |x|^{20}$ a M_1 akceptuje L_1 s pamäťou n^{20}.
$\Longrightarrow L_2 \in NSPACE(n^4)$. Z predpokladu *Tvrdenia* $\Rightarrow L_2 \in NSPACE(n^3)$, tj. $\exists$ nedeterministický T-stroj M_3 akceptujúci L_2 s pamäťou n^3.

Nech M_4 je nedeterministický T-stroj, ktorý na vstupe x:

- najprv si zostrojí (na niektorej pracovnej páske) reťazec $x\#^i$ tak, aby $|x\#^i| = |x|^5$.
- potom sa na „vstupe"$x\#^i$ správa rovnako, ako M_3 na $x\#^i$.

M_4 akceptuje L_1 s pamäťou n^{15}, lebo $|x|^{15} = (|x|^5)^3 = |x\#^i|^3$ a naviac M_3 akceptuje L_2 s pamäťou n^3.
$\Longrightarrow L_1 \in NSPACE(n^{15})$. □

Translačná lema. *Nech $S_1(n)$, $S_2(n)$ a $f(n)$ sú páskovo konštruovateľné, $S_2(n) \geq n, f(n) \geq n$.*
Potom ak $NSPACE(S_1(n)) \subseteq NSPACE(S_2(n)) \Longrightarrow$
$\Longrightarrow NSPACE(S_1(f(n))) \subseteq NSPACE(S_2(f(n)))$.

Tvrdenie 2.7.2. *$NSPACE(n^3) \subsetneq NSPACE(n^4)$.*

Dôkaz. Nech by $NSPACE(n^4) \subseteq NSPACE(n^3)$. Aplikáciou translačnej lemy pre $f(n) = n^3$, $f(n) = n^4$, $f(n) = n^5$ postupne dostaneme:

$$\left.\begin{array}{r} NSPACE(n^{12}) \subseteq NSPACE(n^9) \\ NSPACE(n^{16}) \subseteq NSPACE(n^{12}) \\ NSPACE(n^{20}) \subseteq NSPACE(n^{15}) \end{array}\right\} \Rightarrow NSPACE(n^{20}) \subseteq NSPACE(n^9)$$

$$\left.\begin{array}{r} NSPACE(n^{20}) \subseteq NSPACE(n^9) \\ \text{Savitchova veta} \Rightarrow NSPACE(n^9) \subseteq NSPACE(n^{18}) \\ \text{Veta o pamäťovej hierarchii} \Rightarrow DSPACE(n^{18}) \subsetneq DSPACE(n^{20}) \end{array}\right\} \Rightarrow$$

$$\Rightarrow NSPACE(n^{20}) \subsetneq DSPACE(n^{20}) \subseteq NSPACE(n^{20}) \ \underline{\textit{spor!}}$$ □

Veta 2.7.3. *Ak $\varepsilon > 0$ a $r \geq 0$ potom*

$$NSPACE(n^r) \subsetneq NSPACE(n^{r+\varepsilon})$$

2.8 Veta o medzere

Tvrdenie 2.8.1. *Existuje rekurzívna funkcia* [5] *$S(n) \geq n$, pre ktorú:*

$$DSPACE(S(n)) = DSPACE(2^{S(n)}).$$

Dôkaz. Nech $M_1, M_2, M_3, ...$ je ľubovoľné efektívne očíslovanie všetkých deterministických T-strojov (tj. z čísla i vieme algoritmicky zostrojiť M_i).

Algoritmus pre výpočet $S(n)$: Nech $S_1 < S_2 < ... < S_p$ sú všetky rôzne veľkosti použitej pamäte v čase zastavenia alebo zacyklenia sa niektorého $M_i, i \leq n$, na niektorom $x \in \{0,1\}^n$.

Simulujeme „paralelne"výpočet všetkých $M_i, i \leq n$, na všetkých $x \in \{0,1\}^n$, aby sme postupne zisťovali hodnoty z $\{S_1, ..., S_p\}$ a aby sme našli číslo $m \geq n$ (m bude n alebo niektoré z priebežne zistených čísel z $\{S_1, ..., S_p\}$) t.ž. pre m platí:

[5] pričom nepredpokladáme nič o jej páskovej konštruovateľnosti.

$$(*)\ \ S_i \leq m \ \text{ alebo } \ 2^m < S_i,\ \ \forall i, 1 \leq i \leq p.$$

(T.j. Ak $S_i \leq 2^m$, potom $S_i \leq m$ pre každé i, $1 \leq i \leq p$).

Algoritmus nájde vhodné m pomocou nasledovného testu:

Ak v simulovanom kroku t pre dané m a $\forall M_i, i \leq n$ a $\forall x \in \{0,1\}^n$ platí: M_i sa na x počas t krokov zastavil alebo zacyklil s použitou pamäťou najviac m, alebo M_i na x mal už v kroku t použitej pamäte viac ako 2^m, potom zrejme pre m platí (*); v takom prípade $S(n) = m$ a algoritmus skončí.

Keďže každý $M_i, i \leq n$, na každom $x \in \{0,1\}^n$:

- sa niekedy zastaví alebo zacyklí (s použitou pamäťou $\leq S_p$), alebo
- niekedy použije pamäť $> 2^{\max(n,S_p)}$.

musí existovať krok, v ktorom je splnený vyššie uvedený test pre $m = \max(n, S_p) \Rightarrow$ Algoritmus sa zastaví najneskôr pri simulovaní takéhoto kroku. $\Longrightarrow$ Funkcia $S(n)$ je rekurzívna.

Nech $L \in DSPACE(2^{S(n)}) \Rightarrow \exists M_k$ akceptujúci L s pamäťou $\leq 2^{S(n)}$. Nech $n \geq k, x \in \{0,1\}^n$. Keďže M_k je deterministický T-stroj s pamäťou $\leq 2^{S(n)}$, musí sa na x zastaviť s použitou pamäťou $S_j \leq 2^{S(n)} = 2^m$, pre nejaké $1 \leq j \leq p$. Z (*) $\Rightarrow S_j \leq m = S(n)$. $\Rightarrow M_k$ sa zastaví na každom x dĺžky $\geq k$ s pamäťou $\leq S(n)$. $\Longrightarrow$ $L \in DSPACE(S(n))$. □

Veta [Gap Theorem]. *Pre každú rekurzívnu funkciu $g(n) \geq n$ existuje rekurzívna funkcia $S(n) \geq n$ taká, že platí:*

$$DSPACE(S(n)) = DSPACE(g(S(N))).$$

Poznámka 2.8.1. Podobné výsledky platia aj pre $NSPACE$, $DTIME$, $NTIME$.

Veta [Speedup Theorem]. *Pre každú rekurzívnu funkciu $f(n) \geq n^2$ existuje rekurzívny jazyk L taký, že pre ľubovoľný T-stroj M akceptujúci L v čase $T(n)$ existuje T-stroj M' tiež akceptujúci L, v čase $T'(n)$ tak, že $f(T'(n)) \leq T(n)$ pre skoro všetky n.*

Poznámka 2.8.2. Napríklad pre $f(n) = 2^n : 2^{T'(n)} \leq T(n)$, t.j. $T'(n) \leq \log T(n)$.

Kapitola 3

NP-úplné problémy

Definícia 3.0.1. $L \subseteq \Sigma^*$ je polynomiálne transformovateľný na $L_0 \subseteq \Sigma_0^*$, ak existuje polynóm $p(n)$ a deterministický T-stroj M s časovou zložitosťou $p(n)$, ktorý vstup $x \in \Sigma^*$ pretransformuje na vstup $y \in \Sigma_0^*$ taký, že $x \in L \Leftrightarrow y \in L_0$.

Definícia 3.0.2. L_0 je NP-úplný, ak $L_0 \in NP$ a každý $L \in NP$ je polynomiálne transformovateľný na L_0.

3.1 Cookova-Levinova veta

Definícia 3.1.1. SAT je množina kódov všetkých splniteľných booleovských výrazov (SAT je jazyk nad abecedou $\{\neg, \vee, \wedge, \Rightarrow, \Leftrightarrow, (,), 0, 1\}$).

Poznámka 3.1.1. Booleovské výrazy obsahujúce booleovské premenné, logické spojky $\neg, \vee, \wedge, \Rightarrow, \Leftrightarrow$ a zátvorky $(,)$ budeme kódovať tak, že jednotlivé premenné očíslujeme číslami 1, 2, ... a tieto premenné v booleovskom výraze nahradíme ich číslami v binárnom tvare.

Príklad 3.1.1. Booleovský výraz $(p \vee q) \Rightarrow (\neg p \vee q \wedge r)$ má kód $(1 \vee 10) \Rightarrow (\neg 1 \vee 10 \wedge 11)$.

Veta [Cook-Levin]. *SAT je NP-úplný jazyk.*

Dôkaz. Je zrejmé, že $SAT \in NP$. Nech $L \subseteq \Sigma^*$ je ľubovoľný jazyk z NP. Dá sa ukázať, že pre L existuje polynóm $p(n)$ a nedeterministický T-stroj $M = (\Sigma', Q, \delta, q_0, q_F)$ akceptujúci L v čase $p(n)$, pričom M nemá pracovné pásky, ale môže prepisovať symboly na vstupnej (smerom doprava nekonečnej) páske.

Ľubovoľný výpočet stroja M na vstupe $x \in \Sigma^*$ majúci $p(|x|)$ krokov možno reprezentovať reťazcom $C_0 C_1 \dots C_{p(|x|)}$ (nad abecedou $\Sigma' \cup Q$), kde $|C_i| = p(|x|) + 2$ pre $\forall i$ a C_i je konfigurácia $\cent u_i q^i v_i$, v ktorej je M v i-tom kroku (t.j. M je v stave $q^i \in Q$, v prvých $p(|x|) + 1$ políčkach pásky je slovo $\cent u_i v_i$ a hlava číta najpravejší symbol slova u_i).

Nech $x \in \Sigma^*$, $|x| = n$ a D je ľub. reťazec (nad $\Sigma' \cup Q$) dĺžky $(p(n) + 1)(p(n) + 2)$, t.j. $D = D_0 D_1 \dots D_{p(n)}$, $|D_i| = p(n) + 2$ pre $\forall i$. Reťazec D reprezentuje nejaký akceptačný výpočet dĺžky $p(n)$ stroja M na x práve vtedy, keď:

(1) D_0 je počiatočná konfigurácia $q_0 x B^{p(n) - |x|}$ (B je „blank"symbol).

(2) D_{i+1} je konfigurácia do ktorej M môže prejsť v jednom kroku z D_i, $i = 0, 1, \dots, p(n) - 1$.

(3) $D_{p(n)}$ je niektorá akceptačná konfigurácia.

Náš cieľ je pre dané x, M, p zostrojiť (v polynomiálnom čase vzhľadom ku $|x|$) booleovský výraz $E_x = F \wedge G \wedge H \wedge I$, ktorý bude splniteľný (t.j. $E_x \in SAT$) práve vtedy, keď existuje reťazec D spĺňajúci (1), (2), (3) (t.j. $x \in L$). Tým dokážeme, L je polynomiálne transformovateľný na SAT.

Konštrukcia E_x:

E_x bude obsahovať booleovské premenné $y_{j,s}$, $1 \leq j \leq (p(n)+1)(p(n)+2)$ a $s \in \Sigma' \cup Q$. Interpretácia každej premennej $y_{j,s}$ je taká, že $y_{j,s} = 1$ práve vtedy, keď j-ty symbol reťazca D je symbol s.

Zostrojíme výrazy F, G, H, I tak, aby:

(a) F bol splniteľný práve vtedy, keď pre všetky j $(1 \leq j \leq (p(n)+1)(p(n)+2))$ *práve jedna* premenná $y_{j,s}$ $(s \in \Sigma' \cup Q)$ má hodnotu 1 — t.j. takéto hodnoty premenných $y_{j,s}$ určujú *práve jeden* reťazec D (nad $\Sigma' \cup Q$) dĺžky $(p(n)+1)(p(n)+2)$ a obrátene.

(b) výraz G resp. H resp. I „kontroloval", či reťazec D určený hodnotami premenných $y_{j,s}$ (spĺňajúci F) má vlastnosť (1) resp. (2) resp. (3).

Konštrukcia F: $F := F_1 \wedge F_2 \wedge \cdots \wedge F_h$, $h = (p(n)+1)(p(n)+2)$

$$F_j := \Big(\bigvee_{s \in \Sigma' \cup Q} y_{j,s}\Big) \wedge \neg\Big(\bigvee_{\substack{s \neq r \\ s,r \in \Sigma' \cup Q}} (y_{j,s} \wedge y_{j,r})\Big), \quad j = 1, 2, \ldots, h$$

Pre dané j člen $\bigvee_{s \in \Sigma' \cup Q} y_{j,s}$ [resp. člen $\neg(\bigvee_{\substack{s \neq r \\ s,r \in \Sigma' \cup Q}} (y_{j,s} \wedge y_{j,r}))$] zabezpečuje aby aspoň [resp. najviac] jedna premenná $y_{j,t}$ $(t \in \Sigma' \cup Q)$ mala hodnotu 1.

Konštrukcia G: Nech $x = x_1 x_2 \ldots x_n$, $x_i \in \Sigma$ pre $\forall i$.

$$G := y_{1,\cent} \wedge y_{2,q_0} \wedge y_{3,x_1} \wedge y_{4,x_2} \wedge \cdots \wedge y_{n+2,x_n} \wedge y_{n+3,B} \wedge \cdots \wedge y_{p(n)+2,B}$$

t.j. G skutočne kontroluje, či D_0 je počiatočná konfigurácia $\cent q_0 x_1 x_2 \ldots x_n B^{p(n)-n}$.

Konštrukcia I:

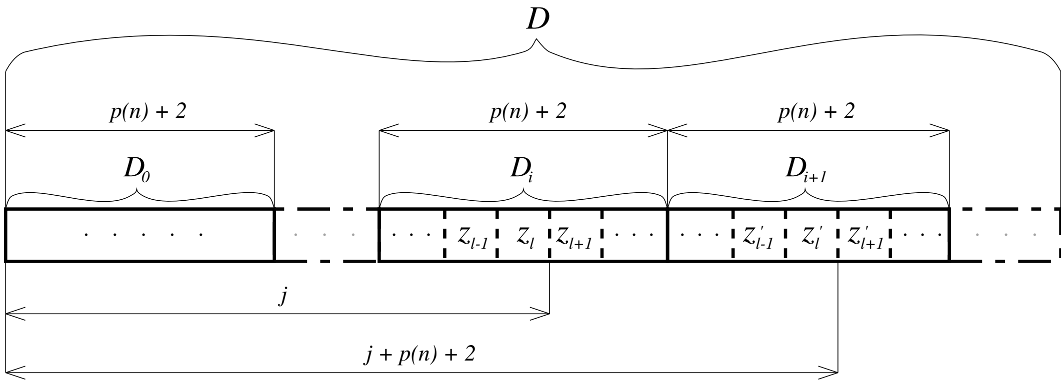
$$I := y_{t,q_F} \vee y_{t+1,q_F} \vee \cdots \vee y_{t+p(n)+1,q_F}, \quad t = p(n)(p(n)+2)+1$$

t.j. I kontroluje, či $D_{p(n)}$ obsahuje akceptačný stav q_F.

Konštrukcia H: Nech $D_i = z_1 z_2 \ldots z_{p(n)+2}$ je konf. stroja M a nech $D_{i+1} = z'_1 z'_2 \ldots z'_{p(n)+2}$ je reťazec (nad $\Sigma' \cup Q$). Reťazec D_{i+1} je konfigurácia, do ktorej sa môže M dostať v jednom kroku z konfigurácie D_i, čo je práve vtedy, keď pre každé l platí (*) a (**):

(*) Ak $z_{l-1}, z_l, z_{l+1} \in \Sigma'$ (t.j. symbol z_l sa nemôže zmeniť v jednom kroku, lebo hlava je dostatočne ďaleko od neho), potom $z'_l = z_l$.

(**) Ak $z_{l-1}, z_{l+1} \in \Sigma'$ a $z_l \in Q$ (t.j. z_{l-1}, z_l, z_{l+1} sa môžu zmeniť v jednom kroku), potom $z'_{l-1} z'_l z'_{l+1}$ je taký reťazec, ktorý môže vzniknúť z reťazca $z_{l-1} z_l z_{l+1}$ v jednom kroku v súlade s prechodovou funkciou δ stroja M.



Pre každý reťazec rqs ($r, s \in \Sigma'$, $q \in Q$) nech:

$$K_{rqs} := \{tuv \mid t, u, v \in \Sigma' \cup Q,\ tuv \text{ môže vzniknúť z } rqs \text{ v 1 kroku v súlade s } \delta \text{ (pozri (**))}\}$$

$$\text{Nech } H_i^* = \bigwedge_{i(p(n)+2)+1 \le j \le (i+1)(p(n)+2)} \left(\bigwedge_{r,s,t \in \Sigma'} (y_{j-1,r} \wedge y_{j,s} \wedge y_{j+1,t} \Rightarrow y_{j+p(n)+2,s}) \right)$$

$$H_i^{**} = \bigwedge_{\cdots \le j \le \cdots} \left(\bigwedge_{\substack{r,s \in \Sigma' \\ q \in Q}} \left(y_{j-1,r} \wedge y_{j,q} \wedge y_{j+1,s} \Longrightarrow \bigvee_{tuv \in K_{rqs}} \left(y_{j+p(n)+1,t} \wedge y_{j+p(n)+2,u} \wedge y_{j+p(n)+3,v} \right) \right) \right)$$

$$i = 0, 1, \cdots, p(n) - 1$$

Potom $H = H_0^* \wedge H_1^* \wedge \cdots \wedge H_{p(n)-1}^* \wedge H_0^{**} \wedge H_1^{**} \wedge \cdots \wedge H_{p(n)-1)}^{**}$ je výraz, ktorý kontroluje, či $D = D_0 D_1 \cdots D_{p(n)}$ má vlastnosť (2), pretože H_i^* resp. H_i^{**} kontroluje, či pre dvojicu D_i a D_{i+1} platí (*) resp. (**).

Dokážme teraz: M akceptuje x v čase $p(|x|)$ (t.j. $x \in L$) práve vtedy, keď $E_x = F \wedge G \wedge H \wedge I$ je splniteľný (t.j. $E_x \in SAT$).

Dôkaz. Nech M akceptuje x v čase $p(n)$, $n = |x| \Longrightarrow \exists D = D_0 D_1 \cdots D_{p(n)}$ také, že platí (1), (2) a (3). Z konštrukcie výrazov F, G, H, I sa dá nahliadnuť, že hodnota každého z nich je 1 pre hodnoty $y_{j,s}$ určujúcich D (t.j. pre $y_{j,s} = 1$ iff j-ty symbol reťazca D je symbol s) a teda $E_x = F \wedge G \wedge H \wedge I$ je splniteľný.

Obrátene: Nech E_x je splniteľný. Vyberme ľubovoľné hodnoty $y_{j,s}$ také, že hodnota výrazu E_x je 1 (t.j. hodnota každého z F, G, H, I je 1). Tieto hodnoty $y_{j,s}$ určujú $D = D_0 D_1 \cdots D_{p(n)}$. Z konštrukcie F, G, H, I vyplýva, že pre D platí (1), (2) a (3) a teda M akceptuje x v čase $p(|x|)$.

Nakoniec dokážme: L je polynomiálne transformovateľný na SAT.

Dôkaz. E_x obsahuje $(p(n)+1)(p(n)+2)|\Sigma' \cup Q|$ rôznych premenných $y_{j,s}$, ($n = |x|$), ktoré možno očíslovať binárnymi číslami dĺžky $O(\log p(n))$ a teda dĺžka kódu výrazu E_x je $O(p^2(n) \log p(n))$, t.j. $O(p^3(n))$, lebo každá $y_{j,s}$ sa v E_x vyskytuje najviac k-krát (pre ľubovoľne veľké n) (k závisí od Σ' a Q). Keďže E_x má „jednoduchú" štruktúru, možno jeho kód zostrojiť deterministickým algoritmom v polynomiálnom čase (vzhľadom ku $|x| = n$). Z toho vyplýva, že L je polynomiálne transformovateľný na SAT a teda SAT je NP-úplný jazyk.

□

3.2 Praktické NP-úplné problémy

Medzi známe NP-úplné problémy patria: SAT, $CSAT$, 3-SAT[1], Hamiltonovská kružnica, Vrcholové[2] pokrytie, Klika, Farbenie grafu k farbami, Problém obchodného cestujúceho (rozhodovacia verzia), 0-1 KNAPSACK (rozhodovacia verzia).

V tejto kapitole si ukážeme niekoľko ďalších praktických NP-úplných problémov.

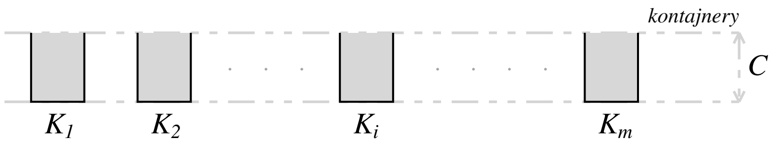
[1] 2-$SAT \in P$

[2] Hranové pokrytie $\in P$

3.2.1 Problém kontajnerov

Problém kontajnerov (Bin-Packing Problem): [3]

`Vstup:` $d_1, \cdots, d_n, m, C \quad (n, d_i, m, C \in \mathbb{N})$
`Otázka:` Možno čísla $d_1, \cdots, d_n$ rozdeliť do m disjunktných množín tak, aby ich suma v každej množine bola najviac C?



Príklad 3.2.1. Možno n objektov umiestniť do m kontajnerov s kapacitou C, kde d_i je rozsah i-teho objektu?

Príklad 3.2.2. Možno vykonať n programov na m počítačoch v čase najviac C, kde d_i je čas potrebný na vykonanie i-teho programu?

Príklad 3.2.3. Možno vyrobiť n súčiastok na m strojoch v čase najviac C, kde d_i je čas potrebný na výrobu i-tej súčiastky?

Poznámka 3.2.1. Problém kontajnerov je NP-úplný pre každé pevné $m \geq 2$.

3.2.2 Problém rozdelenia

Problém rozdelenia (Partition):

`Vstup:` $a_1, \cdots, a_n \quad (n, a_i \in \mathbb{N})$
`Otázka:` $\exists B \subseteq \{1, 2, \cdots, n\} : \quad \sum_{i \in B} a_i = \sum_{i \in \{1,2,\cdots,n\} - B} a_i$?

Poznámka 3.2.2. Problém rozdelenia je formálny jazyk:

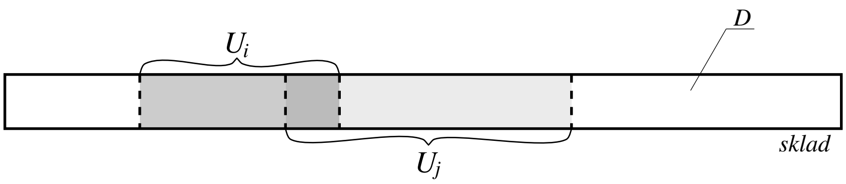
$$L = \Big\{ \text{kód}(a_1)\#\text{kód}(a_2)\# \cdots \#\text{kód}(a_n) \;\Big|\; n \in \mathbb{N},\ a_1, \cdots, a_n \in \mathbb{N}, \quad \exists B \subseteq \{1, 2, \cdots, n\} : \sum_{i \in B} a_i = \sum_{i \in \{1,2,\cdots,n\} - B} a_i, \quad \text{kde „kód}(a_i)\text{" je binárny zápis čísla } i \Big\}$$

Podobne by sme mohli formálne definovať jazyky aj pre ostatné NP-úplné problémy.

3.2.3 Riadenie skladu

Riadenie skladu (Dynamic Storage Allocation):

`Vstup:` $s_1, r_1, d_1, \cdots, s_n, r_n, d_n, D \quad (n, s_i, r_i, d_i, D \in \mathbb{N})$ [4]
`Otázka:` Existujú intervaly $U_1, \cdots, U_n$ (tvaru (a, b), $a, b \in \mathbb{N}_0$) také, že $|U_i| = s_i$ a $U_i \subseteq (0, D) \forall i$, a ak $U_i \cap U_j \neq \emptyset$ pričom $i \neq j$, potom buď $r_i \geq d_j$, alebo $r_j \geq d_i$? [5]

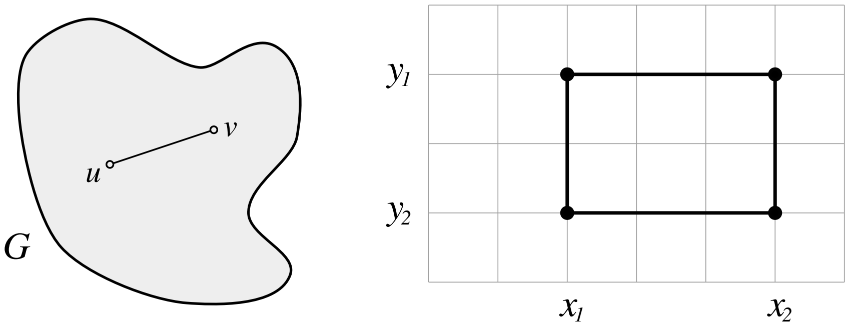


[3]Tiež **Problém hromadnej obsluhy** (Multiprocessor scheduling).
[4]kde n je počet tovarov na dočasné uskladnenie, D je kapacita skladu a s_i resp. r_i resp. d_i je veľkosť resp. čas príchodu do skladu resp. čas odchodu zo skladu i-teho tovaru.
[5]U_i vyznačuje umiestnenie i-teho tovaru.

3.2.4 Hranové vnáranie do mriežky

Hranové vnáranie do mriežky:

`Vstup:` Graf $G = (V, E)$, $m, n \in \mathbb{N}$.
`Otázka:` Existuje injektívna funkcia $f : V \to \{1, 2, \cdots, m\} \times \{1, 2, \cdots, n\}$ taká, že ak $(u, v) \in E$, $f(u) = (x_1, y_1)$ a $f(v) = (x_2, y_2)$, potom buď $x_1 = x_2$ alebo $y_1 = y_2$?



3.2.5 Problém dostatku registrov

Problém dostatku registrov:

`Vstup:` n-členná postupnosť príkazov typu „$x \leftarrow y$älebo „$x \leftarrow y$ `op` z", $K \in \mathbb{N}$.
`Otázka:` Stačí K registrov na výpočet danej postupnosti príkazov?

Príklad 3.2.4. $(a * b - c)/b + a/(a * b - c) \xrightarrow{\text{kompilátor}}$

$$\left.\begin{aligned} d &\leftarrow a * b \\ e &\leftarrow d - c \\ f &\leftarrow e/b \\ g &\leftarrow a/e \\ h &\leftarrow f + g \end{aligned}\right\} \text{ 8 registrov, stačia 4?} \qquad \left.\begin{aligned} d &\leftarrow a * b \\ d &\leftarrow d - c \\ c &\leftarrow d/b \\ b &\leftarrow a/d \\ c &\leftarrow c + b \end{aligned}\right\} \text{ 4 registre}$$

3.2.6 Konštrukcia krížovky

Konštrukcia krížovky:

`Vstup:` Konečná množina slov $W \subseteq \Sigma^*$, booleovská matica $A = (a_{ij})_{n \times n}$, $n \in \mathbb{N}$.
`Otázka:` Nech E je množina dvojíc (i, j) taká, že $a_{ij} = 0$. Existuje funkcia $f : E \to \Sigma$ taká, že postupnosť symbolov v maximálnej súvislej nulovej časti stĺpca/riadku tvorí slovo z W ?

3.2.7 Minimálna množina testov

Minimálna množina testov:

`Vstup:` Booleovská matica $A = (a_{ij})_{m \times n}$, $(m, n \in \mathbb{N})$, $K \in \mathbb{N}$, $K \leq n$.
`Otázka:` Možno z matice A vynechať $n - K$ stĺpcov tak, že všetky riadky výslednej matice budú navzájom rôzne ?

Príklad 3.2.5. Opravár televízorov vie, že nastala jedna zo štyroch možných porúch. Stačia mu dva testy z troch uvedených v tabuľke ?

	T_1	T_2	T_3
p_1	1	0	1
p_2	1	1	0
p_3	0	1	1
p_4	0	0	1

Kapitola 4

NP-optimalizačné problémy

4.1 Optimalizačné versus rozhodovacie problémy

Definícia 4.1.1. <u>**NP-optimalizačný problém**</u> A je daný cieľom (t.j. *min* alebo *max*), polynomiálne ohraničenou reláciou $R \subseteq \Sigma^* \times \Sigma^*$ (t.j. pre R existuje polynóm p taký, že ak $(x, y) \in R$, potom $|y| \leq p(|x|)$) a hodnotovou funkciou $m : \Sigma^* \times \Sigma^* \to \mathbb{N}$, pričom R aj m sú vypočítateľné deterministicky v polynomiálnom čase.

Presnejšie $\{x\#y \mid (x, y) \in R\} \in P$ a pre dvojicu $(x, y) \in R$ možno hodnotu funkcie $m(x, y)$ vypočítať deterministicky v čase $q(|x| + |y|)$, kde q je nejaký polynóm.

Poznámka 4.1.1. Relácia R je reláciou medzi "problémom" (vstupom) x, a "riešením" problému (výstupom) y. Funkcia m vyčísľuje hodnotu (kvalitu) riešenia.

Príklad 4.1.1. Optimalizačný problém obchodného cestujúceho:

- cieľ je *min* (hľadáme *najlacnejšiu* Hamiltonovskú kružnicu).
- $R = \{(x, y) \mid x$ je kompletný graf s ohodnotenými hranami, y je Hamiltonovská kružnica grafu $x\}$
- $m(x, y) =$ cena Hamiltonovskej kružnice y grafu x.

Pre NP-optimalizačný problém A ďalej definujeme:

Definícia 4.1.2. <u>**Konštrukčný problém**</u>: Pre dané x nájdi/zostroj (ak existuje) y také, že $(x, y) \in R$, pričom:

$$m(x, y) = \min_{y' \in \Sigma^*} \{m(x, y') \mid (x, y') \in R\} \text{ pre cieľ } min \text{ a podobne:}$$

$$m(x, y) = \max_{y' \in \Sigma^*} \{m(x, y') \mid (x, y') \in R\} \text{ pre cieľ } max.$$

Definícia 4.1.3. <u>**Hodnotový problém**</u>: Pre dané x urči (ak existuje):

$$\min_{y' \in \Sigma^*} \{m(x, y') \mid (x, y') \in R\} \text{ pre cieľ } min,$$

$$\max_{y' \in \Sigma^*} \{m(x, y') \mid (x, y') \in R\} \text{ pre cieľ } max.$$

Definícia 4.1.4. <u>**Rozhodovací roblém**</u>: Pre dané x a dané $k \in \mathbb{N}$ zisti, či:

$$\exists y : (x, y) \in R \wedge m(x, y) \leq k \text{ pre cieľ } min,$$

$$\exists y : (x, y) \in R \wedge m(x, y) \geq k \text{ pre cieľ } max.$$

Poznámka 4.1.2. Rozhodovací problém možno formulovať aj ako jazyk:

$$L = \{x\#d(k) \mid x \in \Sigma^*, k \in \mathbb{N},\ \exists y : (x,y) \in R \wedge m(x,y) \leq k\} \text{ pre cieľ } \mathit{min} \text{ a podobne:}$$

$$L = \{x\#d(k) \mid x \in \Sigma^*, k \in \mathbb{N},\ \exists y : (x,y) \in R \wedge m(x,y) \geq k\} \text{ pre cieľ } \mathit{max}.$$

kde $d(k)$ je dekadický/binárny zápis čísla k.

Lema 4.1.1. *Rozhodovací problém každého NP-optimalizačného problému patrí do NP.*

Dôkaz. Dôkaz vykonáme bez ujmy na všeobecnosti pre cieľ *min* (pre cieľ *max* je dôkaz podobný).

Ku danému x nedeterministicky „uhádneme" y, zistíme či $(x,y) \in R$ a ak áno, vypočítame hodnotu tohto riešenia pomocou funkcie $m(x,y)$. Nakoniec overíme či je hodnota $m(x,y)$ menšia alebo nanajvýš rovná číslu k. Všetky uvedené operácie je možné vykonať (nedeterministicky) v polynomiálnom čase, ako vidno priamo z definície NP-optimalizačného problému a preto jeho rozhodovací problém patrí do NP. □

Veta 4.1.2. *Nech A je ľubovoľný NP-optimalizačný problém s cieľom min*[1]*, reláciou R a hodnotovou funkciou m, ktorého rozhodovací problém:*

$$L = \{x\#d(k) \mid x \in \Sigma^*, k \in \mathbb{N},\ \exists y : (x,y) \in R \wedge m(x,y) \leq k\}$$

je NP-úplný. Nech možno L akceptovať deterministicky v čase $T(n)$. Potom konštrukčný problém pre A možno riešiť deterministicky v čase $r(n)T(s(n))$ pre vhodné polynómy r, s. [2] [3]

Dôkaz. (1) Pre dané x nájdeme (ak existuje) $K_x = \min_{y \in \Sigma^*}\{m(x,y) \mid (x,y) \in R\}$ takto:

Nech q je polynóm ohraničujúci čas výpočtu funkcie m. Po $q(|x|+|y|)$ krokoch výpočtu má hodnota $m(x,y)$ zapísaná v binárnom tvare najviac $q(|x|+|y|)$ číslic, t.j. $m(x,y) \leq 2^{q(|x|+|y|)} - 1 \leq 2^{q'(|x|)}$ pre vhodný polynóm q', nakoľko R je polynomiálne ohraničená a teda $|y| \leq p(|x|)$.

Pomocou algoritmu pre rozpoznávanie L, zisťujúc, či $x\#d(i) \in L$ pre vhodné čísla i, nájdeme $\min\{i \mid x\#d(i) \in L\} = K_x$ (ak existuje) binárnym prehľadávaním intervalu $\langle 0,\ 2^{q'(|x|)}\rangle$.

(2) Pre dané x a K_x zostrojíme (niektoré) optimálne riešenie y_x také, že $(x, y_x) \in R$ a $m(x, y_x) = K_x$ takto:

Nech $L' = \{x\#d(k)\#z | x \in \Sigma^*, k \in \mathbb{N}, \exists y : (x,y) \in R \wedge m(x,y) \leq k$ a z je prefix reťazca $y\}$. Dá sa ľahko nahliadnuť, že $L' \in NP$ (nedeterministicky „uhádneme" y a overíme všetky podmienky). Keďže L je NP-úplný, musí byť L' polynomiálne transformovateľný na L, presnejšie každý vstup $x\#d(k)\#z$ je polynomiálne transformovateľný na nejaké w, pričom $x\#d(k)\#z \in L' \iff w \in L$.

Pre jednoduchosť, nech $\Sigma = \{0,1\}$. Hľadaný reťazec y_x zostrojíme postupným predlžovaním už nájdeného prefixu z takto:

Ak $u_0 = x\#d(K_x)\#z0$ resp. $u_1 = x\#d(K_x)\#z1$ patrí do L', potom $z0$ resp. $z1$ je dlhší najdený prefix hľadaného y_x. To, či u_0 resp. u_1 patrí do L' možno zistiť tak, že u_0 resp. u_1 najprv transformujeme na w_0 resp. w_1 a potom zistíme, či w_0 resp. w_1 patrí do L. □

Dôsledok 4.1.1. (a) Veta 4.1.2 platí aj keď predpoklady „L je NP-úplný" a „L možno akceptovať deterministicky v čase $T(n)$" nahradíme predpokladom „nejaký NP-úplný jazyk L_0 možno akceptovať deterministicky v čase $T(n)$".

(b) Ak $P = NP$, potom konštrukčný problém každého NP-optimalizačného problému možno riešiť deterministicky v polynomiálnom čase.

[1] pre cieľ *max* analogicky.

[2] Inak povedané, ak platia predpoklady vety, potom sa dá konštrukčný problém pre A riešiť v iba o niečo horšom čase v porovnaní s $T(n)$, pretože $T(n)$ pravdepodobne nebude polynóm.

[3] Nie je známe, či táto veta platí aj bez predpokladu NP-úplnosti jazyka L.

Dôkaz. (a) Dôkaz je totožný s dôkazom Vety 4.1.2, až na to, že v časti (2) jej dôkazu treba nahradiť jazyk L jazykom L_0 a overenie toho, či $x\#d(i) \in L$ (v časti (1)) sa vykoná nasledovne: Keďže $L \in NP$ (viď Lema 4.1.1) a L_0 je NP-úplný, zrejme L je polynomiálne transformovateľný na L_0, z čoho vyplýva, že $x\#d(i) \in L \iff w \in L_0$, kde w je výsledok polynomiálnej transformácie vstupu $x\#d(i)$.

(b) Nech A je ľubovoľný NP-optimalizačný problém a L_0 je ľubovoľný NP-úplný jazyk. Ak $P = NP$, potom možno L_0 akceptovať deterministicky v čase $T(n)$, kde $T(n)$ je polynóm. Z (a) vyplýva, že konštrukčný problém pre A možno riešiť deterministicky v polynomiálnom čase $r(n)T(s(n))$. □

Príklad 4.1.2. Pre daný graf G nájdi (ak existuje) nejakú Hamiltonovskú kružnicu — formálne sa jedná o konštrukčný problém s reláciou:

$$R = \{(x, y) \mid \text{ graf } x \text{ má Hamiltonovskú kružnicu } y\},$$

$$m(x, y) = 1 \;\; \forall x, y, \;\; \text{cieľ je } \mathit{min} \text{ alebo } \mathit{max}.$$

Z Vety 4.1.2 (alebo z Dôsledku 4.1.1 časť (a)) vyplýva, že zostrojiť Hamiltonovskú kružnicu grafu G je zhruba rovnako ťažké ako zistiť, či G má Hamiltonovskú kružnicu.

Veta 4.1.3. *Nech A je ľubovoľný NP-optimalizačný problém, ktorého rozhodovací problém L možno akceptovať v deterministickom čase $T(n)$ (L nemusí byť NP-úplný). Potom hodnotový problém pre A možno riešiť deterministicky v čase $r(n)T(s(n))$ pre vhodné polynómy r, s.*

Dôkaz. Dôkaz je uvedený v časti (1) Dôkazu Vety 4.1.2. □

4.2 Aproximovateľnosť NP-optimalizačných problémov

Definícia 4.2.1. Nech A je NP-optimalizačný problém s cieľom *min* resp. cieľom *max*, reláciou R a hodnotovou funkciou m. Nech:

$$m^*(x) = \min_{y \in \Sigma^*} \{m(x, y) \mid (x, y) \in R\} \quad \text{pre cieľ } \mathit{min},$$

$$m^*(x) = \max_{y \in \Sigma^*} \{m(x, y) \mid (x, y) \in R\} \quad \text{pre cieľ } \mathit{max}.$$

Hovoríme, že NP-optimalizačný problém A je *α-aproximovateľný* ($\alpha > 1$), ak existuje deterministický polynomiálny algoritmus M, ktorý pretransformuje vstup x na výstup y (t.j. $M(x) = y$) s vlastnosťou $(x, y) \in R$, pričom:

$$\alpha m^*(x) \geq m(x, M(x)) \;\; \text{(pre skoro všetky } x\text{) pre cieľ } \mathit{min},$$

$$m^*(x) \leq \alpha m(x, M(x)) \;\; \text{(pre skoro všetky } x\text{) pre cieľ } \mathit{max}.$$

Definícia 4.2.2. Problém A je *dobre* aproximovateľný, ak je α-aproximovateľný pre každé $\alpha > 1$ ($\alpha \to 1$).

Definícia 4.2.3. Problém A je *neaproximovateľný*, ak nie je α-aproximovateľný pre žiadne $\alpha > 1$ ($\alpha \to \infty$).

Poznámka 4.2.1. Ak $P = NP$, potom konštrukčný problém každého NP-optimalizačného problému môžeme riešiť deterministicky v polynomiálnom čase[4], t.j. nemá význam uvažovať aproximovateľnosť.

Veta 4.2.1. *Ak $P \neq NP$, potom existujú NP-optimalizačné problémy A, B a C také, že:*

(i) A je dobre aproximovateľný, ale jeho rozhodovací problém nepatrí do P.

[4]Pozri **Dôsledok 4.1.1 (b)**.

(ii) B je α-aproximovateľný (pre nejaké $\alpha > 1$), ale nie je dobre aproximovateľný.

(iii) C je neaproximovateľný.

Dôkaz. A, B a C sú rovnaké, až na hodnotovú funkciu a sú definované takto: cieľ je *max*,

$$R = \{(x, y) \mid x \text{ je graf, } y \text{ je ľub. postupnosť všetkých vrcholov grafu } x \text{ (bez opakovania)}\}$$

a hodnotová funkcia m_A resp. m_B resp. m_C pre A resp. B resp. C je definovaná takto:

$$m_A(x,y) = \begin{cases} |x| & \text{, ak } y \text{ je hamiltonovská kružnica grafu } x \\ |x|-1 & \text{, inak} \end{cases}$$

$$m_B(x,y) = \begin{cases} 2 & \text{, ak } y \text{ je hamiltonovská kružnica grafu } x \\ 1 & \text{, inak} \end{cases}$$

$$m_C(x,y) = \begin{cases} |x| & \text{, ak } y \text{ je hamiltonovská kružnica grafu } x \\ 1 & \text{, inak} \end{cases}$$

Nech $\overline{M}$ je deterministický T-stroj, ktorý v polynomiálnom čase pretransformuje graf x na postupnosť y_x vrcholov grafu x (v nejakom ľubovoľnom usporiadaní), t.j. $\overline{M}(x) = y_x$.

(i)

A je *dobre* aproximovateľný, lebo:

$$m_A^*(x) = \max_{y \in \Sigma^*}\{m_A(x,y) \mid (x,y) \in R\} \leq |x| \leq \alpha(|x|-1) \leq \alpha m_A(x, \overline{M}(x))$$

pre každé $\alpha > 1$ a skoro všetky x.

Nech HAM je problém, či graf má hamiltonovskú kružnicu. Z definície $m_A(x,y)$ môžeme vidieť, že graf x má hamiltonovskú kružnicu práve vtedy, keď $\exists y : R(x,y) \wedge m_A(x,y) \geq |x|$, čo je práve vtedy, keď riešenie rozhodovacieho problému pre A je "áno"pre graf x a $k = |x|$.

Preto ak by sme vedeli deterministicky a v polynomiálnom čase riešiť rozhodovací problém pre A, potom by aj HAM bol riešiteľný v deterministickom polynomiálnom čase (t.j. $HAM \in P$), čo je spor s predpokladom vety $P \neq NP$, nakoľko vieme, že HAM je NP-úplný.

(ii)

B je 2-aproximovateľný, lebo:

$$m_B^*(x) = \max_{y \in \Sigma^*}\{m_B(x,y) \mid (x,y) \in R\} \leq 2 \leq 2m_B(x, \overline{M}(x)),$$

keďže $m_B(x,y) \in \{1, 2\}$.

Predpokladajme, že problém B je *dobre* aproximovateľný a teda napríklad $\frac{3}{2}$-aproximovateľný, presnejšie:

$$2m_B^*(x) \leq 3m_B(x, M(x)),$$

kde M je deterministický polynomiálny algoritmus z definície α-aproximovateľnosti. Pomocou algoritmu M a pomocou algoritmu pre m_B by sme vedeli deterministicky a v polynomiálnom čase vypočítať $m_B^*(x)$ takto:

• Ak $m_B(x, M(x)) = 1$, potom $m_B^*(x) = 1$ (pretože $m_B^*(x) \in \{1, 2\}$ a teda v tomto prípade $2m_B^*(x) \leq 3m_B(x, M(x)) = 3$).

• Ak $m_B(x, M(x)) = 2$, potom $m_B^*(x) = 2$ (pretože $2 = m_B(x, M(x)) \leq m_B^*(x) \leq 2$).

Z definície m_B vyplýva, že $2 = m_B^*(x) = \max_{y \in \Sigma^*}\{m_B(x,y) \mid (x,y) \in R\}$ práve vtedy, keď $\exists y : (x,y) \in R \wedge m_B(x,y) = 2$, čo je práve vtedy, keď graf x má hamiltonovskú kružnicu.

Preto ak by sme vedeli deterministicky a v polynomiálnom čase vypočítať $m_B^*(x)$, potom by $HAM \in P$, čo je spor s predpokladom $P \neq NP$.

(iii)

Predpokladajme, že C je α-aproximovateľný pre nejaké $\alpha > 1$, teda:

$$m_C^*(x) = \max_{y \in \Sigma^*}\{m_C(x,y) \mid (x,y) \in R\} \leq \alpha m_C(x, M(x)),$$

kde M je deterministický polynomiálny algoritmus z definície α-aproximovateľnosti. Pomocou algoritmu M a algoritmu pre m_C by sme vedeli deterministicky v polynomiálnom čase vypočítať aj $m_C^*(x)$ pre $|x| > \alpha$ takto:

- Ak $m_C(x, M(x)) = 1$, potom $m_C^*(x) = 1$ (lebo inak by $|x| = m_C^*(x) \leq \alpha m_C(x, M(x)) = \alpha$, čiže $|x| \leq \alpha$, čo je spor s predpokladom $|x| > \alpha$).
- Ak $m_C(x, M(x)) = |x|$, potom $m_C^*(x) = |x|$ (lebo inak by $1 = m_C^*(x) \geq m_C(x, M(x)) = |x|$, čo je spor s predpokladom $|x| > \alpha > 1$).

Z definície $m_C(x,y)$ vidíme, že pre $|x| > 1$ platí: $|x| = m_C^*(x) = \max_{y \in \Sigma^*}\{m_C(x,y) \mid (x,y) \in R\}$ práve vtedy, keď $\exists y : (x,y) \in R \wedge m_C(x,y) = |x|$, čo je práve vtedy, keď x má hamiltonovskú kružnicu.

Preto ak by sme vedeli deterministicky a v polynomiálnom čase vypočítať $m_C^*(x)$, potom by $HAM \in P$, čo je spor s predpokladom $P \neq NP$.

□

Veta 4.2.2. *Ak $P \neq NP$, potom NP-optimalizačný* „problém obchodného cestujúceho" (POC) *je neaproximovateľný.*

Dôkaz. Predpokladajme naopak, že POC je α-aproximovateľný pre nejaké α. Nech $G = (V, E)$ je ľubovoľný graf, a nech $G' = (V, V \times V, cena)$ je kompletný graf s ohodnotenými hranami taký, že:

$$cena(i,j) = \begin{cases} 1 & \text{, ak } (i,j) \in E \\ \alpha|V| + 1 & \text{, ak } (i,j) \notin E \end{cases}$$

Nech M je α-aproximačný deterministický polynomiálny algoritmus, ktorý z grafu G' zostrojí jeho Hamiltonovskú kružnicu s cenou $\leq \alpha m^*(G')$, t.j. $cena(G', M(G')) \leq \alpha m^*(G')$, kde $m^*(G')$ je cena najlacnejšej hamiltonovskej kružnice v G'. Funkcia *cena* je funkcia m z definície NP-optimalizačných problémov, a relácia R pre POC je rovnaká, ako v dôkaze Vety 4.2.1.

Platí: $m^*(G') \leq cena(G', M(G')) \leq \alpha m^*(G')$ a cena *každej* hamiltonovskej kružnice grafu G' je buď $|V|$ alebo aspoň $\alpha|V| + 1$ (pretože $cena(i,j) \in \{1, \alpha|V|+1\}$). Teda:

- Ak $m^*(G') = |V|$, potom $|V| = m^*(G') \leq cena(G', M(G')) \leq \alpha m^*(G') = \alpha|V|$.
- Ak $m^*(G') \geq \alpha|V| + 1$, potom $\alpha|V| + 1 \leq m^*(G') \leq cena(G', M(G'))$.

Suma sumárum, G má hamiltonovskú kružnicu práve vtedy, keď $m^*(G') = |V|$, čo je práve vtedy, keď $cena(G', M(G')) \leq \alpha|V|$. Môžeme teda uzavrieť, že pomocou algoritmu M by bolo možné riešiť HAM, čo je spor s $P \neq NP$. □

Veta 4.2.3. *Ak $P \neq NP$, potom:*

(1) „Maximálna klika grafu" *(jej nájdenie) a* „Najdlhšia cesta v grafe medzi dvoma danými vrcholmi" *sú neaproximovateľné NP-optimalizačné problémy.*

(2) „Bin-packing" *(problém kontajnerov) je $\frac{3}{2}$-aproximovateľný, ale nie je $(\frac{3}{2}-\varepsilon)$-aproximovateľný pre žiadne $\varepsilon > 0$ (pokiaľ $P \neq NP$).*

(3) „POC s trojuholníkovou nerovnosťou" *je $\frac{3}{2}$-aproximovateľný ale nie je dobre aproximovateľný.*

(4) „0-1 Knapsack" *(problém batoha) je dobre aproximovateľný NP-optimalizačný problém.*

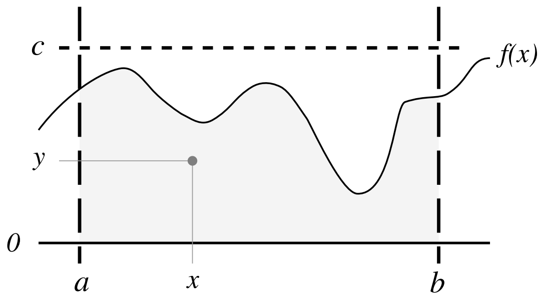
Kapitola 5

Pravdepodobnostné algoritmy

5.1 Výpočet určitého integrálu

Úvodom spomenieme snáď najznámejšiu a pritom najjednoduchšiu metódu pre výpočet určitého integrálu v tvare $\int_a^b f(x)\,\mathrm{d}x$ pravdepodobnostným *Monte-Carlo* algoritmom.

Budeme pre jednoduchosť predpokladať, že funkcia $f(x)$ je na intervale $\langle a, b\rangle$ nezáporná a zhora ohraničená konštantou c, t.j. $\forall x \in \langle a, b\rangle : 0 \leq f(x) \leq c$. Ďalej `rand` je generátor náhodných čísel s uniformnou distribúciou na intervale $\langle a, b\rangle$.



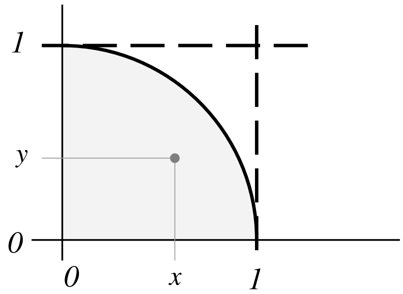
<u>**Algoritmus**</u> pre výpočet $\int_a^b f(x)\,\mathrm{d}x$:

$k \leftarrow 0$
for $i \leftarrow 1$ **to** n **do**
  begin
    $x \leftarrow \texttt{rand}(a, b)$
    $y \leftarrow \texttt{rand}(0, c)$
    if $y \leq f(x)$ **then** $k \leftarrow k + 1$
  end
$\mathit{integrál} \leftarrow kc(b - a)/n$.

Poznámka 5.1.1. Pre $0 < \varepsilon$, $\delta < 1$ platí: $|\mathit{integrál} - \int_a^b f(x)\,\mathrm{d}x| < \varepsilon$ s pravdepodobnosťou aspoň $(1 - \delta)$ pre $n \geq \lceil 1/(4\varepsilon^2\delta)\rceil$.

5.2 Výpočet čísla π

Spomenieme jednoduchý pravdepodobnostný algoritmus, ktorý myšlienkovo vychádza z uvedenej *Monte-Carlo* metódy pre výpočet určitého integrálu.



Algoritmus pre vyčíslenie π :

$k \leftarrow 0$
for $i \leftarrow 1$ **to** n **do**
  begin
    $x \leftarrow \texttt{rand}(0,1)$
    $y \leftarrow \texttt{rand}(0,1)$
    if $x^2 + y^2 \leq 1$ **then** $k \leftarrow k+1$
  end
$\pi \leftarrow 4k/n.$

5.3 Testovanie veľkých prvočísel

Je skutočnosťou, že zatiaľ nepoznáme dostatočne rýchly deterministický algoritmus na testovanie prvočíselnosti veľkých (v zmysle dekadicky aspoň 100-ciferných) čísel. Uvedieme jeden známy praktický algoritmus, ktorý sa snaží riešiť tento problém.

Miller-Rabinov test:
`Vstup:` nepárne $b > 2$, $r \in \mathbb{N}$.

Nech $[2^t \mid (b-1)] \wedge [2^{t+1} \nmid (b-1)]$
  for $i \leftarrow 1$ **to** r **do**
    begin
      vyber náhodne celé a $(1 \leq a \leq b-1)$
`1.`   **if** $x^2 \equiv 1 \pmod b$ **and** $x \not\equiv \pm 1 \pmod b$
        pre nejaké $x \in \{a^{(b-1)/2}, a^{(b-1)/4}, a^{(b-1)/8}, \cdots, a^{(b-1)/2^t}\}$
        then return „b `určite nie je prvočíslo`” (s absolútnou istotou)
`2.`   **if** $a^{b-1} \not\equiv 1 \pmod b$
        then return „b `určite nie je prvočíslo`” (s absolútnou istotou) [1]
    end
`3.` **return** „b `je prvočíslo`” (s pravdepodobnosťou omylu nanajvýš 2^{-r}).

Veta [Miller-Rabin]. *Ak b nie je prvočíslo, potom náhodne vybrané číslo a $(1 \leq a \leq b-1)$ spĺňa podmienku v príkaze* `1.` *alebo* `2.` *s pravdepodobnosťou aspoň* $\frac{1}{2}$.

Dôsledok 5.3.1. Tvrdenie v príkaze `3.` je pravdivé.

Dôvod. Pravdepodobnosť, že ani jedno z r náhodne vybratých čísel a nespĺňa podmienku v príkaze `1.` a ani podmienku v príkaze `2.` je nanajvýš $(\frac{1}{2})^r$. □

[1] na základe Fermatovej vety.

Poznámka 5.3.1. Existuje implementácia *Miller-Rabinovho* testu (založená na princípe procedúry `Mod-Exp`, ktorú upresníme v kapitole *Modulárna aritmetika* (6)) s časovou zložitosťou $O(rm^3)$, ak b je najviac m-bitové číslo.

Poznámka 5.3.2. Pre $m = 340$ (t.j. b je dekadicky 100-ciferné číslo) a $r = 50$:

$$rm^3 \approx 2.10^9, \quad 2^{-r} \approx 10^{-15}$$

5.4 Pravdepodobnostné generovanie veľkých prvočísel

Problémom, ktorým sa budeme zaoberať v tejto stati je nájdenie „efektívneho" spôsobu náhodného generovania veľkých (rádovo 100-ciferných) prvočísel. Na základe znalosti *Miller-Rabinovho* testu sa nám ponúka takýto algoritmus:

(1) Generátorom náhodných čísel vytvor náhodné 100-ciferné číslo b.

(2) Zisti, či b je prvočíslo (*Miller-Rabinov* test).

(3) Ak `nie`, opakuj (1) a (2) až do nájdenia prvočísla.

Poznámka 5.4.1. Algoritmus vykoná (1) a (2) v priemere 230-krát, lebo v priemere každé 230-te medzi 100-cifernými číslami je prvočíslo, keďže platí Erdösova veta:

$$\lim_{n\to\infty} \frac{P(n)}{n/\ln n} = 1,$$

kde $P(n)$ je počet prvočísel v intervale $\langle 2, n\rangle$ ($n \approx 10^{100}$, $\ln 10^{100} \approx 230$).

5.5 RSA šifrovací systém

Začnime stručnou charakteristikou šifrovacieho systému RSA a jeho vlastnosťami.

– Každý účastník U komunikačnej siete vlastní:

- verejný kľúč P_U,
- tajný kľúč S_U,

  pričom oba tieto kľúče si vytvára sám.

– Verejné kľúče sú *známe všetkým* účastníkom.

– Tajné kľúče si udržujú účastníci *v tajnosti*.

– Pre kľúče platí:

- P_U, S_U sú jedno-jednoznačné funkcie z D do D (kde D je množina správ).
- $S_U(P_U(M)) = M = P_U(S_U(M))$ pre ľubovoľnú správu z D.

$$M \longrightarrow \overset{A}{\boxed{P_B}} \xrightarrow{P_B(M)} \overset{B}{\boxed{S_B}} \longrightarrow S_B(P_B(M)) = M$$

5.5.1 Vytvorenie kľúčov

V tejto stati si popíšeme spôsob vytvárania verejného resp. tajného kľúča pre účastníka komunikačnej siete U.

(1) Vyber náhodné veľké (dekadicky 100-ciferné) prvočísla p, q.

(2) Vypočítaj $n = pq$, $\phi(n) = (p-1)(q-1)$.

(3) Vyber nepárne číslo e nesúdeliteľné s $\phi(n)$. Presnejšie, pomocou procedúry `Euklid` (pozri kapitolu *Modulárna aritmetika* (6)) nájdi $e \in \{3, 5, 7, \dots\}$, a celé x, y také, že $1 = \phi(n)x + ey$.

(4) Vypočítaj $d = e^{-1} \bmod \phi(n)$, t.j. prvok inverzný k e v zvyškovej triede čísel modulo $\phi(n)$, presnejšie: $[d \leftarrow y;\ \textbf{if}\ y < 0\ \textbf{then}\ d \leftarrow y + \phi(n)]$. Potom pre d platí $1 = (e \cdot d) \bmod \phi(n)$, $d \geq 0$.

(5) Nech $D = Z_n = \{0, 1, \dots, n-1\}$. Potom:

- $P_U(M) = M^e \bmod n$,
- $S_U(M) = M^d \bmod n$,

pre všetky správy $M \in D$.

Na výpočet $P_U(M)$ a $S_U(M)$ použi procedúru `Mod-Exp` (pozri kapitolu *Modulárna aritmetika*). Platí:

$$P_U(S_U(M)) = M = S_U(P_U(M)) \quad (\forall\, M \in D)$$

5.5.2 Bezpečnosť RSA

– AKO zistiť S_U? Keďže P_U (t.j. dvojica (e, n)) je známa, stačí zistiť d – potom budeme poznať (d, n), teda S_U).

– AKO zistiť d, keď poznáme e, n? Nie je známy "lepší" spôsob, než z čísla n zistiť p, q, následne $\phi(n)$ a na záver d (z čísel $\phi(n)$ a e).

– AKO zistiť p, q z čísla n ? *Ťažko!* (aspoň podľa doterajších skúseností, a na tom je založená bezpečnosť RSA).

Kapitola 6

Modulárna aritmetika

Lema 6.0.1. *Nech a, b, n sú najviac m-bitové čísla.* Jednoduché *algoritmy pre výpočet $a \cdot b$, $\lfloor a/b \rfloor$, $a \bmod n = a - \lfloor a/n \rfloor n$ vyžadujú $O(m^2)$ bitových operácií.*

6.1 Výpočet $a^b \bmod n$

procedure `Mod-Exp` (a, b, n):

```
    Nech b_m b_{m-1} ... b_0 je bin. reprezentácia čísla b.
    d ← 1
    for i ← m to 0 do
        begin
1.          d ← (d · d) mod n
2.          if b_i = 1 then d ← (d · a) mod n
        end
    return d.
```

Lema 6.1.1. *Procedúra* `Mod-Exp` *vráti $d = a^b \bmod n$ v čase $O(m^3)$, ak a, b, n sú najviac m-bitové čísla.*

Dôkaz. Nech $b = \sum_{i=0}^{m} b_i 2^i$, $b_i \in \{0, 1\}$ $\forall i$. Potom:

$$\begin{aligned} \lfloor b/2^i \rfloor &= b_m 2^{m-i} + b_{m-1} 2^{m-i-1} + \cdots + b_i 2^0 \\ \lfloor b/2^{i-1} \rfloor &= b_m 2^{m-i+1} + b_{m-1} 2^{m-i} + \cdots + b_i 2^1 + b_{i-1} 2^0 = 2\lfloor b/2^i \rfloor + b_{i-1} \end{aligned}$$

z čoho vyplýva:

$$a^{\lfloor b/2^{i-1} \rfloor} = a^{2\lfloor b/2^i \rfloor + b_{i-1}} = a^{\lfloor b/2^i \rfloor} \cdot a^{\lfloor b/2^i \rfloor} \cdot a^{b_{i-1}} \qquad (*)$$

Ak po vykonaní riadkov `1.` a `2.` s parametrom i platí $d = a^{\lfloor b/2^i \rfloor} \bmod n$, potom po vykonaní týchto riadkov s parametrom $i - 1$ platí $d = a^{\lfloor b/2^{i-1} \rfloor} \bmod n$, lebo:

$$\begin{aligned} (p \cdot q) \bmod n &= ((p \bmod n) \cdot (q \bmod n)) \bmod n, \\ (p \cdot q) \bmod n &= ((p \bmod n) \cdot q) \quad \bmod n, \end{aligned}$$

pre všetky p, q, n. Navyše z `(*)` vyplýva:

$$a^{\lfloor b/2^{i-1} \rfloor} \bmod n = \underbrace{(((\overbrace{(\underbrace{a^{\lfloor b/2^i \rfloor} \bmod n}_{d}) \cdot (\underbrace{a^{\lfloor b/2^i \rfloor} \bmod n}_{d})) \bmod n}^{\text{riadok 1.}}) \cdot a^{b_{i-1}}) \bmod n}_{\text{riadok 2.}}$$

Teda $d = a^{\lfloor b/2^0 \rfloor} \bmod n = a^b \bmod n$ po vykonaní `Mod-Exp` (a, b, n). □

6.2 Najväčší spoločný deliteľ – nsd(a, b)

Z Euklidovho algoritmu vieme, že $\text{nsd}(a, b) = \text{nsd}(b, a \bmod b)$.

Veta 6.2.1. $(\forall a \in \mathbb{N})(\forall b \in \mathbb{N}_0)$ *existujú celé* x, y *také, že* $\text{nsd}(a, b) = ax + by$, $|x| \leq b$, $y \leq a$.

Výpočet $d = \text{nsd}(a, b)$; x; y:

$\underline{\textbf{procedure}}$ $\texttt{Euklid}$ $(\overbrace{a}^{r_i}, \overbrace{b}^{s_i})$:

  if $b = 0$ **then return** $(a, 1, 0)$

  $(d, z, x) \leftarrow \texttt{Euklid}\ (\underbrace{b}_{r_{i+1}}, \underbrace{\overbrace{a}^{r_i} \bmod \overbrace{b}^{s_i}}_{s_{i+1}})$

  $y \leftarrow z - \lfloor a/b \rfloor x$

  return (d, x, y)

end

Lema 6.2.2. *Procedúra* $\texttt{Euklid}$ *vráti* (d, x, y) *také, že* $d = \text{nsd}(a, b) = ax + by$, $|x| \leq b$, $|y| \leq a$, *v čase* $O(m^3)$, *ak* a, b *sú najviac* m*-bitové čísla* ($a > 0$, $b \geq 0$).

Dôkaz. Dôkaz korektnosti tohto algoritmu vyplýva z algebry, a tak sa budeme zaoberať výlučne jeho časovou zložitosťou.

Nech r_i resp. s_i je hodnota prvého resp. druhého vstupného parametra procedúry $\texttt{Euklid}$ v i-tom rekurzívnom volaní. Nech je procedúra $\texttt{Euklid}$ volaná k-krát ($k \geq 4$). Potom platí:

(a) $1 \leq s_i < r_i$ pre $2 \leq i \leq k - 1$,

(b) $r_i/2 > r_{i+2}$ pre $2 \leq i \leq k - 2$.

Dôvod pre (a) : Keďže je procedúra volaná k-krát, musí platiť $s_k = 0$ a $s_i \geq 1$ pre $i < k$. Zrejme platí: $s_{i+1} = r_i \bmod s_i < s_i = r_{i+1}$.

Dôvod pre (b) : Uvažujme dva prípady:

(1) $s_i \leq r_i/2 \implies r_{i+2} = s_{i+1} = r_i \bmod s_i < s_i \leq r_i/2$.

(2) $s_i > r_i/2 \implies 2 > r_i/s_i > 1$ (lebo $r_i > s_i \geq 1$, pozri (a)) $\implies \lfloor r_i/s_i \rfloor = 1 \implies r_{i+2} = s_{i+1} = r_i \bmod s_i = r_i - \lfloor r_i/s_i \rfloor s_i = r_i - s_i < r_i/2$.

Z (b) vyplýva, že $k = O(\log r_2) = O(m)$, lebo a, b (a teda aj $r_2 = b$) sú najviac m-bitové čísla a počas každých dvoch volaní (počnúc druhým) sa hodnota prvého vstupného parametra zmenší aspoň na polovicu.

Teda celkový čas výpočtu je $O(m^3)$, lebo (možno dokázať, že) hodnoty vstupných aj výstupných parametrov sú v každom rekurzívnom volaní procedúry $\texttt{Euklid}$ najviac m-bitové čísla.

$\square$

Kapitola 7

Kolmogorovská zložitosť

Nech A_1, A_2, A_3, … je *efektívne očíslovanie* všetkých programov (napr. v C alebo T-stroje), t.j. z čísla i vieme algoritmicky zostrojiť A_i a obrátene, a $A_i : B^* \to B^*$ pre všetky i, kde $B = \{0, 1\}$.

Problém: Aké (minimálne) množstvo informácie postačí na *algoritmickú* konštrukciu daného objektu (reťazca) x?

Príklad 7.0.1. Ak x je rozvoj čísla π na n miest, stačí poznať n a algoritmus generujúci π na ľubovoľný počet miest.

Riešenie problému: Nájdi „najkratšiu" dvojicu (m, p) takú, že $A_m(p) = x$, kde $m \in \mathbb{N}$ a $p \in B^*$; dĺžka kódu dvojice (m, p) je množstvo postačujúcej informácie.

Definícia 7.0.1. Nech $m \in \mathbb{N}$, $p \in B^*$. Kód dvojice (m, p) je reťazec $\langle m, p\rangle := 1^{|\overline{m}|}\, 0\, \overline{m}\, p$, kde $\overline{m} \in B^*$ je číslo m v binárnom tvare.

Príklad 7.0.2. $\langle 6, 1001\rangle = 1^{|110|}\, 0\, 110\, 1001 = 111\, 0\, 110\, 1001$, naznačujúc jednoznačnú spätnú rekonštruovateľnosť dvojice.

Fakt 1. (a) $|\langle m, p\rangle| = |1^{|\overline{m}|}\, 0\, \overline{m}\, p| = 2|\overline{m}| + 1 + |p| \le 2\log_2 m + 3 + |p|$, pretože platí $|\overline{m}| \le 1 + \log_2 m$.
(b) $(m, p) \ne (l, q) \iff \langle m, p\rangle \ne \langle l, q\rangle$

Definícia [Kolmogorovská zložitosť]. Kolmogorovská zložitosť reťazca $x \in B^*$:

$$C(x) := \min\big\{|\langle m, p\rangle| : m \in \mathbb{N},\, p \in B^*,\, A_m(p) = x\big\}\,.$$

Príklad 7.0.3. $C(\pi_n) \le \log(n) + O(1)$, kde π_n je binárny rozvoj čísla π na n miest.

Dôvod. $\exists A_l \forall n : A_l(\overline{n}) = \pi_n \implies C(\pi_n) \le |\langle l, \overline{n}\rangle| \le 2\log l + 3 + |\overline{n}| \le \underbrace{2\log l + 4}_{O(1)} + \log n.$ □

Príklad 7.0.4. $C(y_n) \le \log\log\log\log |y_n| + O(1)$, kde $y_n = \overbrace{11\ldots1}^{2^{2^{2^n}}}$.

Dôvod. $\exists A_m \forall n : A_m(\overline{n}) = y_n$, potom:

$$\begin{aligned} C(y_n) &\le |\langle m, \overline{n}\rangle| \\ &\le 2\log m + 3 + \overline{n} \\ &\le 2\log m + 4 + \log n \\ &= \log n + O(1) \\ &= \log\log\log\log 2^{2^{2^n}} + O(1) \\ &= \log\log\log\log |y_n| + O(1) \end{aligned}$$

□

Fakt 2. Existuje $c \geq 0$ také, že $C(x) \leq |x| + c$ pre ľubovoľný reťazec $x \in B^*$.

Dôkaz. $\exists A_m : A_m(x) = x$, $\forall x \in B^* \Longrightarrow C(x) \leq |\langle m, x\rangle| \leq |x| + \underbrace{\log m + 3}_{c}$.

□

Definícia [c-nestlačiteľnosť]. Reťazec $x \in B^*$ je c-nestlačiteľný ($c \geq 0$), ak $C(x) \geq |x| - c$.

Lema 7.0.3. *Aspoň $2^n(1 - 2^{-c}) + 1$ reťazcov dĺžky n je c-nestlačiteľných.* [1]

Dôkaz. Nech M je množina všetkých c-nestlačiteľných reťazcov dĺžky n, a ďalej nech $M' = \{0, 1\}^n - M$.

Pre ľubovoľný reťazec $x \in M'$ (t.j. stlačiteľný aspoň o $c + 1$ bitov) existuje dvojica (m_x, p_x), taká, že $A_{m_x}(p_x) = x$, pričom $C(x) = |\langle m_x, p_x\rangle| < |x| - c = n - c$, teda existuje vzor kratší ako $n - c$.

Počet reťazcov tvaru $\langle m_x, p_x\rangle$ kde $x \in M'$ je najviac $2^{n-c} - 1$ (počet úplne všetkých reťazcov dĺžky menšej ako $n - c$).

Ak $x, y \in M'$, $x \neq y$, potom $\langle m_x, p_x\rangle \neq \langle m_y, p_y\rangle$, lebo $x \neq y \Rightarrow A_{m_x}(p_x) \neq A_{m_y}(p_y)$ $\Rightarrow (m_x, p_x) \neq (m_y, p_y) \Longrightarrow_{\text{Fakt 1 (b)}} \langle m_x, p_x\rangle \neq \langle m_y, p_y\rangle$.

Nakoniec môžeme povedať, že mohutnosť množiny M' je menšia ako počet reťazcov tvaru $\langle m_x, p_x\rangle$, teda ako $2^{n-c} - 1$, čiže $|M| = |\{0, 1\}^n| - |M'| \geq 2^n - (2^{n-c} - 1) = 2^n(1 - 2^{-c}) + 1$. □

Dôsledok 7.0.1. Pre každý kompresný program A a pre každú konštantu $c \geq 0$ existuje $c' \geq 0$ také, že $|A(x)| \geq |x| - c'$ pre každý c-nestlačiteľný reťazec $x \in B^*$. [2]

Dôvod. Nech A_j je „dekompresný” program ku A, t.j. $A_j(A(x)) = x$ pre ľubovoľný reťazec $x \in B^*$. Preto pre každé c-nestlačiteľné $x \in B^*$ platí:

$$|x| - c \leq C(x) \leq |\langle j, A(x)\rangle| \leq 2 \log j + 3 + |A(x)|\,,$$

t.j. $c' = c + 2 \log j + 3$. □

Veta 7.0.4. *Funkcia $C(x)$ je totálna, ale nie je rekurzívna.*

Dôkaz. Totálnosť vyplýva z dôkazu Faktu 2. Pre všetky n nech x_n je prvý (v lexikografickom usporiadaní) binárny reťazec, taký, že $n \leq C(x_n)$. Nech by $C(x)$ bola rekurzívna. To by znamenalo, že existuje algoritmus A_k, taký, že $A_k(\overline{n}) = x_n$ pre všetky n, kde $\overline{n}$ je číslo n v binárnom tvare. Teda:

$$C(x_n) \leq |\langle k, \overline{n}\rangle| \leq 2 \log k + 3 + |\overline{n}| \leq 2 \log k + 4 + \log n$$

lebo $|\overline{n}| \leq 1 + \log n$, čo je v spore s $n \leq C(x_n)$ pre dosť veľké n. □

Veta 7.0.5. *Existuje totálna rekurzívna funkcia $g(t, x)$ (monotónne nerastúca v t), taká, že:*

$$\lim_{t\to\infty} g(t, x) = C(x)\,.$$

Dôkaz. Nech $A_m(x) = x$ pre všetky $x \in B^*$, t.j. $C(x) \leq |\langle m, x\rangle|$ pre všetky $x \in B^*$. Potom:

$$g(t, x) = \min\Big(\big\{|\langle j, p\rangle| : A_j(p) = x \text{ počas } \leq t \text{ krokov; } |\langle j, p\rangle| < |\langle m, x\rangle|\big\} \cup \big\{|\langle m, x\rangle|\big\}\Big)\,.$$

□

[1] Ako dôsledok, aspoň jeden reťazec dĺžky n je nestlačiteľný.

[2] Teda c-nestlačiteľné reťazce sa nedajú veľmi stlačiť žiadnym kompresným programom.

Veta 7.0.6. *Ak A'_1, A'_2, A'_3, ... je iné efektívne očíslovanie všetkých programov a C' je príslušná Kolmogorovská zložitosť, potom existuje konštanta c, taká, že:*

$$|C(x) - C'(x)| \leq c\,, \quad \forall x \in B^*$$

Dôkaz. Nech A je program, ktorý zo vstupu $\langle l, q\rangle$ urobí výstup $A_l(q)$ pre všetky $l \in \mathbb{N}$, $q \in B^*$.[3] Keďže $A = A'_k$ pre nejaké k, platí: $A'_k(\langle l, q\rangle) = A_l(q)$. Nech $C(x) = |\langle m, p\rangle|$, kde $A_m(p) = x$. Nakoľko $A'_k(\langle m, p\rangle) = A_m(p) = x$, potom:

$$C'(x) \leq |\langle k, \langle m, p\rangle\rangle| = |1^{|\overline{k}|}\, 0\, \overline{k}\, \langle m, p\rangle| = 2|\overline{k}| + 1 + |\langle m, p\rangle| = 2|\overline{k}| + 1 + C(x).$$

a podobne $C(x) \leq 2|\overline{k'}| + 1 + C'(x)$ pre nejaké k'. □

7.1 Aplikácie nestlačiteľných reťazcov

Tvrdenie 7.1.1. *Pre nekonečne veľa n platí: Počet prvočísel v intervale $\langle 2, n\rangle$ je aspoň*

$$\frac{\log_2 n}{3 \log_2 \log_2 n} - O(1).$$

Dôkaz. Pre ľubovoľné n, nech x_n je n-tý binárny reťazec v lexikografickom usporiadaní. Číslo n – a teda aj reťazec x_n – možno algoritmicky zostrojiť z m-tice $(e_1, e_2, \ldots, e_m)$, pre ktorú:

$$n = p_1^{e_1} p_2^{e_2} \ldots p_m^{e_m}, \text{ kde } p_i \text{ je } i\text{-te prvočíslo, } e_i \in \mathbb{N}_0.$$

Platí: $0 \leq e_i \leq \log_2 n \ \forall i \leq m$, lebo ak by $e_j > \log_2 n$ pre nejaké $j \leq m$, potom by:

$$n = p_1^{e_1} p_2^{e_2} \ldots p_m^{e_m} \geq p_j^{e_j} \geq 2^{e_j} > 2^{\log_2 n} = n \quad \text{— spor!}$$

m-ticu $(e_1, e_2, \ldots, e_m)$ možno zakódovať reťazcom:

$$z_n = 1^{|\overline{e}_1|} 0\, \overline{e}_1 \ldots 1^{|\overline{e}_m|} 0\, \overline{e}_m, \text{ kde } \overline{e}_i \text{ je číslo } e_i \text{ v binárnom tvare.}$$

$\Rightarrow \exists$ alg. A_k taký, že $A_k(z_n) = x_n \ \forall n$.

$\Rightarrow C(x_n) \leq |\langle k, z_n\rangle| \leq 2\log_2 k + 3 + m(2\log_2 \log_2 n + 3)$, lebo $|\overline{e}_i| \leq 1 + \log_2 e_i \leq 1 + \log_2 \log_2 n$, keďže $e_i \leq \log_2 n$ (pozri vyššie).

$\Rightarrow C(x_n) \leq 3m \log_2 \log_2 n + O(1)$.

Dá sa dokázať: $|x_n| \geq \log_2 n - 1$. Z toho vyplýva, že $\log_2 n - 1 \leq |x_n| \leq C(x_n) \leq 3m \log_2 \log_2 n + O(1)$ pre 0-stlačiteľné x_n, a teda:

$$m \geq \frac{\log_2 n}{3 \log_2 \log_2 n} - O(1) \text{ pre 0-nestlačiteľné } x_n.$$

□

Tvrdenie 7.1.2. *Pre skoro všetky nepárne n platí, že ľubovoľný zásobníkový automat rozpoznávajúci $L = \{x2x^R | x \in B^*\}$ musí použiť na niektorom vstupe dĺžky n zásobník dĺžky $\Omega(n)$.*

Dôkaz. Nech A je zásobníkový automat pre L a nech A má množinu stavov Q a abecedu zásobníka $\Gamma = \{0, 1\}$. Pre každý reťazec $x \in B^*$ nech (q_x, w_x), kde $q_x \in Q$, $w_x \in \Gamma^*$, je konfigurácia, do ktorej sa dostane A na vstupe $x2x^R$, keď číta symbol 2.

[3] Program A je univerzálny stroj: na vstupe $\langle l, q\rangle$ simuluje A_l na vstupe q.

Platí, že ak $x \neq y$, potom $(q_x,\, w_x) \neq (q_y,\, w_y)$, lebo inak by A musel akceptovať aj $y2x^R \notin L$, keďže akceptuje $x2x^R$. Z toho vidieť, že každé $x \in B^*$ možno jednoznačne určiť a algoritmicky zostrojiť z trojice ("popis automatu A", q_x, w_x), ktorú možno zakódovať binárnym reťazcom

$$z_x = 1^{|\mathrm{bin}(A)|}\, 0\, \mathrm{bin}(A)\, 1^{|\mathrm{bin}(q_x)|}\, 0\, \mathrm{bin}(q_x) w_x$$

kde $\mathrm{bin}(A)$ resp. $\mathrm{bin}(q_x)$ je binárny kód automatu A resp. stavu q_x ($|\mathrm{bin}(q_x)| = \lceil \log |Q| \rceil$).

Existuje teda algoritmus A_k, taký, že $A_k(z_x) = x$ pre všetky x, t.j. $C(x) \leq |\langle k,\, z_x \rangle| = |w_x| + O(1)$. Potom možno nahliadnuť, že $|x| \leq C(x) \leq |w_x| + O(1)$ pre každý 0-nestlačiteľný reťazec x, t.j. $|w_x| = \Omega(|x2x^R|)$ pre skoro každý 0-nestlačiteľný reťazec x. $\square$

Kapitola 8

Informačná vzdialenosť

Nech $A_1^{(2)}$, $A_2^{(2)}$, $A_3^{(2)}$, ... je efektívne očíslovanie všetkých programov $A_i^{(2)} : B^* \times B^* \to B^*$ pre všetky i.

Definícia [Informačná vzdialenosť]. Informačná vzdialenosť (od reťazca $y \in B^*$ ku $x \in B^*$) je definovaná:

$$C(x|y) = \min\big\{|\langle m, p\rangle| : A_m^{(2)}(p, y) = x,\ m \in \mathbb{N},\ p \in B^*\big\}.$$

Poznámka 8.0.1. Informačná vzdialenosť je minimálne množstvo informácie potrebnej na konštrukciu x, ak poznáme y.

Príklad 8.0.1. $\exists c \in \mathbb{N}\ \forall x \in B^* : C(x|x^R) \leq c$, lebo existuje program $A_m^{(2)}$ taký, že $A_m(\varepsilon, x^R) = x$, z čoho vyplýva $C(x|x^R) \leq |\langle m, \varepsilon\rangle| = c$.

Lema 8.0.3. *(a)* $\exists c, c' : C(x|y) \leq C(x) + c \leq |x| + c' \quad \forall x, y \in B^*$, [1]
(b) $\exists c, c' : \forall n\ \exists x, y \in \{0,1\}^n : \ C(x|y) \leq c \ \wedge\ C(y|x) \geq n - 2\log_2 n - c'$. [2]

Dôkaz. (a) Nech $A_l(q) = x$ a $C(x) = |\langle l, q\rangle|$. Zrejme existuje program $A_k^{(2)}$ taký, že

$$A_k^{(2)}(\langle j, v\rangle, y) = A_j(v) \quad (\forall j \in \mathbb{N},\ v, y \in B^*)$$

a teda $A_k^{(2)}(\langle l, q\rangle, y) = A_l(q) = x$. Z toho vyplýva:

$$C(x|y) \leq |\langle k, \langle l, q\rangle\rangle| = |1^{|\bar{k}|}0\bar{k}\langle l, q\rangle| = 2|\bar{k}| + 1 + |\langle l, q\rangle| = C(x) + \underbrace{2|\bar{k}| + 1}_{c}.$$

(b) Pre dané n, nech $x = 0^n$ a y nech je 0-nestlačiteľný reťazec dĺžky n. Ľahko nahliadneme, že existuje program $A_m^{(2)}$ taký, že $A_m^{(2)}(\varepsilon, v) = 0^{|v|}$ pre ľubovoľné $v \in B^*$, a teda:

$$A_m^{(2)}(\varepsilon, y) = 0^{|y|} = 0^n \implies C(0^n|y) \leq |\langle m, \varepsilon\rangle| = c.$$

Opačným smerom, nech $A_l^{(2)}(p, 0^n) = y$ pričom $C(y|0^n) = |\langle l, p\rangle|$, t.j. ak poznáme l, p, n, vieme zostrojiť y; čo možno vyjadriť aj tak, že existuje A_k taký, že $A_k(1^{|\bar{n}|}0\bar{n}1^{|\bar{l}|}0\bar{l}p) = y$. Potom:

$$C(y) \leq |\langle k,\ 1^{|\bar{n}|}0\bar{n}1^{|\bar{l}|}0\bar{l}p\rangle| = 2|\bar{k}| + 1 + 2|\bar{n}| + 1 + \overbrace{|\langle l, p\rangle|}^{C(y|0^n)} \leq C(y|0^n) + 2\log n + 2|\bar{k}| + 4.$$

z čoho vyplýva, že: $n \leq |y| \leq C(y) \leq C(y|0^n) + 2\log n + \underbrace{2|\bar{k}| + 4}_{c'}$ pre každé 0-nestlačiteľné y. □

[1] Jednoduchý súvis informačnej vzdialenosti a Kolmogorovskej zložitosti. Poslednú časť nerovnosti sme dokázali už v predchádzajúcej kapitole o Kolmogorovkej zložitosti (7) (**Fakt 2**).

[2] Teda stačí málo informácie na konštrukciu x z y, ale opačným smerom je jej potrebné podstatne väčšie množstvo.

Fakt. Pre dané y existuje najviac $2^{k+1}-1$ reťazcov x s vlastnosťou $C(x|y) \leq k$.

Dôkaz. Pre každé takéto x musí existovať reťazec $\langle m_x, p_x \rangle$ dĺžky najviac k taký, že $A_{m_x}(p_x) = x$ a $C(x|y) = |\langle m_x, p_x \rangle|$. Ale takýchto $\langle m_x, p_x \rangle$ je najviac $2^{k+1}-1$. □

Lema 8.0.4. *Pre ľubovoľné $n \in \mathbb{N}$ existujú reťazce $x, y \in \{0, 1\}^n$, $x \neq y$ také, že $C(x|y) \geq n-1$ aj $C(y|x) \geq n-1$.*

Dôkaz. Nech $x_1, x_2, \ldots, x_{2^n}$ sú všetky binárne reťazce dĺžky n a nech $\mathbf{A} = (a_{ij})$ je booleovská matica rádu $2^n \times 2^n$ taká, že:

$$a_{ij} = \begin{cases} 0 & \text{, ak } C(x_i|x_j) \leq n-2 \\ 1 & \text{, ak } C(x_i|x_j) \geq n-1 \end{cases}$$

Z vyššie uvedeného **Fakt**-u vyplýva, že v každom stĺpci matice $\mathbf{A}$ je nanajvýš $2^{n-1}-1$ núl, t.j. aspoň $2^{n-1}+1$ jednotiek. V celej matici sa teda nachádza aspoň $2^n(2^{n-1}+1)$ jednotiek. Na diagonále matice je triviálne nanajvýš 2^n jednotiek.

V celej matici $\mathbf{A}$ (okrem diagonály) je teda aspoň $2^n(2^{n-1}+1) - 2^n = 2^{2n-1}$ jednotiek. Ak by pre každú dvojicu indexov i, j, $1 \leq i < j \leq 2^n$, ktorých je $\binom{2^n}{2}$, platilo $a_{ij} = 0$ alebo $a_{ji} = 0$, potom by v celej $\mathbf{A}$ (okrem diagonály) bolo nanajvýš $\binom{2^n}{2} = 2^{2n-1} - 2^{n-1}$ jednotiek, čo je menej ako zdôvodnená spodná hranica počtu jednotiek 2^{2n-1}. Tým prichádzame ku sporu, z čoho vyplýva, že musí existovať dvojica i, j s vlastnosťou $a_{ij} = 1 = a_{ji}$. □

Na záver si položme otázku, ako minimálne zložitý musí byť program realizujúci (neznámu) funkčnú závislosť výstupov y_i na vstupoch x_i, $i = 1, 2, \ldots, t$.

Lema 8.0.5. *$\exists c \geq 0$ také, že ak pre nejaké reťazce $x_1, x_2, \ldots, x_t, y_1, y_2, \ldots, y_t \in \{0, 1\}^+$ a program A_m platí $A_m(x_i) = y_i$ pre $\forall i$, potom:*

$$C(\textit{bin. kód } A_m) \geq C(y|x) - c\,,$$

kde $x = 1^{|x_1|}\,0\,x_1 \ldots 1^{|x_t|}\,0\,x_t$ je kód postupnosti $x_1, x_2, \ldots, x_t$, podobne $y = 1^{|y_1|}\,0\,y_1 \ldots 1^{|y_t|}\,0\,y_t$ je kód postupnosti $y_1, y_2, \ldots, y_t$.

Dôkaz. Nech $C(\text{bin. kód } A_m) = |\langle l, p \rangle|$, kde $A_l(p) = \text{bin. kód } A_m$. Potom existuje program $A_r^{(2)}$, taký, že . Program $A_r^{(2)}$ pracuje nasledovne:

a) z reťazca $\langle l, p \rangle$ zistí $\bar{l}$ a p

b) z $\bar{l}$ zostrojí program A_l

c) zostrojí (bin. kód A_m) $= A_l(p)$ (t.j. simuluje A_l na vstupe p)

d) z x zostrojí $x_1, x_2, \ldots, x_t$

e) zostrojí $y_1 = A_m(x_1)$, $ldots$, $y_t = A_m(x_t)$ (t.j. simuluje A_m na x_i pre všetky i)

f) zostrojí $y = 1^{|y_1|}\,0\,y_1 \ldots 1^{|y_t|}\,0\,y_t$

Keďže $A_r^{(2)}(\langle l, p \rangle, x) = y$, potom

$$\begin{aligned} C(y|x) &\leq |\langle r, \langle l, p \rangle \rangle| \\ &= |1^{|\bar{r}|}\,0\,\bar{r}\,\langle l, p \rangle\rangle| \\ &= 2|\bar{r}| + 1 + |\langle l, p \rangle| \\ &= 2|\bar{r}| + 1 + C(\text{bin. kód } A_m) \\ &= c + C(\text{bin. kód } A_m) \end{aligned}$$

□

Poznámka 8.0.2. Program A_m je komplikovaný a dlhý, ak je závislosť y_i na x_i komplikovaná, t.j. ak je hodnota $C(y|x)$ veľká.

Literatúra

Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., Protasi, M.: *Complexity and Approximation.* Springer, 1999.

Brassard, G., Bratley, P.: *Algorithmics: Theory and Practice.* Prentice-Hall, 1988.

Cormen, T., Leiserson, C., Rivest, R.: *Introduction to Algorithms.* MIT Press, 1994.

Garey, M. R., Johnson, D. S.: *Computers and Intractability - A Guide to the Theory of NP-completeness.* W. H. Freeman, 1979.

Hopcroft, J. E., Ullman, J. D.: *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, 1979.

Hromkovič, J.: *Algorithmics for Hard Problems.* Springer, 2nd edition, 2003.

Li, M., Vitániy, P.: *Introduction to Kolmogorov Complexity and its Applications.* Springer, 1993.

Papadimitriou, C. H.: *Computational Complexity.* Addison-Wesley, 1994.

Sipser, M.: *Introduction to the Theory of Computation.* Thompson Course Technology, 2nd edition, 2006.