Složitost

Filip Hlásek

Příspěvek popisuje dva základní koncepty teoretické informatiky, Turingovy stroje a složitost. Kromě definic důležitých pojmů uvádí také několik souvisejících tvrzení, vět a cvičení.

Uvažujme libovolný problém s jasně daným vstupem a očekávanou odpovědí. Budeme zkoumat, kolik operací počítačového programu je potřeba ke správnému výpočtu. Některé problémy lze řešit poměrně efektivně, jiné oproti tomu nelze řešit vůbec. Ke zcela přesnému popisu výpočtu zavedeme abstraktní stroj, pomocí něhož ukážeme některé vlastnosti reálných počítačů.

Problémem v našem smyslu budeme rozumět přesný popis toho, jak může vypadat *vstup* a jaký je očekávaný *výstup*. Vstupem bude typicky posloupnost číslic, případně písmen či jiných dalších znaků. Obecně budeme seznam všech použitých symbolů značit pojmem *abeceda*, což bude vždy konečná množina. Očekávaný výstup je taktéž posloupnost znaků abecedy.

Příkladem takového problému může být například sečtení dvou čísel zadaných v desítkové soustavě oddělených mezerou. Abeceda v tomto případě obsahuje deset cifer a jeden speciální znak mezeru. Dalším triviálním problémem je pro libovolný vstup odpovědět „Drbu vrbu.“ Příkladem náročnějšího problému může být hledání prvočíselného rozkladu přirozeného čísla či hledání nejdelší kružnice v grafu.

Program řešící daný problém je typicky posloupnost počítačových instrukcí, které pro každý povolený vstup vyprodukují požadovaný výstup. Zápis programu v libovolném rozumném programovacím jazyku, jako jsou C, Java či Pascal, popisuje jednotlivé kroky, které budou během výpočtu vykonány.

Turingovy stroje

Programy zapsané v běžných programovacích jazycích je obvykle velmi náročné analyzovat. Je složité porozumět tomu, co se děje v procesoru počítače či v jeho paměti a z teoretického pohledu téměř nemožné přesně odhadnout, jak dlouho výpočet potrvá, případně kolik přesně operací počítač vykoná. Proto se matematici rozhodli navrhnout abstraktní počítač, který bude dostatečně jednoduchý ke zkoumání. Je sice mnohem náročnější ho programovat a je méně výkonný, ale ukazuje se, že tento rozdíl není natolik podstatný a model nám poskytuje důležité nástroje ke zkoumání

reálných počítačů.

Definice.

(1) *Abeceda* je libovolná konečná množina. Její prvky budeme nazývat *symboly* či *znaky*. Například $\{0,1\}$ je binární abeceda, $\{a,b,\ldots,z\}$ je abeceda malých anglických písmen a $\{7,X,\heartsuit,\natural\}$ jiná možná abeceda.
(2) *Slovem* nad abecedou Σ nazveme konečnou posloupnost jejích symbolů. Slovo u délky n (značíme $|u|=n$) zapíšeme jako $u=a_1a_2\ldots a_n$.
(3) *Jazyk* nad abecedou Σ je nějaká množina slov nad abecedou Σ. Jazyk může být jak konečný, např. $\{00,01,10,11\}$, tak nekonečný, např. $\{0,01,010,0101,\ldots\}$.

Dále zavedeme teoretický model počítače poprvé popsaný britským matematikem Alanem Turingem. Tento stroj sestává z potenciálně nekonečné pracovní pásky, což je posloupnost políček, z nichž na každém může být zapsaný právě jeden symbol dané abecedy. Výpočetní jednotka stroje je tvořena takzvanou *hlavou*, která je vždy nastavena na jedno z políček pásky a může z něj přečíst aktuální symbol či na něj symbol zapsat. Dále je výpočetní jednotka vždy v jednom ze *stavů* z předem známé množiny možných stavů. Na začátku výpočtu je stroj v nějakém stavu q_0, na pásce je zapsaný vstup a hlava ukazuje na první symbol vstupu.

Samotný výpočet se poté skládá z kroků, kde v každém kroku se pouze na základě aktuálního stavu jednotky a čteného symbolu na pásce stroj rozhodne, do kterého stavu se přesune, zda se pohne hlava a co se zapíše na pásku. Formálně se jedná o funkci dvou parametrů (aktuálního stavu a čteného symbolu), jejímž výstupem je trojice (q,a,m). Zde q vyjadřuje nový stav stroje, symbol a je zapsán na pásku na aktuální pozici hlavy a m popisuje, jak se pohne hlava po vykonání operace. Popsané funkci se říká *přechodová funkce*. Dostane-li se stroj do jednoho z *koncových stavů*, výpočet končí a vstup je přijat. Pokud v aktuálním stavu není definována přechodová funkce, případně pokud se stroj nikdy nezastaví, je vstup odmítnut.

Popsaný stroj nemá žádný výstup a pouze *rozhoduje*, zda zadaný vstup patří do zkoumaného jazyka, či nikoliv. Řešení takovýchto problémů je přesto dostatečně zajímavé a ukážeme si, že po vyřešení rozhodovací verze problému je již vypsání požadovaného výstupu většinou poměrně jednoduché.

Definice. *Deterministický Turingův stroj (DTS)* je pětice (Q,Σ,δ,q_0,F), kde

(1) Q je konečná množina stavů.
(2) Σ je abeceda.
(3) $\delta\colon (Q\setminus F)\times\Sigma\to Q\times\Sigma\times\{-1,0,1\}$ je přechodová funkce (program) Turingova stroje. Zjednodušeně pro každý stav a symbol z abecedy říká, do jakého stavu se stroj přesune, co se zapíše na pásku a kterým směrem se pohne hlava.
(4) $q_0\in Q$ je počáteční stav stroje.
(5) $F\subset Q$ je množina koncových strojů.

Definice. *Konfigurací Turingova stroje* $T=(Q,\Sigma,\delta,q_0,F)$ nazveme konečnou část pásky, na kterou byl zapsán nějaký symbol, společně se stavem stroje a pozicí

jeho hlavy. Konfigurace určuje všechny potřebné informace pro pokračování výpočtu stroje.

Cvičení. Navrhněte DTS, který přijme zadaný vstup právě tehdy, když se skládá ze všech stejných symbolů.

Definice. *Deterministickým Turingovým strojem s výstupem* nazveme deterministický Turingův stroj, který má navíc jednu výstupní pásku s vlastní hlavou. Na výstupní pásku může pouze zapisovat a hlava se při každém zápisu pohne doprava. Výstup je tedy zapisován po znacích.

Cvičení. Uvažme graf a hledejme v něm nejdelší cestu, tj. neopakující se posloupnost vrcholů, kde každé dva po sobě jdoucí jsou spojené hranou. Uvažme DTS T, který pro zadanou délku k rozhodne, zda v grafu existuje cesta délky k. Navrhněte DTS s výstupem, který může využívat T a nalezne nejdelší cestu v zadaném grafu.

Definice. Od dříve popsaného stroje se *n-páskový deterministický Turingův stroj* liší tím, že má n pásek (konečně mnoho). Každá z pásek má vlastní hlavu a přechodová funkce se rozhoduje podle symbolů pod všemi těmito hlavami, zapisuje v každém kroku na všechny pásky a posouvá všemi hlavami. Navíc v tomto případě obvykle vymezujeme vstupní pásku, která obsahuje zadaný vstup a není na ni možné zapisovat, slouží pouze ke čtení vstupu. Ostatní pásky poté nazýváme pracovní pásky.

Věta. *Každý problém řešitelný n-páskovým DTS je řešitelný také jednopáskovým DTS.*

Definice. *Nedeterministický Turingův stroj (NTS)* je stroj, který se od Turingova stroje liší přechodovou funkcí. Ta v tomto případě neudává přesný stav, symbol a posun hlavy, ale nabízí místo toho několik možností těchto trojic. V případě, že funkce nabízí pro každou konfiguraci stroje pouze jednu možnost, jedná se o deterministický stroj. Nedeterminismus se projeví, pokud je možností více, stroj si poté může vybrat libovolnou z nich (například náhodně). Vstup bude nedeterministickým strojem přijat, pokud existuje popsaná posloupnost rozhodnutí, při kterém by byl vstup přijat (jinými slovy – stroj si může vybírat možnosti takovým způsobem, aby nakonec vstup přijal).

Cvičení. Ukažte, že každý problém řešitelný pomocí NTS je také řešitelný pomocí DTS. Jinými slovy, že umíme nedeterminismus simulovat pomocí deterministického stroje.

Věta. (Gödelovo číslo) *Každý Turingův stroj T je možné zakódovat konečným binárním slovem. Takovému slovu budeme říkat Gödelovo číslo stroje T či kód stroje T a značit jej $Kód(T)$.*

Definice. *Univerzální Turingův stroj U* je DTS takový, který umí simulovat libovolný DTS T. To znamená, že pro libovolný vstup v stroje T, pokud předložíme stroji U jako vstup $Kód(T)$ a v, dopadne výpočet stejně, jako by dopadl výpočet

stroje T nad vstupem v.

Věta. *Existuje univerzální Turingův stroj.*

Definice. (Halting problém) Uvažme libovolný DTS T a jeho vstup v. Hledáme takový stroj, jehož vstupem bude $Kód(T)$ a v, vždy se zastaví a přijme vstup právě tehdy, když se T zastaví nad vstupem v.

Věta. *Neexistuje DTS řešící Halting problem.*

Složitost problémů

Definice. Nechť T je DTS a označme $L(T)$ jazyk přijímaný strojem T. Řekneme, že T má *polynomiální časovou složitost*, pokud existuje polynom p takový, že pro každé $w \in L$ vykoná T nad vstupem w nejvýše $p(|w|)$ kroků. Stroj smí tedy vykonat nejvýše polynomiálně mnoho kroků vzhledem k velikosti vstupu.

Definice. Označme P množinu všech problémů řešitelných DTS s polynomiální časovou složitostí.

Definice. Řekneme, že NTS T má *polynomiální časovou složitost*, pokud pro každý přijímaný vstup nejkratší z přijímacích výpočtů vykoná nejvýše polynomiálně mnoho kroků vzhledem k délce vstupu.

Definice. Označme NP množinu všech problémů řešitelných NTS s polynomiální časovou složitostí.

Tvrzení. $P \neq NP$

Definice. Řekneme, že problém A je *polynomiálně převoditelný* na problém B, pokud existuje DTS s výstupem s polynomiální časovou složitostí takový, že každý vstup problému A převede na vstup problému B (jako vstup má vstup problému A a na výstup dá vstup problému B) tak, aby problém B měl stejný výstup jako problém A. Značíme $A \propto B$.

Definice. (Hádací páska, orákulum) Turingův stroj můžeme také doplnit o tzv. *orákulum*, což je páska, ze které je možné pouze číst a na které se vyskytuje náhodná posloupnost nul a jedniček.

Definice. Turingův stroj s orákulem T může občas odpovědět chybně. Konkrétně jsou možné dva druhy chyb:

(1) A: stroj přijme vstup, který neměl přijmout,
(2) B: stroj nepřijme vstup, který měl přijmout.

Pravděpodobnosti jednotlivých chyb budeme značit $Pr(A)$, resp. $Pr(B)$. Řekneme, že T s polynomiální časovou složitostí náleží do třídy

(1) P, pokud $Pr(A) = 0$ a $Pr(B) = 0$,
(2) NP, pokud $Pr(A) = 0$ a $Pr(B) < 1$,
(3) RP, pokud $Pr(A) = 0$ a $Pr(B) \leq \frac{1}{2}$,
(4) BPP, pokud $Pr(A) \leq \frac{1}{3}$ a $Pr(B) \leq \frac{1}{3}$.