České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra telekomunikační techniky

A7B32KBE - Teorie výpočetní složitosti

Ing. Tomáš Vaněk, Ph.D.

Teorie složitosti

- zabývá se výpočetními úlohami
- zkoumá jejich **řešitelnost** a **náročnost** na zdroje
- rozlišujeme:
  - teorii vypočitatelnosti (má úloha řešení?)
  - teorii výpočetní složitosti (pokud má úloha řešení, jaké má nároky na výpočet?)
- úlohy se ve výpočetní teorii označují jako problémy (problém obchodního cestujícího, zavazadlový problém,…)
- problém je dán množinou zadání, kterých může existovat nekonečně mnoho
- jedno konkrétní zadání označujeme jako instanci

Teorie složitosti – co můžeme zkoumat

- **výpočet funkce** – má vždy jednoznačný výsledek
- **rozhodovací problémy** – speciální případ výpočtu funkce, kdy nám stačí odpověď ano/ne
- **optimalizační problémy** – hledáme v množině přípustných řešení, takové které splní nějaké kriteriální funkce
- **vyhledávací problémy** – hledáme libovolný objekt, který splňuje nějakou vlastnost

Teorie složitosti - příklad

- Mějme funkci $f(x) = 3x^2 - 7y$
- Otázka pro výpočet funkce
  - Jaká je hodnota funkce pro $x = 2$ a $y = 1$?
  - Odpověď: 5
- Otázka pro rozhodovací problém
  - Může funkce nabývat záporných hodnot?

  Odpověď: Ano
- Otázka pro optimalizační problém
  - pro které hodnoty x a y je hodnota funkce menší než 5

Teorie složitosti

- výpočty vyžadují nějaké zdroje – nejčastěji čas a paměť
- rozlišujeme časovou složitost a prostorovou (paměťovou) složitost v závislosti na velikosti vstupu instance
- časovou složitost – počet kroků zvoleného modelu počítače nutný k vyřešení problému
- prostorová složitost – počet prvků instance nebo počet bitů nutných k zakódování instance
- k posouzení složitosti problémů lze přistoupit dvěma způsoby:

  - nejhorší případ

  - průměrný případ

Teorie složitosti

O – shora neostře omezená složitost

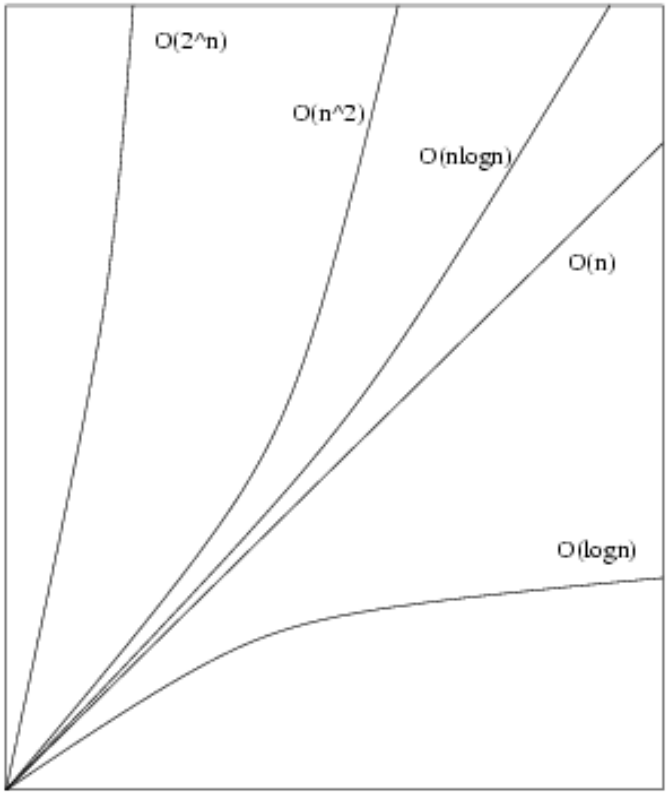
Ω – zdola neostře omezená složitost

Θ – z obou stran omezená složitost

o – shora ostře omezená složitost

ω – zdola ostře omezená složitost

nejčastěji se setkáme s variantou - O



Teorie složitosti

- přesné vyjádření může být velmi náročné
- soustředíme se jen na rozhodující člen, který nám nejvíce ovlivňuje funkci vyjadřující složitost
- ostatní členy se zanedbají

Příklad vyjádření složitosti

- mějme funkci $T_M(n)$, která vyjadřuje časovou složitost problému
- například $T_M(n) = a.n^3 + b.n^2 + c.n$
- pro libovolně zvolené konstanty *a*, *b*, *c* (v 'rozumných mezích') bude funkce $T_M(n)$ závislá na nejrychleji rostoucím členu vyjádřeném velikostí vstupu *n*
- v tomto případě platí, že $T_M(n) \in O(n^3)$

Teorie složitosti

- v současné době jsou zvládnutelné problémy s polynomiální složitostí
- problémy vyjádřené exponenciální nebo faktoriálovou složitostí jsou už pro malé velikosti vstupu nezvládnutelné
- i polynomiální složitost může být nezvládnutelná, ale se složitostmi *100 000n* nebo n^{1000} se setkáme zřídka ne-li vůbec
- pro většinu praktických problémů řešitelných v polynomiálně omezeném čase známe algoritmy s relativně nízkým stupněm polynomu
- zjednodušení - zanedbáme rozdíly mezi různými stupni polynomiální složitosti

Teorie složitosti

g(n)	n = 10	n = 20	n = 30	n = 40	n = 50
n	10 ms	20 ms	30 ms	40 ms	50 ms
n.log n	10 ms	26 ms	44 ms	64 ms	85 ms
n^2	100 ms	400 ms	900 ms	1,6s	2,5 s
2^n	1 s	17 min	12 d	25 r	$4 \cdot 10^4$ r
n!	1 h	$8 \cdot 10^7$ r	$8 \cdot 10^{21}$ r	$3 \cdot 10^{37}$ r	$1 \cdot 10^{54}$ r
n^n	115 d	$3 \cdot 10^{15}$ r	$7 \cdot 10^{33}$ r	$4 \cdot 10^{53}$ r	$3 \cdot 10^{74}$ r

Příklad pro složitost algoritmu $O(g(n))$, kde n je velikost vstupu a jeden výpočet trvá 1ms

Příklady na rozhodující člen

- najděte rozhodující člen
- $A(n) = n + 5 \,.\, \log n$

  $A(n) \in O(n)$
- $B(n) = n^4 + 2^n$

  $B(n) \in O(2^n)$
- $C(n) = 10n^n + 2n! + 3^n$

  $C(n) \in O(n^n)$

Teorie složitosti - výpočetní modely

- jsou definovány vstupem, výstupem a množinou dovolených operací

Realizují se pomocí:

- Turingova stroje
- RAM (Random Access Machine)
- Booleovských obvodů

časová složitost – délka výpočtu (počtem kroků)

paměťová složitost - velikost paměti pro vstup délky *n* v **nejhorším** případě

Churchova -Turingova teze

Každý algoritmus může být vykonán Turingovým strojem.

- libovolný program napsaný v konvenčním programovacím jazyce může být transformován na Turingův stroj a naopak
- Churchova – Turingova teze není větou a tedy nemůže být dokázána
- Churchova – Turingova teze může být vyvrácena, pokud bude objevena metoda akceptovatelná jako algoritmus, který nebude možno vykonat Turingovým strojem

Turingův stroj

- stroj, který pomocí čtecí hlavy zpracovává potenciálně nekonečnou pásku rozdělenou na buňky
- sekvenční přístup k paměti (pásce)
- během jednoho kroku se lze přesunout pouze na sousední buňku
- na začátku výpočtu je na pásce uložen vstup
- po ukončení výpočtu je za výstup považován řetězec neprázdných buněk na pásce
- velikost vstupu – počet obsazených buněk před začátkem výpočtu
- časová složitost ~ délce výpočtu (počet kroků, které stroj vykoná než se zastaví)
- paměťová složitost ~ velikosti paměti (počet buněk, které jsou navštíveny během výpočtu) pro vstup délky *n* v nejhorším případě

RAM

- skládá se z :
  - programové jednotky
  - operační paměti
  - vstupní jednotky
  - výstupní jednotky
- operační paměť se skládá z potenciálně nekonečného množství adresovaných buněk
- do každé buňky lze uložit libovolně velké číslo
- ke každé pamětové buňce lze libovolně přistoupit (rozdíl proti T. stroji)
- na počátku je ve všech buňkách v operační paměti uložena nula

RAM

- vstupní jednotka se skládá ze:
  - vstupní pásky
  - čtecí hlavy
- čtecí hlava po přečtení políčka pásky se přesune na další políčko a **nevrací se**
- výstupní jednotka se skládá z:
  - výstupní pásky
  - zapisovací hlavy,
- zapisovací hlava se po zapsaní políčka na pásku přesune na další políčko a opět se **nevrací**

Booleovské obvody

- výpočet je prováděn kombinováním Booleovských hodnot (0, 1) za použití Booleovských spojek (např. ∧, ∨, ¬)
- jsou konstruovány na určitou délku vstupu, případně je možné doplnit kratší vstup bezvýznamnými bity

Turingův stroj vs. RAM

- Turingův stroj lze simulovat pomocí RAM v lineárním čase
- RAM lze simulovat pomocí Turingova stoje v polynomiálním čase
- proto jsou problémy řešené v polynomiálním čase pro oba stroje shodné

Algoritmus x Problém

- algoritmus je konkrétní realizace řešení daného problému
- dobrý programátor by neměl překročit složitostí algoritmu složitost problému, tzn. že by problém s polynomiální složitostí neměl být řešen algoritmem s exponenciální složitostí

Rozhodnutelnost problémů

Definice: *Problém je určen trojicí(IN, OUT, p), kde IN je množina (přípustných) vstupů, OUT je množina výstupů a p je zobrazení (funkce) p: IN → OUT.*

Možné stavy:

1) Problém je **rozhodnutelný** - algoritmus pro daný problém skončí a vydá výsledek.
2) Problém je **částečně rozhodnutelný** - algoritmus pro jednu z odpovědí ANO/NE skončí a pro druhou je jeho běh nekonečný.
3) Problém je **nerozhodnutelný**.

Příklad nerozhodnutelného problému

Problém zastavení (halting problem)

„Znáte-li zdrojový kód programu a jeho vstup, rozhodněte, zda program zastaví, nebo zda poběží navždy bez zastavení„

1936 – Alan Turing dokázal, že tento problém je algoritmicky nerozhodnutelný

Důkaz sporem viz.:

http://cs.wikipedia.org/wiki/Probl%C3%A9m_zastaven%C3%AD

Třída složitosti

- množina problémů s podobnou složitostí

Definice: Třída složitosti je množina problémů, které můžou být vyřešeny abstraktním strojem M, za použití O(f(n)) zdrojů R, kde n je velikost vstupu.

- v současné době je evidováno kolem 450 tříd, jejich seznam je k nalezení na stránce: http://qwiki.caltech.edu/wiki/Complexity_Zoo

Převoditelnost problémů

Problém P_1 je polynomiálně převoditelný na problém P_2, jestliže existuje Turingův stroj M s polynomiální časovou složitostí, který pro libovolný vstup w problému P_1 sestrojí vstup w' problému P_2, přičemž platí, že odpověď na otázku problému P_1 pro vstup w je stejná jako odpověď na otázku problému P_2 pro vstup w'.

Úplnost problémů

O problému P_1 ve třídě složitosti X řekneme, že je X těžký, jestliže pro každý problém $P_2 \in X$ platí, že problém P_2 je polynomiálně převeditelný na problém P_1. Je-li navíc $P_1 \in X$, říkáme, že P_1 je X-úplný

Úplné problémy patří mezi nejtěžší v dané třídě složitosti.s

Vztahy mezi třídami složitosti

- i mezi nejdůležitějšími třídami složitosti jsou pochyby o vztazích mezi nimi
- většinou víme, která třída je větší a obsahuje méně náročnou třídu
- přesto je možné, že se někdy obě třídy rovnají

Nejdůležitější třídy složitosti

- nejdůležitější je třída P - obsahuje problémy, jejichž řešení je nalezitelné v polynomiálním čase a v praxi zvládnutelné
- zajímavá je třída NP, u které nebylo potvrzeno ani vyvráceno, zda je rovna třídě P

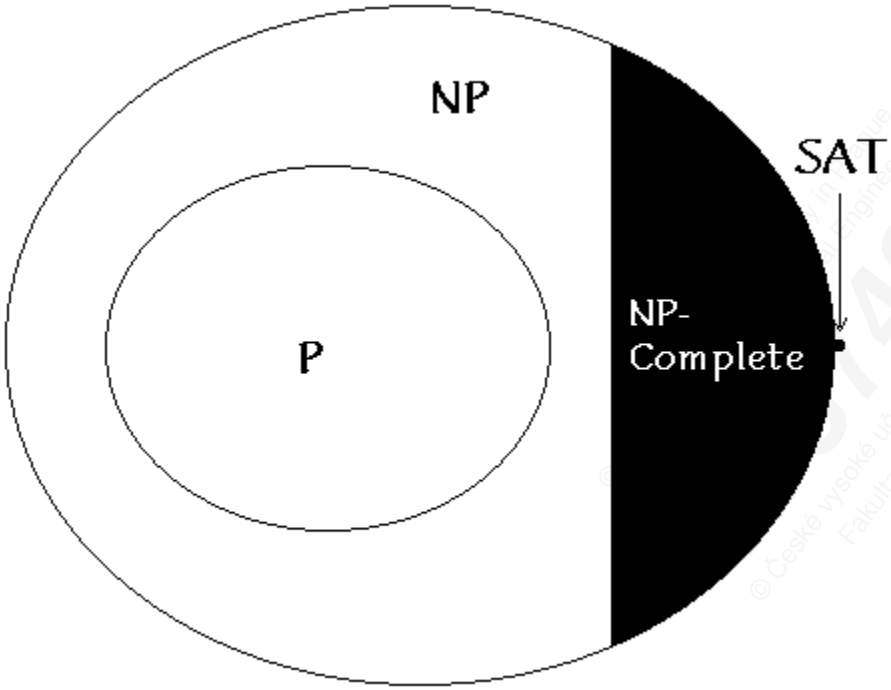
Třída P (PTIME)

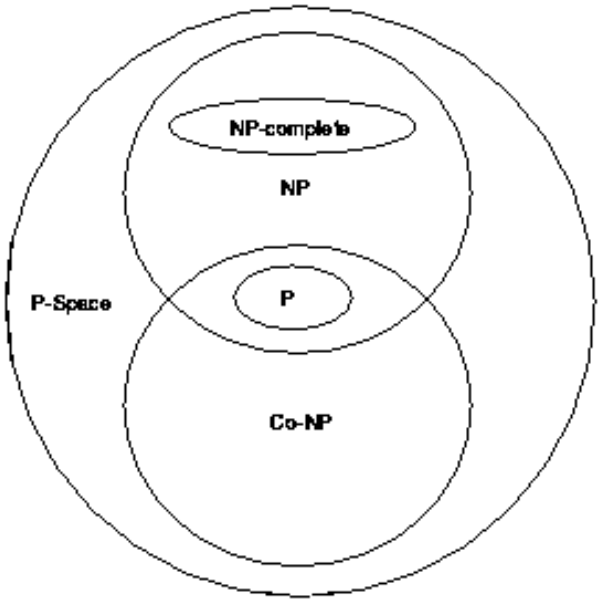
- obsahuje zvládnutelné problémy
- existují pro ní dostatečně rychlé, v praxi použitelné algoritmy
- třída P obsahuje problémy, které jsou řešeny deterministickými Turingovi stroji v polynomiálním čase a za použití neomezeného množství paměti

Třída NP (NPTIME)

- obsahuje problémy, které jsou řešeny nedeterministickými Turingovi stroji v polynomiálním čase
- výpočet může větvit do *n* cest
- při rozvětvení z nichž se každá možnost ověřuje deterministickým Turingovým strojem, když jedna uspěje, pak známe řešení
- cílem je efektivně najít možnou odpověď, která se dá v polynomiálním čase ověřit

Předpokládané vztahy P a NP k ostatním třídám





SAT - Boolean satisfiability problem (rozhodovací problém, pravdivost logické formule s operátory AND, OR, NOT)

- 1900 David Hilbert přednáška v Paříži na kongresu matematiků
- představil 23 v té době nevyřešených matematických problémů - „úkoly“ pro 20. století

http://aleph0.clarku.edu/~djoyce/hilbert/problems.html

- 22 vyřešeno – zbývá pouze Riemmanova hypotéza
- rozložení prvočísel na číselné ose (R. zeta funkce)
- důležité pro kryptografii

- 2000 - Clay Mathemathics Institute vyhlásil na počet D. Hilberta sedm problémů 3. tisíciletí
- 2 problémy mají přímou souvislost s kryptografií

http://www.kosmas.cz/knihy/125770/problemy-pro-treti-tisicileti/

(P = NP)?

1) Riemmanova hypotéza
2) vztah mezi třídami P a NP

Na vyřešení každé otázky byla vypsána odměna ve výši 1 000 000 $

- pravděpodobně se nerovnají
- pro vyřešení jsou důležité ty nejtěžší problémy v NP a to NP-úplné
- nalezení efektivní řešení NP-úplného problému znamená, že P = NP
- říká se, že „*rovnost by měla fatální dopady na kryptografii veřejného klíče*“
- http://www.win.tue.nl/~gwoegi/P-versus-NP.htm

Příklady P problémů

- test prvočíselnosti, hledání společného dělitele

Příklad: Výstupní hodnota logického obvodu

- známe strukturu logického obvodu, hodnoty vstupních proměnných
- v obvodu je jedna výstupní brána
- jaká je výstupní hodnota logického obvodu?

www.fit.vutbr.cz/~meduna/mti/2004_05/erlebach.pdf

Příklady NP úplných problémů

Příklad 1: problém obchodního cestujícího

- zadáno n-měst, vzdálenosti mezi nimi, délka požadované trasy
- lze navštívit všechny města právě jednou, tak aby nebyla překročena délka požadované trasy?

Příklad 2: problém batohu

- zadán objem batohu, objemy předmětů různé velikosti, požadované procento zaplnění
- je možné zaplnit objem batohu z X procent?

Třída co-NP

- problém X je členem třídy co-NP pouze tehdy, pokud jeho doplněk leží ve třídě NP
- vztahy k ostatním třídám:
  - P je podmnožinou NP a co-NP, rovnost je málo pravděpodobná
  - NP a co-NP by se rovněž neměly rovnat, což by znamenalo, že NP-úplný nemůže být v co-NP a naopak co-NP-úplný nemůže být v NP

Příklad Co-NP úplného problému:

Určení, zda-li je daná boolovská formule tautologií (tzn. bez ohledu na vstupní kombinací proměnných je výstup pravdivý) .

Třídy PSPACE a NPSPACE

- podle Savitchova teorému jsou si rovny
- obsahují množinu problémů, které můžou být vyřešeny deterministickými nebo nedeterministickými Turingovi stroji za použití polynomiální velikosti paměti a neomezeného množství času
- vztahy k ostatním třídám
  - $NL^* \subseteq P \subseteq NP \subseteq PSPACE = NPSPACE$

*NL – rozhodnutelné problémy řešitelné v logaritmickém prostoru nedeterministickým Turingovým strojem

Příklad PSPACE-úplný problém

- QBF (problém pravdivosti kvantifikovaných booleovských formulí)
  - formule $\exists x_1 \forall x_2 \exists x_3 \forall x_4 \ldots \exists x_{2n-1} \forall x_{2n} F(x_1, x_2, \ldots, x_n)$, kde $F(x_1, x_2, \ldots, x_n)$ je booleovská formule v konjunktivní normální formě
  - je daná formule pravdivá?
- další příklady: hry Sokoban, Mahjong

Třídy EXPTIME a EXPSPACE

- jedná se o prokazatelně nezvládnutelné problémy, neexistuje pro ně polynomiální algoritmus
- mají exponenciální složitost, např. 2^n
- vztahy k ostatním třídám:
  - $P \subseteq NP \subseteq PSPACE \subseteq EXPTIME \subseteq NEXPTIME \subseteq EXPSPACE$
  - $P \subseteq EXPTIME$, $PSPACE \subseteq EXPSPACE$, $NP \subseteq EXPSPACE$, $P \subseteq EXPSPACE$
  - pokud P=NP, pak EXPTIME = EXPSPACE

Příklady EXPTIME a EXPSPACE

- příklad EXPSPACE: RE^2 (ekvivalence regulárních výrazů s mocněním)
  - zadány dva regulární výrazy v nichž je možné použít mocnění (lze psát a^2 místo a.a)
  - reprezentují zadané výrazy tentýž jazyk?

- příklad EXPTIME: zobecněná hra (šachy,dáma, GO)
  - velikost hrací desky libovolná (n x n)
  - může první hráč vyhrát nezávisle na tazích soupeře?

Třída BPP

- BPP (Bounded error, Probabilistic, Polynomial time)
- problémy řešitelné v polynomiálním čase na pravděpodobnostních Turingových strojích
- pravděpodobnost chyby menší než 1/3
- omezíme ji několikanásobným provedením výpočtu

Vztahy k ostatním třídám:

- BPP = co-BPP, vztah BPP a NP nejasný

Třída BQP

- BQP - Bounded error, Quantum, Polynomial Time
- problémy řešené v polynomiálním čase kvantovými počítači
- pravděpodobnost chyby menší než ¼
- vztahy k ostatním třídám:
  - $BQP \supset BPP \supset P$
  - $BQP \subset PSPACE$