1 Složitost

1.1 Operační a paměťová složitost

Nezávislé určení na konkrétní implementaci

Několik typů operací = sčítání T+, logické T_L, přiřazení $T_{A(assign)}$, porovnání $T_{C(compare)}$, výpočet adresy pole prvku T_M

T(n) - Operační (časová) složitost – čas potřebný k provedení algoritmu

S(n) - Paměťová složitost – rozsah paměťového prostoru potřebný k provedení algoritmu

- *Časová složitost* (*time complexity*)
  = nejdelší čas výpočtu, který algoritmus zabere při vstupu velikosti n
  = čas výpočtu při nejméně příznivém případu, je jistou funkcí f čísla n
  ≈ počet operací (porovnání, …)
- *Paměťová složitost* (*space complexity*)
  = paměťové nároky programu.

Výpočet $f(n)$ může být složitý, proto se při odhadu času výpočtu zaměřujeme na dominantní složku v $f(n)$, která pro velká n reprezentuje téměř celý čas výpočtu.

⇒
myšlenka *asymptotických odhadů* – vyšetřovaná funkce $f(n)$ se porovnává s nějakou „jednodušší" funkcí $g(n)$ vyjadřující meze, v nichž se provedení výpočtu pohybuje.

1.2 Operační složitost v průměrném, nejhorším a nejlepším případě

$T_{min}(n)=T_{VAR}+T_{KON}$

$T_{MAX}(n)=nT_{VAR}+T_{KON}$

$T_{STR}(n)=T_{KON}+\sum(p(v_i)\cdot T_{VAR})=T_{KON}+1/n\cdot(\sum(T_{VAR}*i) = T_{KON}+T_{VAR}*(n+1)/2$

Průměrná časová složitost: Nechť V = {v_i}, i = 1,2,3,…a je prostor řešení algoritmu A, $p(v_i)$ je pravděpodobnost, že nastane řešení v_i ,Ti je časová složitost i-tého řešení. Pak průměrná čas, složitost je

$$Tstř(n) = \sum_{i=1}^{n} Ti \cdot p(vi)$$

1.3 Asymptotická složitost

Vyjadřuje se řádem růstu skutečně časově složitosti T(n) či paměťové S(n).

Def: f(n) roste max. resp. min. tak rychle jako g(n): f(n)=O(g(n)) resp. =Ω(g(n)), jestliže lze nalézt konstanty n0>1 a c>0 tak, že pro n>n0 ptatí |f(n)|<=c*g(n), resp. >=.

Pokud platí oboje, rostou řádově stejně, pak f(n)=Θ(g(n)) a říkáme g(n) je řádem růstu f(n) nebo též *f*(*n*) *roste asymptoticky tak rychle* jako *g*(*n*)
O(g(n)), Ω(g(n)) jsou řádem růstu horního a dolního odhadu.

1.2.1 Asymptotická složitost algoritmu

Nechť $f, g : \mathrm{N} \to \mathrm{R}^+$

Asymptotická horní mez $f(n)$ je shora omezena $g(n)$, $f(n) = O(g(n))$, pokud existuje kladné c takové, že pro všechna $n > n_0$ platí: $f(n) \leq c \cdot g(n)$.

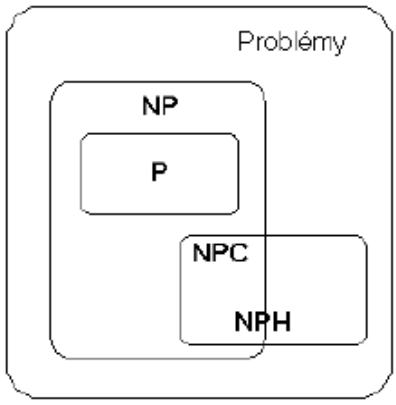
Asymptotická dolní mez $f(n) = \Omega(g(n)) \iff \exists c > 0, n_0 \in \mathrm{N} : \forall n > n_0 : f(n) \geq c \cdot g(n)$.

Asymptotický odhad $f(n) = \Theta(g(n)) \iff f(n) = O(g(n)) \wedge f(n) = \Omega(g(n))$, tzn. $c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$.

1.2.2 Složitost problému

Složitost problému se určuje jako složitost algoritmu, který ho řeší. Máme-li algoritmus $f(n)$, který řeší problém Π, pak je jeho složitost $O(f(n))$.

1.4 Třídy problémů P a NP:



1.4.1 definice

certifikát = struktura, která umožňuje ověřit, že odpověď ANO je správná

Turingův stroj - deterministický

Definice 11.11: Turingův stroj je uspořádaná pětice $M = \langle Q, A, \delta, s, F \rangle$, kde

- Q je konečná **množina stavů**
- A je **abeceda** obsahující mj. prázdný symbol (mezeru) $\sqcup$ a symbol konce pásky $\triangleright$, ale neobsahující symboly $\leftarrow$ a $\rightarrow$
- $s \in Q$ je **počáteční stav**
- $F \subseteq Q$ je množina **koncových stavů**
- $\delta : (Q - F) \times A \mapsto Q \times (A \cup \{\leftarrow, \rightarrow\})$ je **přechodové zobrazení** takové, že
  (a) pro všechna $q \in Q - F$ platí, že je-li $\delta(q, \triangleright) = (p, b)$, pak $b = \rightarrow$
  (b) pro všechna $q \in Q - F$ a $a \in A$ platí. že je-li $\delta(q, a) = (p, b)$, pak $b \neq \triangleright$

Definice 11.12: Konfigurací Turingova stroje $M = \langle Q, A, \delta, s, F \rangle$ nazýváme uspořádanou trojici

$$(q, \alpha, \beta) \in Q \times \triangleright A^* \times (A^*(A - \{\sqcup\}) \cup \{\epsilon\}).$$

Obsah pásky vyjádřený dvojicí (α, β) pro $\alpha = wa$, $w \in \triangleright A^*$, $a \in A$ lze přehledněji zapsat jako jediný řetěz, v němž podtrhneme právě čtený symbol, tedy jako $w\underline{a}\beta$. Konfiguraci (q, wa, β), $w \in \triangleright A^*$, $a \in A$ tedy budeme zapisovat jako $(q, w\underline{a}\beta)$. Je-li stav q konfigurace koncový, nazýváme celou konfiguraci $(q, w\underline{a}\beta)$ **koncovou**. Podobně nazýváme $(s, \triangleright \underline{\sqcup} w)$ **počáteční konfigurací** Turingova stroje M pro vstup w.

Páska

- G symbolů pásky, symbol B (prázdné políčko) b∉ G
- S množina vstupních symbolů S⊆G
- čtecí hlava
- konečný automat
  - vstup - symbol pod hlavou
  - množina stavů Q, q0 počáteční stav
  - Výstup, nový symbol na pásku, -1,1 pohyb hlavy
  - d: (Q\{qano, qne} x G -> Q x G x {-1,1}
- inicializace - stav q0, páska na políčku 0

Program M pro Tstroj, řeší rozhodovací problém P if se výpočet zastaví ve stavu qano pro všechny instance, které mají řešení ano, zastaví se v qne if pro instance ostatní

- řeší v čase t if se pro Ano instance zastaví nejdéle po t krocích
- řeší v paměťovém prostoru p if nejvýše p políček pásky obsahuje na konci symbol různý od b

- Třída **P** je tvořena takovými problémy M, pro které existuje program M pro T stroj takový, že jej řeší v čase omezeném polynomem.

- Třída **PSPACE** je tvořena takovými problémy M, pro které existuje program M pro T stroj takový, že jej řeší v prostoru omezeném polynomem.

- Třída **EXPTIME** je tvořena takovými problémy M, pro které existuje program M pro T stroj takový, že jej řeší v exponenciálním čase $2^{p(x)}$, kde p(x) je polynom

P <= PSPACE <= EXPTIME

P (polynomiální) – existuje alg. (program pro Turnigův stroj), který jej řeší v polynomiálně omezeném čase (počtu operací).

Třídou složitosti P nazýváme množinu všech rozhodovacích úloh řešitelných v polynomiálním čase. Tedy úlohu považujeme za řešitelnou v polynomiálně omezeném čase pokud existuje algoritmus, který ji řeší v čase $O(n^k)$ pro nějakou konstantu k.

Rozhodovací problém patří do *třídy P* jestliže pro něj existuje program pro deterministický Turingův stroj, který jej řeší v čase $O(n^k)$, kde n je velikost instance a k je konečné číslo.

Nedeterministický Turingův stroj

- model I = uhodne certifikát, deterministicky jej zkontroluje
- model II = konstruuje certifikát odhady (postupnými)
  - Program M pro nedet. T stroj řeší problém P if pro každou Ano instanci problému P zastaví ve stavu qano
  - řeší v čase t if se výpočet zastaví nejvýše po T krocích
- Třída **NP** Problém P patří do třídy NP iff existuje program M pro nedet. T stroj, který jej řeší v polynomiálním čase

Předpoklad oprávněný $P \subseteq NP$

NP (Non-deterministic P) – existuje prg. pro nedeterministický Turingův stroj, který ho řeší v polynom. čase.

neboli: Pro každou instanci rozhodovacího problému, jejíž řešení je ANO, existuje certifikát takový, že problém kontroly je P.

(Nedeterministický TS: - vycucá si řešení-certifikát z prstu (orákulum), normální TS certifikát ověřuje (nerozhoduje!))

NP – Při definici třídy složitosti **NP** hraje základní roli pojem nedeterministického algoritmu.

Nedeterministický algoritmus dovoluje obejít obtížnou část problému, nalezení předpokládaného řešení a deterministicky postupovat při jeho ověření. má 2 fáze

1) nedeterministická fáze - do paměti se při ní uloží nějaký řetězec znaků s, který můžeme považovat za (uhádnuté) předpokládané řešení. Obsah řetězu s se při různých výpočtech algoritmu může lišit
2) deterministická fáze - znamená provedení běžného deterministického algoritmu, který využívá vedle vstupních dat úlohy rovněž řetězu s. tuto fázi lze chápat jako ověření předpokládaného řešení s, které skončí rozhodnutím o výsledku.

Nedeterministický algoritmus nazveme polynomiálně omezený, pokud pro libovolná vstupní data délky n, pro která je výsledek ano, existuje výpočet algoritmu složitost $O(n^k)$ pro nějakou konstantu k.

2.1.4 Řešení problému nedeterministickým Turingovým strojem

Program M nedeterministického Turingova stroje *řeší rozhodovací problém* Π *v čase* t, jestliže se výpočet zastaví po t krocích pro každou instanci $I \in \Pi_{ANO}$ problému Π, kde Π_{ANO} je množina instancí problému Π takových, které mají výstup ANO.

Jestliže nedeterministický Turingův stroj řeší problém Π v čase $T(n)$, pak deterministický Turingův stroj řeší Π v čase $2^{O(T(n))}$.

2.1.5 Třída NP

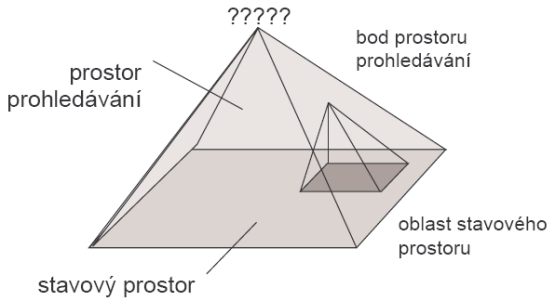
Rozhodovací problém Π *patří do třídy NP*, jestliže pro něj existuje program pro nedeterministický Turingův stroj, který každou instanci $I \in \Pi_{ANO}$ problému Π řeší v čase $O(n^k)$, kde n je délka vstupních dat a k je konečné číslo.

Rozhodovací problém Π *patří do třídy NP*, jestliže pro každou instanci $I \in \Pi_{ANO}$ problému Π existuje konfigurace Y taková, že kontrola, zda Y je řešením, patří do P (tzn. že omezující podmínky lze vyhodnotit v polynomiálním čase). V této souvislosti nazýváme Y *certifikátem*.

2.1.6 Třída co-NP

Existují i problémy mimo NP. Například rozhodovací problém, zda je graf prost Hamiltonových kružnic, nebo zda je booleovská formule nesplnitelná. Problémem je, že při odpovědi „ano, je nesplnitelná“ nemáme žádný certifikát, kterým bychom to doložili. Na takovéto problémy lze pohlížet jako na „protějšky“ NP problémů – označují se *co-NP*.

1.4.2 souvislost se stavovým prostorem



Stav Nechť $X = \{x_1, x_2, \ldots, x_n\}$ jsou konfigurační proměnné problému Π. Nechť $Z = \{z_1, z_2, \ldots, z_m\}$ jsou vnitřní proměnné algoritmu A řešícího instanci I problému Π. Pak každé ohodnocení s proměnných $X \cup Z$ je *stav* algoritmu A řešícího I.

Stavový prostor Nechť $S = \{s_i\}$ je množina všech stavů algoritmu A řešícího I. Nechť $Q = \{q_j\}$ je množina operací $S \rightarrow S$ takových, že $q_j(s_i) \neq s_i$ pro všechna s_i, q_j. Pak dvojici (S, Q) nazveme *stavovým prostorem* algoritmu A řešícího I.

1.4.3 polynomiální redukce

P1 je polynom. redukovatelný na P2, když existuje algoritmus (program pro Turnigův stroj), který převede
každou instanci P1 na insatnci P2 tak, že zachová řešení. Pol. redukce je tranzitivní, třídy P i NP jsou vůči ní uzavřené.

1.5 NP-úplné a NP-těžké problémy

nejobtížnější - NP úplné. každá jiná třída složitosti NP je na ně redukovatelná v polynomiálním čase. NP-úplnost splnitelnosti logických formulí. Základem je redukce. Máme li řešit nějakou úlohu, zkoušíme, zda není speciálním případem nějaké obecnější úlohy, jejíž algoritmus známe. Chceme řešit P1 a známe P2. navrhneme zobrazeni R, které pro vstup x je ano iff P2(x) dá ano. složením P2 a R dostaneme algoritmus P1.

def:
Nechť R je zobrazení množiny vstupů úlohy P1 do množin vstupů úlohy P2. Zobrazení R nazýváme **polynomiální redukcí** P1 na P2 if platí

1) R lze počítat v polynomiálně omezeném čase
2) Pro každý vstup x úlohy P1 je výsledek úlohy P2 na vstup R(x) roven výsledku úlohy P1 na vstup x

O úlohách P1 a P2, pro které existuje takové zobrazení R říkáme, že P1 je **polynomiálně redukovatelná** na P2, P1< P2

úlohu P patřící do třídy složitosti **NP** nazýváme **NP-úplnou**, pokud pro libovolnou úlohu Q ze třídy **NP** platí Q< P

X-těžký Problém Π je *X-těžký*, jestliže se efektivní řešení všech problémů ze třídy X dá zredukovat na efektivní řešení problému Π. Efektivním řešením je například řešení v polynomiálním čase nebo řešení s omezenou chybou. Zredukovat na řešení Π znamená vyřešit pomocí Π.

X-úplný Problém Π je X úplný, jestliže je X-těžký a sám patří do třídy X.

NPC (NP Complete = úplný) – problém Π je NPC když: je NP a každý NP problém je možné na Π polynomiálně redukovat

2.3.1 NPC problémy

Karpova redukce Rozhodovací problém Π_1 je Karp-redukovatelný na Π_2 (značí se $\Pi_1 \propto \Pi_2$), jestliže existuje polynomiální program pro deterministický Turingův stroj, který převede každou instanci I_1 problému Π_1 na instanci I_2 problému Π_2 tak, že výstup obou instancí je shodný.

Karpova redukce je *tranzitivní*, tedy platí, že $(\Pi_1 \propto \Pi_2) \wedge (\Pi_2 \propto \Pi_3) \Rightarrow \Pi_1 \propto \Pi_3$. Zároveň lze pomocí Karpovy redukce vytvářet *třídy polynomiální ekvivalence* ($(\Pi_1 \propto \Pi_2) \wedge (\Pi_2 \propto \Pi_1) \Rightarrow \Pi_1$ a Π_2 jsou polynomiálně ekvivalentní).

NP-úplný problém Problém Π je NP-úplný (NPC, NP-Complete), jestliže $\Pi \in$ NP a pro všechny problémy $\Pi' \in$ NP platí, že $\Pi' \propto \Pi$.

Důsledkem je, že máme-li problém $\Pi \in$ NP a existuje-li třeba i jediný $\Pi' \in$ NPC takový, že $\Pi' \propto \Pi$, pak z tranzitivity plyne, že $\Pi \in$ NPC (česky: když najdeme jeden NPC problém, který jde převést na náš problém Π, pak musí být náš problém taky NPC, jelikož všechny NP jdou převést na náš problém přes Π' ve dvou krocích).
NPC jsou nejtěžší, vzájemně převoditelné, problémy v NP.

Věta: Pokud optimalizační Π je NPH, pak jeho rozhodovací varianta je NPC.

2.3.2 NPH problémy

Turingova redukce Rozhodovací problém Π_1 je Turing-redukovatelný na Π_2 (značíme třeba $\Pi_1 \overset{T}{\propto} \Pi_2$, jestliže existuje polynomiální program pro (deterministický) Turingův stroj, který řeší každou instanci I_1 problému Π_1 tak, že používá program M_2 pro problém Π_2 jako podprogram (jehož trvání považujeme za jeden krok).

Karpova redukce je speciálním případem Turingovy redukce, kdy voláme podprogram pouze jednou a přímo používáme výsledek tohoto podprogramu.

NP-těžký problém Problém Π je NP-těžký (NPH, NP-Heavy), jestliže pro všechny problémy $\Pi' \in$ NP platí, že $\Pi' \overset{T}{\propto} \Pi$.

Z definice NPH problémů mimo jiné plyne to , že NPC $\subset$ NPH.

1.6 Cookův teorém

Věta: Problém splnitelnosti logických funkcí je NP-úplný (zkráceně: SAT je NPC).

Velice důležitá věta, která přináší řadu důsledků. Především říká, že NPC není prázdná a otevírá tak cestu k důkazům NP-úplnosti převodem. Jsou známy tisíce NPC problémů, které tvoří třídu ekvivalence. Nezdá se proto, že by třída problémů P byla shodná s NP.

Toho, že SAT je NP-úplný, a předchozího důsledku se využívá k dokazování, že je nějaký problém NP-úplný ($\Pi \in \mathrm{NP}, \mathrm{SAT} \propto \Pi \Rightarrow \Pi \in \mathrm{NPC}$).

1.7 Princip důkazu příslušnosti problému k třídě NP-těžký

Nechť P patří do třídy NP, potom je P NP-úplný iff existuje NP-úplný problém Q, Q< P.

Důkaz

1) je-li P NP-úplný pak za Q lze vzít P
2) nechť platí druhá část tvrzení a mějme libovolnou úlohu R ze třídy NP. Potom je R< Q a protože polynomiální redukovatelnost je tranzitivní je také R< P, takže P je NP-úplný