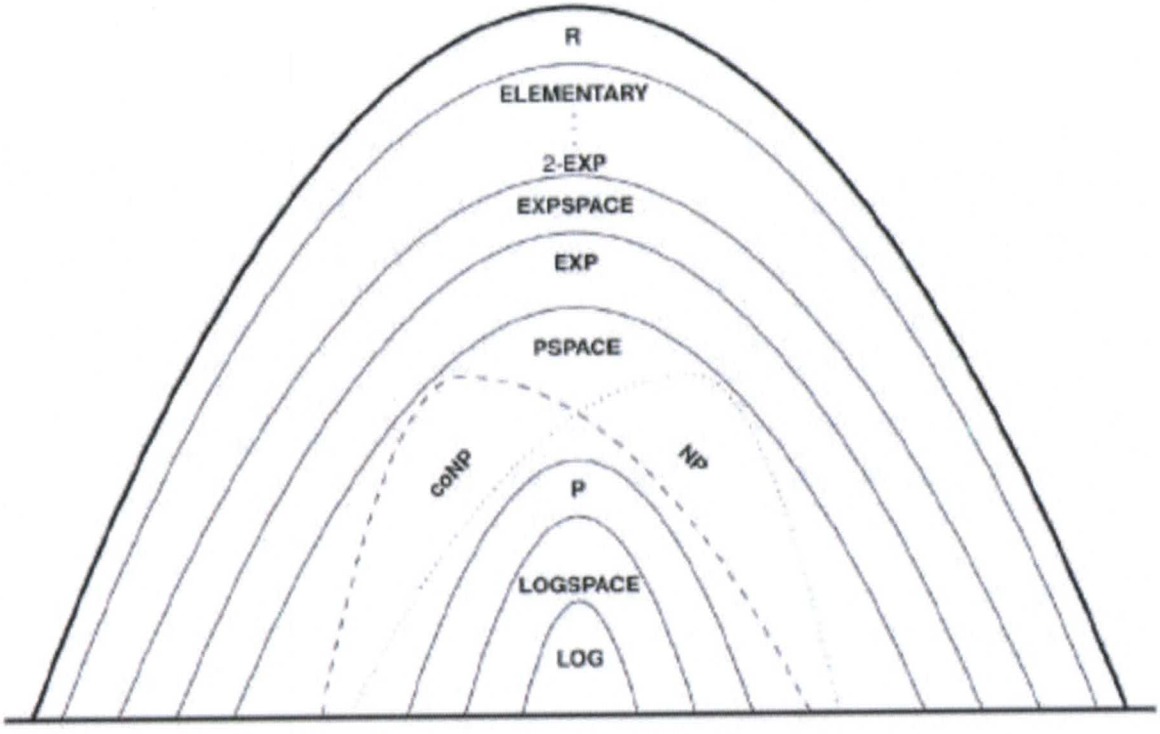
19. Pojem redukce a pojem úplného problému. NP-úplné, PSPACE-úplné a P-úplné problémy.

Pro zopakování, v rámci teorie složitosti zkoumáme, jak efektivně lze problém algoritmicky řešit. Je zavedeno mnoho složitostních tříd:



Sestavením algoritmu, který běží např. v polynomiálním čase na nedeterministickém Turingově stroji např. dokážeme příslušnost algoritmu do třídy NP. Nedokážeme tím však, že algoritmus nejde řešit efektivněji (např. v polynomiálním čase na deterministickém Turingově stroji – třída P).

K dokázání spodní hranice složitosti algoritmu se využívá technika porovnávání s jinými problémy, tzv. redukce. Redukce popisuje tvrzení, že nějaký problém je alespoň tak těžký jako nějaký jiný problém. Zároveň musí být zachována správnost odpovědi. Redukce z B do A znamená, že A je alespoň tak těžké jako B za předpokladu, že transformace instance problému B do problému A není příliš složitá. To vychází z toho, že pokud by A bylo lehčí, mohli bychom problém B redukcí převést na problém A a řešit jej tedy efektivněji, než bylo původní řešení problému B, což je spor.

Polynomiální redukce

Nechť $L_1 \subseteq \Sigma^*{}_1$ a $L_2 \subseteq \Sigma^*{}_2$ jsou dva jazyky. L_1 je redukovatelný na L_2 (zapisujeme $L_1 \leq L_2$) tehdy a jen tehdy, když existuje funkce R: $\Sigma^*{}_1 -> \Sigma^*{}_2$ taková, že:

1. R je spočitatelná deterministickým Turingovým strojem v polynomiálním čase
2. $w \in L_1 \Leftrightarrow R(w) \in L_2$ (zachování správnosti odpovědi – zachování příslušnosti do jazyka)

Redukce je však obecnější pojem a nemusí být pouze polynomiální, ostatní typy redukcí se liší pouze v bodě 1 – je zpřisněna/rozvolněna možná časová složitost redukční funkce (bude demonstrováno např. na P-úplnosti, viz dále).

Uveďme nyní příklad triviální redukce. Nechť $\Sigma = \{0,1,2,3,4,5,6,7,8,9\}$, L_1 je množina všech lichých čísel nad Σ, zatímco L_2 je množina všech sudých čísel nad Σ. Redukce R definovaná jako R(w) = w + 1, $w \in \Sigma^*$ je polynomiální redukce, protože w je sudé pravě tehdy, když w + 1 je liché a zároveň R může být na DTS spočítána v lineárním čase.

Redukce má několik základních vlastností. Jestliže $A \leq B$ a $B \in P$, pak také $A \in P$ (z důvodu, že vstup problému A můžeme převést polynomiální redukcí deterministicky v polynomiálním čase na vstup problému B a ten vyřešit v polynomiálním čase – celkově stále polynomiální). Zároveň je redukce tranzitivní. Také to znamená, že B je minimálně tak složité jako A, nebo složitější.

Těžké a úplné problémy

Nechť C je třída složitosti. Jazyk L označujeme jako C-těžký, jestliže pro každý $L' \in C: L' \leq L$, jinými slovy jazyk L je nejsložitější jazyk ze třídy C, tzn. každý jazyk z dané třídy je na něj možné redukovat. Dále pak označujeme jazyk jako C-úplny, jestliže je C-těžký a zároveň náleží do třídy C.

Uzávěr vzhledem k redukci

Nechť C je třída složitosti a $L \in C$. Třída C je uzavřena vzhledem k redukci, jestliže pro každý $L' \subseteq \Sigma^*: L' \leq L => L' \in C$. Třídy P, NP a PSPACE jsou uzavřeny na redukci, zatímco třeba třída LOGSPACE není.

Teorémy o úplnosti a těžkosti

Nechť C a C' jsou dvě složitostní třídy:

1. Nechť A je C-úplný a $A \leq B$ a $B \in C$. Pak B je také C-úplný.
2. Nechť A je C-úplný. Pak co-A je co-C-úplný.
3. Nechť A je C-těžký, $C' \subseteq C$, C' je uzavřená vzhledem k redukci a $A \in C'$. Pak C' = C.

NP-úplné problémy

Dle definici výše jsou NP-úplné problémy takové problémy, které náleží do třídy NP a zároveň jsou NP-těžké, tzn. je možné na ně redukovat všechny ostatní problémy ze třídy NP (jsou tedy nejtěžšími problémy ze třídy). To znamená, že neexistuje známý polynomiální algoritmus řešitelný deterministickým Turingovým strojem pro jejich řešení (kdyby existoval, pak P=NP a všechny NP-úplné problémy by bylo možné řešit v polynomiálním čase), jeho existence však není vyloučena (vyloučení by vyžadovalo důkaz $P \neq NP$). Neexistence polynomiálního algoritmu znamená, že pro řešení větších instancí problému nebude fungovat naivní brute-force algoritmus (exponenciální složitost) a pro praktické řešení takových problémů bude nutné využít nějakých heuristik nebo aproximace.

Jsou dva způsoby, jak dokázat, že je problém NP-úplný. V obou případech nejprve musíme dokázat, že problém náleží do třídy NP (zapsáním algoritmu, který běží v polynomiálním čase na NTS). NP-těžkost můžeme dokázat dvěma způsoby:

1. Pro každý jazyk L' z NP ukázat, že $L' \leq L$ (složité – tímto způsobem byla dokázana NP-úplnost problému SAT)
2. Pro nějaký NP-úplný jazyk L' ukázat, že $L' \leq L$, tzn. sestrojit redukci.

SAT

První problém, pro který byla dokázána NP-úplnost v rámci Cookova teorému. V SAT problému řešíme, jestli je zadaná formule výrokové logiky splnitelná, tzn. má model. Příslušnost do NP je triviální – sestrojíme NTS, který uhodne přiřazení proměnným a v polynomiálním čase pro dané ohodnocení ověří pravdivostní hodnotu formule. NP-těžkost byla ukázána tak, že pro libovolný NPTIME Turingův stroj M a jeho vstup w byla sestavena polynomiální redukce. Výstupní formule je splnitelná, právě když w patří do jazyka M. Redukce víceméně kóduje chování Turingova stroje do výrokové formule.

CNF

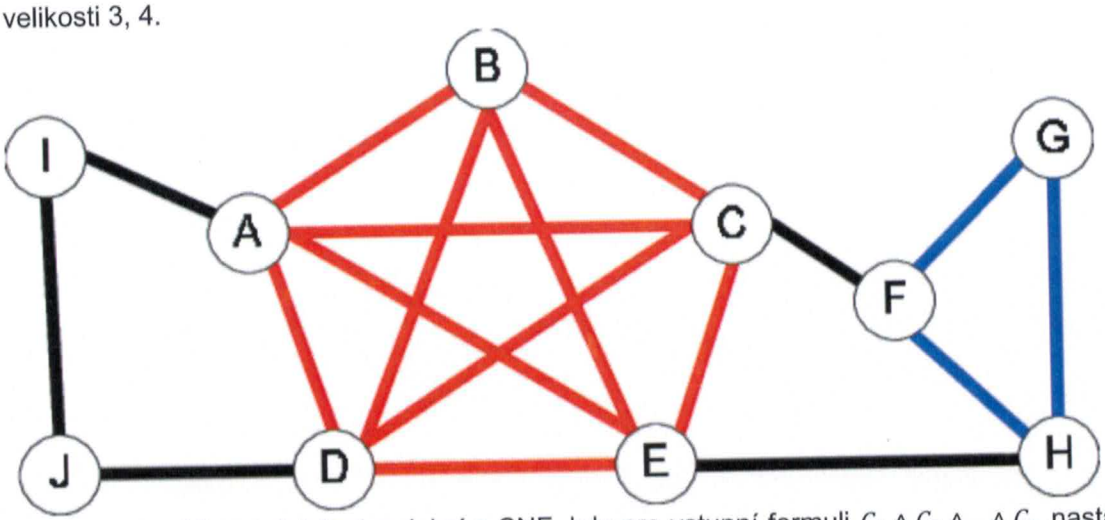
Je zadaná formule v konjukntivní normální formě (CNF) splnitelná? NP-těžkost můžeme ukázat tzv. Tseitinovou transformací, která převede v polynomiálním čase libovolnou formuli do CNF (naivní transformace pomocí algebraických operací, např. Demorganových a distributivních zákonů by mohla vést na exponenciální explozi formule – nejednalo by se o polynomiální redukci).

k-CNF

Podobně jako CNF, ale vstupní formule je omezena tak, že každá klauzule má právě *k* literalů. Tzn. např. 3-CNF má klauzule ve tvaru $x_1 \vee x_2 \vee x_3$. 2-CNF je možné řešit v polynomiálním čase, pro všechna ostatní k je problém NP-úplný. Důkaz NP-těžkosti: redukcí z CNF, např. v 3-CNF vytvoříme pro klauzuli typu $a \vee b \vee c \vee d$ klauzule $(a \vee b \vee x) \wedge (\neg x \vee c \vee d)$, což je ekvivalentní s původní formulí a má lineární velikost vzhledem k původní formuli.

CLIQUE

Nechť G = (V, E) je graf a $k \in N$. Problém CLIQUE se ptá, zda G obsahuje kliku (úplný podgraf – podgraf, kde každý uzel je spojený s každým) velikosti k. Obrázek níže má barevně zvýrazněné 2 kliky – kliku velikosti 5 v podgrafu A, B, C, D, E a kliku velikosti 3 v podgrafu F, G, H. Je zde ale obecně více klik: 10 klik velikosti 1, mnohé kliky velikosti 2, uvnitř kliky o velikosti 5 je několik klik velikosti 3, 4.



NP-těžkost můžeme dokázat redukcí z CNF, kdy pro vstupní formuli $C_1 \wedge C_2 \wedge \ldots \wedge C_n$ nastavíme k=n a vytvoříme graf G=(V, E), kde:

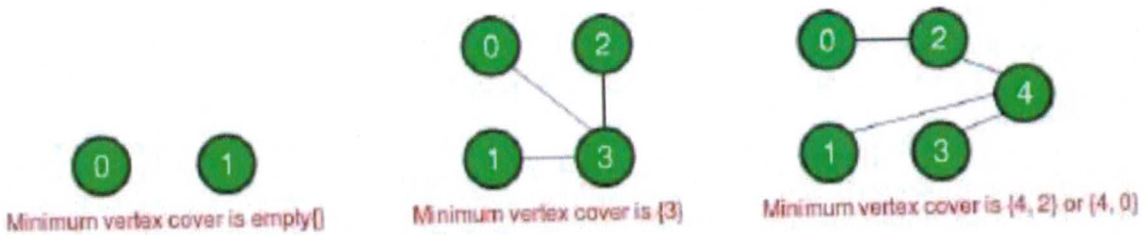
- V = $\{(\sigma, i) \mid \sigma \; je \; literál \; vyskytující \; se \; v \; C_i\}$
- E = $\{\{(\sigma, i), (\delta, j)\} \mid i \neq j \wedge \sigma \neq \neg\delta\}$ (spojíme literály, které nejsou v konfliktu)

INDEPENDENT SET

Nechť B = (W, J) je graf a $m \in N$. Problém INDEPENDENT SET se ptá, zda graf B obsahuje nezávislou množinu vrcholů (množinu vrcholů, kde žádné dva vrcholy nejsou spojeny) velikosti nejméně m. Důkaz NP-těžkosti: redukcí z CLIQUE – pro graf G = (V, E) a číslo k nastavíme m = k a sestrojíme graf $B = (V, V^2 - E)$. Graf má kliku právě tehdy, když jeho komplement má nezávislou množinu. Na obrázku výše můžeme vidět např. nezávislou množinu velikosti 3 – {I, D, G}.

VERTEX COVER

Nechť H = (U, F) je graf a $I \in N$. Problém VERTEX COVER se ptá, zda má graf H v sobě tzv. vertex cover (vrcholové pokrytí) velikosti nejvíce I. Tzn. jestli existuje množina vrcholů $S \subseteq U$ velikosti nejvíce I taková, že všechny hrany z H jsou incidentní nejméně s jedním uzlem S.



Důkaz NP-těžkosti: redukcí z INDEPENDENT SET – pro graf B = (W, J) a m nastavíme I = |W| - m a H = B.

GRAPH COLORING

Nechť M = (Y, L) je graf a $p \in N$. V rámci barvení grafu se ptáme, zda vrcholy M mohou být obarveny p barvami tak, že žádné dva sousedící uzly nemají stejnou barvu. Pro $p = 2$ je problém řešitelný v polynomiálním čase (umístíme jednu barvu a poté střídáme jediným možným způsobem, aby nebyly vedle sebe dva uzly stejné barvy) – graf je 2-obarvitelný, pokud je bipartitní. Problém je NP úplný pro 3 a více barev. Důkaz se provádí redukcí z 3-CNF.

HAMILTON PATH

V tomto problému řešíme, zda v zadaném grafu existuje Hamiltonovská cesta nebo kružnice, tzn. cesta, která prochází každým vrcholem právě jednou.

SUBSET SUM

Nechť S je konečná množina elementů, w je jejich váhová funkce, w: S -> Z a W nějaká zadaná cílová váha. SUBSET SUM se ptá, zda existuje podmnožina S' ⊆ S, taková, že součet vah elementů z S' je přesně W.

PARTITION

Nechť T je konečná množina prvků a v je jejich váhová funkce v: T -> Z. Problém PARTITION se ptá, zda existuje rozdělení množiny T na dvě disjunktní podmnožiny (T', T - T') takové, že jejich celková váha je stejná, tzn. $\sum_{t \in T'} \quad v(t) = \sum_{t \in T - T'} \quad v(t)$. NP-těžkost ukážeme redukcí ze SUBSET SUM. Pro elementy S, váhovou funkci w a cílovou váhu W nastaví T = $S \cup \{z\}$, kde $z \notin S$ a v = w ∪ $\{z -> \sum_{s \in S} \quad w(s) - 2W\}$. Jestliže původní SUBSET SUM má řešení s váhou W, pak můžeme rozdělit vytvořenou množinu T na dvě podmnožiny s váhou $\sum_{s \in S} \quad w(s) - W$.

KNAPSACK

Nechť R je konečná množina elementů, u je jejich váhová funkce, u: R->Z, v je jejich funkce hodnoty, v: R->Z, U maximální váha a V cílová hodnota. Problém KNAPSACK se ptá, zda existuje podmožina $R' \subseteq R$ taková, že $\sum_{r \in R'} \quad u(r) \leq U \wedge \sum_{r \in R'} \quad v(r) \geq V$. NP-těžkost můžeme ukázat

redukcí ze SUBSET SUM, kde pro elementy S, váhovou funkci w a cílovou váhu W nastavíme R = S, u = w, v = w, U = W, V = W.

PSPACE-úplné problémy

PSPACE je třída problémů, které jdou řešit s využitím deterministického Turingova stroje s využitím polynomiálního prostoru. Jedná se o problémy těžší než NP – pro třídu NP problémů existuje krátký důkaz pozitivní odpovědi, který je možné rychle (v polynomiálním čase) ověřit – např. máme-li zadán model SAT problémů, je velmi lehké ověřit, zda je to doopravdy model. Naopak pro problémy ze třídy PSPACE nemusí existovat krátký důkaz, důkaz ovšem může být vygenerovaný a zkontrolovaný s využitím polynomiálního prostoru.
Na základě Savitch teorému víme, že PSPACE = NPSPACE.

Quantified Boolean Formula (QBF)

PSPACE-úplný problém, jedná se o složitější variantu problému SAT – je zadána formule φ výrokové logiky nad proměnnými x_1, x_2, …, x_n a kvantifikátory Q_1, Q_2, …, $Q_n \in \{\forall, \exists\}$. QBF problém se ptá, zda je formule $Q_1 x_1 \ldots Q_n x_n$: φ platná. SAT je speciálním případem QBF, kde všechny kvantifikátory jsou existenční.

Důkaz náležitosti do PSPACE je možné provést eliminací kvantifikátorů – existenční kvantifikátor je nahrazen disjunkcí dvou členů – první odpovídá případu, že kvantifikovaná proměnná je rovna 1, druhý odpovídá případu, že kvantifikovaná proměnná je rovna 0. Podobně pro univerzální kvantifikátor, ovšem namísto disjunkce je použita konjunkce. Je potřeba udělat nejvíce n takových přepsání, každé potřebuje O(n) místa – celkove O(n^2) místa. Naivní řešení by spotřebovalo exponenciálně místa – je nutné využít backtracking.

Důkaz PSPACE-těžkosti je náročnější, podobně jako Cookův teorém zakódujeme běh Turingova stroje do formule, jejíž velikost je polynomiální vzhledem k velikosti TS a vstupního slova.

GAME

Hra je trojice (C, E, s), kde (C, E) je orientovaný graf a $s \in C$ je počáteční vrchol. Dva hráči se střídají v posunu žetonu po hranách grafu, každý vrchol může být navštíven jen jednou. Hráč vyhrává v momentě, kdy druhý hráč už nemůže žeton nikam posunout. Problem GAME se ptá, zda může první hráč vždy vyhrát. Problém je PSPACE-úplný. PSPACE-těžkost se dokáže redukcí z QBF, kdy kvantifikátory jsou převedeny do grafu – jeden hráč vybírá přiřazení existenčním kvantifikátorům, zatímco druhý hráč vybírá přiřazení univerzálním kvantifikátorům.

Další problémy

Mnoho dalších, v praxi zajímavých, problémů je PSPACE-complete, např.:

- Problém, zda slovo náleží do kontextové gramatiky

- Problém univerzality, inkluze, ekvivalence konečných a Buchiho automatů, minimalizace konečných automatů
- LTL model checking

P-úplné problémy

P-těžkost, resp. P-úplnost není definována vzhledem k polynomiální redukci, nýbrž vzhledem k tzv. NC-redukci. NC (Nick's class) je třída jazyků, které jsou rozhodnutelné v paralelním $log^{O(1)}n$ čase s využitím $n^{O(1)}$ procesorů. Jedná se tedy o třídu problémů efektivně řešitelných na PRAM (Parallel Random Access Machine) – mají polylogaritmickou složitost a využívají polynomiální počet procesorů. Zřejmě $NC \subseteq P$, podobně jako u problému $P \subseteq NP$ však nevíme, zda je inkluze ostrá, nebo ne.

Stejně jako v definice v úvodu této otázky je jazyk L P-těžký, jestliže $\forall L' \in P: L' \leq_{NC} L$. Je P-úplný, jestliže je P-těžký a náleží do P.

Circuit Value Problem (CVP)

Obvod nad proměnnými X je konečný orientovaný acyklický graf s označenými uzly:

- vstupní uzly jsou označeny proměnnými z množiny X nebo konstantami 0/1
- hradlové uzly mají jednu nebo více vstupní hranu a jsou označeny jednou z Booleovských funkcí
- výstupní uzel nemá výstupní hrany

Nechť C je obvod a x je vstup obvodu. (C, x) $\in CVP \Leftrightarrow$ C(x) = 1, tzn. obvod pro daný vstup dá na výstup jedničku. Tento problém je P-úplný.

Další problémy

Následující problémy jsou také P-úplné:

- rozhodnutí, zda slovo patří do bezkontextové gramatiky
- kontrola prázdnosti jazyka bezkontextové gramatiky
- kontrola konečnosti jazyka bezkontextové gramatiky

Pokud v textu najdete chybu, nebudete něčemu rozumět nebo budete mít dojem, že by bylo vhodné něco doplnit, kontaktujte na discordu uživatele Fifinas.