Těžké problémy

Představme si, že jsme v bludišti a hledáme (náš algoritmus hledá) nejkratší cestu ven. Rychle nás napadne, že bychom mohli použít prohledávání do šířky a cestu najít v čase lineárním ku velikosti bludiště. To je asymptoticky nejlepší možné řešení, v nejhorším případě bude totiž bludiště jedna dlouhá nudle a i nejkratší cesta bude dlouhá lineárně vůči velikosti bludiště.

Ve skutečném životě však „kulišáci“ znají lepší řešení – podvádět! Prostě si od kamaráda půjčíme mapu bludiště s vyznačenou nejkratší cestou a pak poběžíme hned tou nejkratší cestou, aniž bychom kdekoli ztráceli čas.

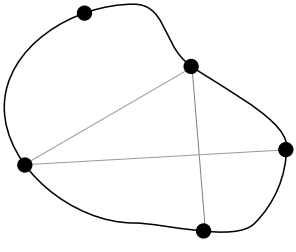
V nudlovém bludišti (nejkratší cesta má zhruba stejně vrcholů jako celý graf) jsme si vůbec nepomohli (takže je řešení asymptoticky stejně dobré). V alespoň trochu spletitém bludišti už budeme v cíli dříve než náš kamarád, který bloudí (prohledává) do šířky.

Existují tedy problémy, kde by se i v nejhorším možném případě vyplatilo podvádět pomocí taháku? Ano, zde je příklad – opět jsme v labyrintu, ale tentokrát jsou na všech stanovištích umístěny koláčky. Labyrint je to zvláštní, cesty se v něm nekříží, ale je tam plno nadchodů a podchodů.

Naším cílem je najít okružní cestu ze startovního místa zpátky na start, abychom každé stanoviště s koláčkem prošli právě jednou (protože víc než jeden koláček nám nedají).

Kdybychom tady chtěli použít procházení do šířky, bylo by to opět možné – ale tentokrát bychom se museli mnohokrát vracet, protože posloupnost stanovišť (začátek, první, druhé) může být špatná, zatímco posloupnost (začátek, druhé, první) už může být dobrá.

Přesněji řečeno, už by neplatilo, že při prohledávání do šířky každé stanoviště navštívíme nejvýše jednou, ale každou *posloupnost* stanovišť navštívíme nejvýše jednou. Projít všechny nám potrvá, matematicky řečeno, exponenciálně mnoho času vůči velikosti bludiště.



Kamarád s tahákem je na tom pořád dobře – prostě si pořídí jiný plán, na kterém bude mít vyznačenou cestu, po které má jít, aby vyhrál.

Ta cesta má stejně křižovatek jako bludiště samo, a tak bude jeho nápověda lineárně velká vůči velikosti bludiště a průchod napovězenou cestou bude trvat také lineárně. Podvodník tedy vyhrává i asymptoticky. Bídák!

Všechny by nás zajímalo, jestli by bylo možné najít tu nejlepší cestu bez podvádění v rozumně krátkém (řekněme polynomiálním) čase. Tato otázka je ekvivalentní známé otázce P vs. NP. Pojďme ta tajemná písmena přesně definovat.

Podvádíme s certifikáty

V teorii složitosti se často omezujeme jen na jeden typ problému, takzvaný *rozhodovací problém*. To je vlastně otázka, na kterou existují dvě možné odpovědi: ANO

a NE. Například „Existuje cesta z bludiště délky k?“ nebo „Je součet čísel $8 + 3$ roven 5?“

Ve zbytku kuchařky už budeme pracovat jen s nimi – skoro vždy se rychlé řešení rozhodovacího problému dá převést na rychlé řešení příslušného vyhledávacího problému, jako *Nalezněte nejkratší cestu z bludiště.*

Rozhodovací problém (dále už jen problém) bude náležet do *třídy problémů P* (třída je zde jen pomocné označení pro nekonečnou množinu), pokud existuje polynomiální algoritmus, který pro zadaný vstup odpoví korektně ANO nebo NE.

Taháku z předchozí kapitolky se v literatuře říká *certifikát*. Formálně to je jen jakási polynomiálně velká informace. Můžeme si jej představit jako data, která náš program nalezne v „našeptávajícím“ vstupním souboru, ke kterému program z třídy P nemá přístup.

Problém bude náležet do *třídy problémů NP* (nepoctivci), pokud existuje algoritmus a ke každé odpovědi ANO vhodný certifikát tak, že algoritmus je schopen pomocí certifikátu ověřit, že odpověď je skutečně ANO. Čili má-li ten program správný tahák, musí být schopen bludištěm projít rychle.

Zde si dejme pozor na to, že definice nedovoluje „podvádět na druhou“ – nemůžeme si do pomocného souboru prostě uložit ANO a pak jej vypsat. Tak by se pak dal řešit libovolně složitý problém, i problémy mimo třídu NP! Jen na okraj – takové opravdu existují.

Onen algoritmus musí být schopen řešení ověřit, tedy odpovědět ANO tehdy a jen tehdy, pokud mu to napověděl certifikát a odpověď je správná. Kdyby byla skutečná odpověď NE a certifikát chybně tvrdil, že ANO, algoritmus musí být napsán tak, aby oznámil NE.

Co přesně bude certifikát, záleží na zadané úloze – často to bývá právě ono nejlepší možné řešení, kterého se stačí držet a najdeme hledanou odpověď (nebo zjistíme, že úloha nemá řešení).

Asi vám bylo hned jasné, že každý program z P patří také do NP – jakmile známe polynomiální řešení bez nápovědy, certifikátem může být i třeba prázdný soubor! Horší je to s problémy, pro které potřebujeme pro polynomiální vyřešení nějaký certifikát a zatím to lépe neumíme.

Příkladem buď problém z povídání o bludišti. Říká se mu *Hamiltonovská kružnice.*

Název problému: Hamiltonovská kružnice

Vstup: Neorientovaný graf.

Problém: Existuje v zadaném grafu kružnice procházející všemi vrcholy právě jednou?

Certifikát: Posloupnost vrcholů hamiltonovské kružnice.

Ověření v polynomiálním čase s certifikátem: Projdeme postupně vrcholy a ověříme, že jsou opravdu zapojeny do kružnice a kružnice je správné délky. Vrátíme NE, pokud tomu tak není.

Zatím nikdo nepřišel s řešením, které by nepoužívalo vůbec žádný certifikát. Dokonce zatím nikdo nenalezl problém, který by byl v NP, ale bez certifikátu už jej nelze řešit v polynomiálním čase. Kdyby takový neexistoval, třídy P a NP by se rovnaly. To je jádro otevřeného problému P vs. NP.

Převoditelnost a NP-úplnost

Když řešíme nějakou algoritmickou úlohu, obvykle přijdeme na nějaké přímé řešení využívající základních technik (prohledávání do šířky, dynamické programování, zametání přímkou). Vzácně se může i stát, že v problému rozpoznáme problém jiný – občas lze geometrický problém převést na třídění čísel nebo umíme popsat situaci nějakým vhodným grafem.

Ukazuje se, že se ve třídě NP často vyplatí problémy převádět, neboť přímá řešení neznáme. Dokonce tak můžeme i zjistit, do které z probíraných tříd problém patří.

Převodem budeme rozumět polynomiální algoritmus, který upraví vstup jednoho problému na vstup jiného problému. Musí navíc problémy převést tak, aby správná odpověď (ANO nebo NE) na vstup prvního problému byla tatáž, jako správná odpověď na vstup druhého problému.

Jednoduchým převodem je úprava problému *Existuje cesta z bludiště ze zadaného políčka délky d?* na *Existuje cesta v grafu délky c začínající v zadaném vrcholu?*.

Do výstupního grafu za každou křižovatku dáme vrchol, za každou cestu mezi křižovatkami hranu a ke hraně si poznamenáme, jak dlouhá byla. Hodnotu c pak můžeme nechat stejně velkou, jako d.

Pokud najdu správnou cestu v tomto grafu, pak nutně podobná cesta je i v bludišti, a pokud cesta v grafu není, pak není ani v bludišti. Převod je tedy korektní.

Nadefinujme si nyní pojem, který nám bude sloužit jako zkratka za to, že problém je ve třídě NP a je alespoň tak těžký jako ostatní problémy v NP. Nemůžeme jen tak ledabyle říci „je v NP a není v P“, protože to nevíme. To je právě ta slavná otázka.

Uděláme tedy krok stranou – budeme říkat, že problém je *NP-úplný,* pokud onen problém je v NP a zároveň jdou všechny ostatní problémy v NP převést na tento problém.

Všechny problémy v NP na něj jdou převést? Pokud tuto definici vidíte poprvé, asi to působí dost zvláštně – je těžké si představit, že všechny grafové, geometrické, počítací problémy, o kterých víte, že jsou v P (a tedy i v NP) jdou převést na nějaký NP-úplný superproblém.

Ale je to správně, ba co víc, Cookova věta říká, že existuje alespoň jeden takový problém. (Samotná definice NP-úplného problému nezaručuje, že takový problém vůbec existuje.)

Ukazuje se však, že není sám, jsou jich stovky. Dokazovat existenci dalších NP-úplných problémů je však o dost lehčí, než dokázat Cookovu větu! Stačí totiž jen najít následující dva kroky:

- Dokázat, že problém je v NP – najít certifikát a polynomiální algoritmus, co jej využívá.

- Převést zadání libovolného NP-úplného problému na zadání našeho problému tak, že náš algoritmus vlastně vyřeší onen NP-úplný problém.

To postačí, protože pak libovolný jiný problém v NP nejprve převedeme na zvolený NP-úplný problém a pak pustíme námi vymyšlený převod. Zřetězení dvou polynomiálních algoritmů (převodů) je opět polynomiální algoritmus, takže podmínka převoditelnosti je splněna.

Ukážeme si důkaz NP-úplnosti jednoho problému na příkladu, pokud nám uvěříte, že již probíraný problém *Hamiltonovská kružnice* je NP-úplný. Nejprve zadefinujme jiný problém:

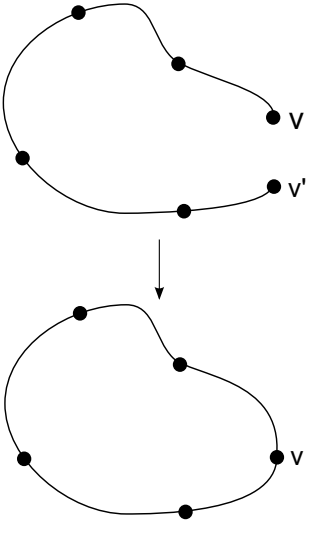
Název problému: Hamiltonovská cesta.

Vstup: Neorientovaný graf, dva speciální vrcholy x a y.

Problém: Existuje cesta z x do y (posloupnost vrcholů, ve které se žádné dva neopakují), která prochází každým vrcholem právě jednou?

Certifikát: Posloupnost vrcholů tvořící správnou cestu.

Řešení v NP: Projděme cestu z certifikátu a ověřme, že vrcholy jdou za sebou, je jich správný počet a žádný jsme nevynechali.



Důkaz NP-úplnosti: Převedeme předchozí problém (hamiltonovskou kružnici) na hledání hamiltonovské cesty. Uvažme graf G, ve kterém chceme najít hamiltonovskou kružnici.

Vyberme si libovolný vrchol v a vytvořme vrchol v', který bude kopií vrcholu v – do grafu přidáme hranu mezi u a v', pokud už v něm je hrana mezi u a v.

Na upravený graf zavoláme řešení problému *Hamiltonovská cesta* mezi vrcholy v a v'. Pokud taková cesta existuje, tak nutně v původním grafu G existuje hamiltonovská kružnice.

Cesta z vrcholu v' přesně odpovídá pokračováni kružnice poté, co přijde do vrcholu v.

Pseudopolynomiální algoritmy

Znáte problém batohu? Jeho varianty jsou oblíbené na programovacích soutěžích. Zadat se může třeba takto: mějme na vstupu seznam n dvojic kladných přirozených čísel, kde každá dvojice označuje váhu a cenu nějakého předmětu. Nakonec dostaneme na vstupu ještě číslo b, které udává nosnost našeho batohu.

Otázka zní: Jaký je nejcennější možný náklad, který přesto nepřesahuje váhový limit batohu?

Možná víte, že úloha jde řešit dynamickým programováním – vytvořím si pole `podbatoh[]` od 1 do b, kde `podbatoh[`i`]` je maximální hodnota, kterou bych si odnesl

v batohu o nosnosti i. Postupně od první věci do poslední pak projdu celé pole `podbatoh[]` „zprava doleva“ od b do 1 a zkusím, jestli je výhodnější do batohu vložit novou věc a volné místo doplnit starými (optimální volné místo pro předchozí věci máme napočítané), nebo si nechat jen ty staré. Tuto hodnotu pak zapíšeme jako aktuální pro váhu i na místo `podbatoh[`i`]`.

Po n průchodech tohoto pole dostaneme řešení pro všechny věci dohromady na políčku `podbatoh[`b`]`. Celková složitost je $\mathcal{O}(nb)$, to je polynom, algoritmus je tedy polynomiální.

Světe div se, toto řešení je ve skutečnosti exponenciální. Kde jsme v řešení udělali chybu? Nikde – naše složitost závisela na b, ovšem když se podíváme do vstupních dat, tak pokud jsou zapsána v binárním (nebo ternárním a vyšším) tvaru, tak zápis čísla b byl veliký $\mathcal{O}(\log_2 b)$, ale naše složitost závisela na $b = 2^{\log_2 b}$, tedy exponenciálně vůči velikosti vstupu.

Problém batohu, respektive jeho rozhodovací verze, je dokonce NP-úplný problém.

Algoritmům, které řeší nějakou úlohu a jsou polynomiální oproti *hodnotě* čísel na vstupu, ale exponenciální ve *velikosti zápisu* těchto čísel, říkáme *pseudopolynomiální algoritmy*. Některé další NP-úplné problémy mají pseudopolynomiální řešení (jako například *Dva loupežníci* níže), ale dá se dokázat, že na některé jiné problémy pseudopolynomiální algoritmus neexistuje (pokud P $\neq$ NP).

Mimochodem: pokud bychom na vstupu zapisovali čísla v unárním zápisu, každý pseudopolynomiální problém by ležel v P.

Poznámky na závěr

Otázku „Je třída P rovna NP?“ se již snažilo rozlousknout mnoho matematiků a informatiků. Tato teorie přinesla spoustu zajímavých výsledků, například už se podařilo dokázat, že některými technikami tuto domněnku nelze nikdy dokázat, ani vyvrátit.

Kdyby platilo P = NP, pak by mnoho lidi zajásalo – mnoho přirozených problémů, které nastávají i v reálném životě, by najednou byla řešitelná rychle. Navíc by krachlo dosavadní šifrování a bylo by možné najít rychle důkaz ke každému pravdivému tvrzení výrokové logiky.

Tato rovnost by se dala hypoteticky ukázat velice snadno – stačilo by najít jeden polynomiální algoritmus pro libovolný NP-úplný problém! Většina informatiků studujících složitost se však domnívá, že se třídy nerovnají.

To ale neznamená, že si to nemáte zkusit dokázat! Naopak, bojovat s NP-úplnými problémy je užitečné i v reálném světě – například jde mnohdy vymyslet dobrá aproximace NP-úplného problému.

Například nenajdeme hamiltonovskou kružnici v polynomiálním čase, ale nalezneme nějakou relativně dlouhou kružnici, která nám v praxi může stačit, pokud podle ní třeba chceme vést náročný cyklistický závod.

O aproximacích už toho bylo napsáno mnoho, viz například [ADS2]. O NP-složitosti můžete něco najít tamtéž, nebo zkuste [Algo].

Existují i problémy, které jsou mimo P i NP, a dokonce existuje spousta různých dalších tříd problémů. Je jich celá zoologická zahrada – můžete ji najít na internetu.

Seznam NP-úplných problémů

Sedíte-li nad zatím nevyřešenou úlohou, kterou jste nalezli jinde než v KSP, pak se klidně může stát, že bude NP-úplná. Abyste mohli mezi NP-úplnými úlohami převádět, tak je dobré znát jich aspoň hrstku, podle toho, je-li problém grafový, rovnicový a tak dále.

V následujícím seznamu najdete několik úloh, které jsou zaručeně NP-úplné. Převody se nám sem sice nevešly, ale mnoho z nich (ne-li všechny) zvládnete vymyslet sami – zkuste si to!

Název problému: Hamiltonovská kružnice

Vstup: Neorientovaný graf.

Problém: Existuje v zadaném grafu kružnice procházející všemi vrcholy právě jednou?

Název problému: Hamiltonovská cesta.

Vstup: Neorientovaný graf, dva speciální vrcholy x a y.

Problém: Existuje cesta z x do y (posloupnost vrcholů, ve které se žádné dva neopakují), která prochází každým vrcholem právě jednou?

Název problému: Splnitelnost

Vstup: Logická formule. Tu tvoří proměnné a logické spojky negace $\neg$, konjunkce $\wedge$ a disjunkce $\vee$. Například

$$(x \wedge (\neg y)) \vee z.$$

Problém: Můžeme proměnným přiřadit hodnoty 0 nebo 1 tak, že výsledná vyhodnocená formule má hodnotu 1?

Název problému: Součet podmnožiny

Vstup: Seznam nezáporných celých čísel, speciální číslo k.

Problém: Existuje podmnožina čísel, jejíž součet je přesně k?

Název problému: Batoh

Vstup: Seznam dvojic nezáporných čísel, kde dvojice označuje hodnotu a váhu předmětu. Přirozené číslo b – nosnost batohu, přirozené číslo k.

Problém: Umíme vložit do batohu předměty o hodnotě alespoň k, aniž bychom překročili limit váhy b?

Název problému: Dva loupežníci

Vstup: Seznam nezáporných celých čísel.

Problém: Existuje rozdělení seznamu na dvě hromádky tak, že každé číslo bude v právě jedné hromádce a v každé hromádce bude stejný součet čísel?

Název problému: Klika

Vstup: Neorientovaný graf, číslo k.

Problém: Existuje v grafu úplný podgraf o velikosti k, tedy k vrcholů takových, že mezi každými dvěma z nich vede hrana?

Název problému: Nezávislá množina

Vstup: Neorientovaný graf, číslo k.

Problém: Existuje v grafu prázdný podgraf o velikosti k, tedy k vrcholů takových, že žádné dva z nich nejsou spojeny hranou?

Název problému: Trojbarevnost grafu

Vstup: Neorientovaný graf.

Problém: Lze vrcholy tohoto grafu obarvit třemi barvami tak, že každá hrana sousedí s vrcholy dvou různých barev?

Název problému: Rozparcelování roviny

Vstup: Seznam bodů v rovině, kde každý má navíc přiřazenu jednu z b barev, číslo k.

Problém: Umíme rozdělit rovinu pomocí k přímek tak, že v každé oblasti jsou jen body stejné barvy?

Název problému: 3D párování

Vstup: Seznam mužů, žen a zvířátek, následovaný seznamem kompatibilních trojic tvaru $\{\text{muž}, \text{žena}, \text{zvířátko}\}$. Tyto trojice říkají, která trojice muž, žena a zvířátko by se dohromady snesla.

Problém: Můžeme všechny muže, ženy a zvířátka z prvního seznamu rozdělit do trojic tak, že každá trojice je kompatibilní a každá bytost je právě v jedné trojici?

Martin Böhm

Úloha 23-5-5: NP-úplný metr

Následující problém si pojmenujeme Metr a vaším úkolem je dokázat, že je NP-úplný.

Jistě znáte skládací metry. Mají typicky pět článků po dvaceti centimetrech. Mějme metr nepravidelný, jehož jednotlivé články jsou různě dlouhé. Tyto délky dostaneme v pořadí na vstupu, stejně tak délku pouzdra, do kterého bychom chtěli metr uložit.

Podaří se nám to? Pro články délek 6, 3, 3 a pouzdro délky 6 odpověď jistě zní ANO, pro vstup 6, 3, 4 a stejně dlouhé pouzdro to už ale NEpůjde.