Základy kombinatoriky a teorie grafů

Cvičení #11 – P vs. NP

Příklady

Jsou to jenom přepsané příklady z Průvodce labyrintem algoritmů, pořádné vysvětlení a často i hinty nebo řešení najdete tam.

1. Tahle úloha se trochu špatně zadává přesně, takže to bude delší. Kdyby vám to přesto nebylo jasné, ptejte se :). Turingův stroj má jednu nekonečnou pásku (pozice jsou, řekněme, očíslované přirozenými čísly), na níž se nacházejí nuly a jedničky. Vstup začíná na pozici nula, někde končí a dál pokračují samé nuly.[1] Často je potřeba na vstupu zadat přirozené číslo n předem neznámé velikosti. Jak to udělat, pokud můžete použít $2\lfloor\log_2(n)\rfloor + c$ bitů, kde c je nějaká konstanta? Uměli byste to s $\lfloor\log_2(n)\rfloor + o(\log(n))$ bity?

   Správné řešení není napsat prostě na pásku binární kód čísla, protože pak nelze poznat, kde končí číslo a kde začínají nuly značící prázdné pozice. Správné řešení není ani napsat binární kód a zakončit ho jedničkou – Turingův stroj nemá v konečném čase možnost poznat, jestli poslední jednička, kterou přečetl, je ta ukončovací, anebo jestli ještě někde napravo od aktuální pozice budou další data.

2. Převeďte SAT $\rightarrow$ 3-SAT. Jinak řečeno, najděte polynomiální algoritmus, který na vstupu dostane CNF formuli φ (např. $(x_1 \vee x_7 \vee \neg x_9 \vee x_{12}) \wedge (x_1 \vee \neg x_7) \wedge \cdots$) a vyrobí jinou CNF formuli ψ (klidně s jinými proměnnými), v níž má každá klauzule nejvýše tři literály (tj. $(x_1 \vee \neg x_2 \vee x_3)$ je povolené, $(x_1 \vee x_2 \vee x_3 \vee \neg x_4)$ není) a která je splnitelná právě tehdy, když φ je (tj. existuje přiřazení proměnných φ takové, že s ním je φ pravda, právě když existuje přiřazení proměnných ψ s nimiž je ψ pravda).

3. Převeďte 3-SAT na nezávislou množinu. Tj. najděte polynomiální algoritmus, který na vstupu dostane 3-CNF formuli φ a vyrobí graf G a číslo k takové, že G obsahuje nezávislou množinu velikosti (alespoň) k, právě když φ je splnitelná.

4. Převeďte nezávislou množinu na SAT. Tj. najděte polynomiální algoritmus, který na vstupu dostane graf G a číslo k a vyrobí CNF formuli φ, která je splnitelná, právě když G obsahuje nezávislou množinu velikosti k.

5. Problém kliky dostane na vstupu graf G a číslo k a ptá se, zda G obsahuje kliku (úplný graf) velikosti alespoň k. Převeďte problém kliky na problém nezávislé množiny a naopak.

6. Předpokládejte, že máte k dispozici černou skříňku, která umí vyřešit problém kliky v čase 1. Vyřešte v polynomiálním čase s pomocí (opakovaného) volání této skříňky problém, kdy na vstupu dostanete graf G a máte zjistit velikost největší kliky v G.

7. 3,3-SAT je další omezení 3-SATu, kde je navíc slíbené, že každá proměnná se vyskytuje v nejvýše třech klauzulích (ať už v negaci nebo gaci). Převeďte 3-SAT na 3,3-SAT.

8. E3,E3-SAT je omezení 3,3-SATu, kde navíc slibujeme, že každá klauzule má právě tři literály a každá proměnná se vyskytuje v právě třech klauzulích. Dokažte, že E3,E3-SAT je polynomiálně řešitelný, a to proto, že každá taková formule je ve skutečnosti splnitelná.

9. Dokažte NP-úplnost problému 3-obarvitelnosti grafu (tj. na vstupu je neorientovaný graf G a máte rozhodnout, zda je 3-obarvitelný). To znamená dokažte, že je v NP a převeďte na něj nějaký NP-úplný problém (doporučuju 3-SAT).

10. Ukažte, že problém 2-obarvitelnosti grafu leží v P (najděte efektivní algoritmus).

11. V problému součtu podmnožiny dostanete zadanou (multi)množinu přirozených čísel a přirozené číslo S a máte rozhodnout, zda existuje její pod(multi)množina, která má součet právě S. Dokažte, že jde o NP-úplný problém. Všechna čísla jsou zadaná v bínárce. (Což znamená, že do velikosti vstupu číslo n přispěje jen $\log(n)$ bity.)

[1] Tohle je jedna možnost, jak zavést Turingův stroj, tady to píšu jen kvůli motivaci problému.

12. Problém batohu: Na zkoušku z kombinatoriky a teorie grafů máte povolené omezené množství taháků, konkrétně můžete mít W slov (na vstupu v binárce). Protože jste poslední, kdo ji ještě nemá, nasbírali jste taháky od všech svých spolužáků. Každý tahák má nějaké množství slov a nějakou hodnotu (na vstupu v binárce). Víte, že abyste zkoušku dali, potřebujete mít taháky se součtem hodnot alespoň V (na vstupu v binárce), a jelikož řešíte tuhle úlohu, nemáte už čas dělat si vlastní. Otázka je, zda umíte vybrat takovou podmnožinu taháků, že má dohromady nejvýše W slov a hodnotu alespoň V.

    Převeďte součet podmnožiny na problém batohu a dokažte, že problém batohu je NP-úplný.

Hinty

Stránky znamenají strany v Průvodci labyrintem algoritmů, konkrétně ve verzi na stránce `http://pruvodce.ucw.cz/static/pruvodce.pdf`.

1. Bylo by fajn mít k dispozici větší abecedu než $\{0, 1\}$, třeba $\{0, 1, \clubsuit\}$, kde $\clubsuit$ znamená „tady končí číslo“. Obecně můžete abecedu s 2^k znaky zapsat pomocí $\{0, 1\}$ tak, že každému znaku původní abecedy bude odpovídat k znaků té binární. Zlepšit to pak můžete tím, že nejdřív zapíšete pomocí téhle myšlenky počet bitů čísla a potom to samotné číslo už jen základní abecedou.
2. str. 435
3. str. 436
4. str. 437
5. str. 438
6. str. 438
7. Vyrobte si bipartitní graf, kde vlevo jsou proměnné, vpravo klauzule a použijte Hallovu větu.
8. To dáte ;).
9. Viz obrázek v Průvodci na straně 445
10. Hladově. Začněte v nějakém vrcholu, řekněte, že je černý. Všichni jeho sousedi potom musí být bílí atd.
11. To, že umíte najít polynomiální algoritmus, který ověří řešení, je jednoduché. Představte si, že máte čísla, řekněme, v desítkové soustavě a že umíte sčítat bez přenosu (jak se řekne česky carry?). Ze zadané 3-SAT formule potom budete chtít vyrobit nějakou množinu čísel, která budete spíš chápat jako dlouhé vektory číslic. Každé proměnné i každé klauzuli přiřadíte jednu pozici a každému literálu (tj. proměnné či její negaci) přiřadíte vektor, který bude mít jedničku na pozici příslušící dané proměnné a potom jedničky na pozicích všech klauzulí, které by daný literál splnil. Číslo, které chcete sečíst, bude mít na všech pozicích odpovídajících proměnným jedničky (tj. vyberete právě jednu pravdivostní hodnotu) a na pozicích pro klauzule nějaká větší čísla, třeba devítky (chcete, aby to nebyla nula, tj. aby ta klauzule byla něčím splněná, a chcete určitou variabilitu v tom, jestli klauzuli splní jeden, dva, nebo tři literály). Potom ještě přidáte nějaký balast navíc, který se dá přičíst, aby člověk z klauzule splněné nějakým počtem literálů uměl vždycky udělat tu devítku. Teď už si stačí rozmyslet, jak vyřešit, aby se nesčítalo s přenosem (třeba mezi důležité pozice naházet dostatek nul).
12. To, že je v NP je lehké, převést na batoh vlastně taky (stačí říct, že hodnoty jsou rovné počtům slov a položit $W = V = S$).