Viggo Kann 2021-08-26

DD2350 Algoritmer, datastrukturer och komplexitet

Uppgifter till övning 12

Komplexitetsklasser och repetition

Uppgifter på komplexitetsklasser

co-NP-fullständighet Ett diskret tekniskt diagnosproblem kan modelleras på följande sätt: komponenttillstånd, systemtillstånd och omgivningstillstånd representeras av booleska variabler och själva systemet definieras med en boolesk formel φ. Man vet att i alla *möjliga* (fungerande) världar kommer systemvariablerna att ha värden så att φ är sann.

Anta att komponenttillstånden alla är tvåvärda och att en komponent c därför representeras av en boolesk variabel x_c (där sant betyder att komponenten fungerar och falskt att den är trasig). Vi vet att en komponent c är trasig om den formel som vi får om vi i φ sätter in värden för alla variabler som representerar kända komponent-, system- och omgivningstillstånd samt sätter x_c till sann inte är satisfierbar.

Visa att det är co-NP-fullständigt att avgöra ifall c är trasig.

Komplexitetsklassrelation PSPACE är komplexitetsklassen som består av alla språk för vilka det existerar en *deterministisk* turingmaskin som känner igen språket i polynomiskt *minne*. EXPTIME består av alla språk för vilka det existerar en *deterministisk* turingmaskin som känner igen språket i exponentiell tid. Visa att PSPACE$\subseteq$EXPTIME.

Repetition inför teoritentan

Följande uppgift är en typisk uppgift på teoritentan.

Avgör korrekthet för påståenden Är följande påståenden sanna eller falska? För varje deluppgift ger riktigt svar 1 poäng och ett övertygande bevisat riktigt svar 2 poäng,

a) Problemet att avgöra ifall ett tal med n siffror är ett primtal ligger i komplexitetsklassen co-NP.

b) Det finns en konstant $c > 1$ så att $n^3 \in O(c^{\log n})$.

c) Binära träd implementerar man vanligen genom att införa två pekare i varje post (`left` och `right`). När man implementerar ternära träd (där varje nod har tre söner) går det inte att klara sig med mindre än tre pekare i varje post.

Gammal teoritenta Gå igenom en gammal teoritenta.

Tips! Alla gamla tentor finns på webbsidan
https://www.kth.se/social/course/DD2350/page/tidigare-teoritentor/
I kursrummet finns också övningsquiz där du kan öva själv på formulering av definitioner och motiveringar inför teoritentan.

Lösningar

Lösning till co-NP-fullständighet
För att visa att problemet ligger i co-NP så ska vi visa att vi i polynomisk tid kan verifiera en nej-lösning, det vill säga att c inte är trasig. Vi vet att c inte är trasig om och endast om det finns en variabeltilldelning som satisfierar formeln. Om vi gissar variabelvärden behöver vi alltså bara verifiera att formeln med denna variabeltilldelning är sann. Detta går att göra i polynomisk tid.

För att visa att problemet är co-NP-svårt reducerar vi co-SAT, alltså problemet att avgöra ifall en given boolesk formel ϕ *inte* är satisfierbar. Eftersom SAT är NP-fullständigt så är per definition co-SAT co-NP-fullständigt.

Givet ϕ, konstruera ett system som innehåller den extra komponenten c (representerad av x_c) och som definieras av formeln $\varphi = \phi \vee \neg x_c$.

Notera nu att om x_c sätts till sann blir $\varphi = \phi$ så problemet att avgöra ifall c är trasig är precis samma som problemet att avgöra ifall ϕ inte är satisfierbar. Alltså är reduktionen korrekt. □

Lösning till Komplexitetsklassrelation
Om en turingmaskin använder polynomiskt minne finns det en konstant k så att antalet använda rutor på bandet är $O(n^k)$ där n är indatas längd. Om alfabetet består av tre tecken (0, 1, blank) så är antalet olika möjliga konfigurationer på bandet begränsat av $3^{O(n^k)}$, antalet möjliga platser för läs/skrivhuvudet är $O(n^k)$ och antalet möjliga tillstånd i turingmaskinen är ändligt, dvs $O(1)$. Det totala antalet konfigurationer för en turingmaskin som använder polynomiskt minne är alltså $O(n^k) \cdot 3^{O(n^k)}$, dvs exponentiellt i n. Eftersom turingmaskinen inte kan återkomma till samma konfiguration flera gånger (då skulle den gå i en oändlig slinga) så är detta också en övre gräns på tiden. Alltså kan varje problem som kan lösas med polynomiskt minne lösas i exponentiell tid. □

Lösning till Avgör korrekthet för påståenden

a) *Sant.* Problemet ligger i co-NP om komplementproblemet ligger i NP. Komplementproblemet är i detta fall att avgöra ifall ett tal med n siffror kan faktoriseras i minst två faktorer (större än 1). Detta problem ligger i NP eftersom en lösning (dvs en faktorisering av talet) kan verifieras i polynomisk tid (genom att man multiplicerar ihop faktorerna och kollar att produkten blir det givna talet).

b) *Sant.* Om vi antar att $\log n$ är logaritmen i basen 2 så vet vi att $c^{\log n} = 2^{\log c^{\log n}} = 2^{\log n \log c} = n^{\log c}$. Om vi väljer $c \geq 8$ så är $\log c \geq 3$ och $n^3 \in O(n^{\log c}) = O(c^{\log n})$.

c) *Falskt.* För allmänna träd räcker det med två pekare i varje post (`firstchild` och `next`).

□
