Viggo Kann, KTH CSC

Lösningar till teoritenta i Algoritmer, datastrukturer och komplexitet för KTH DD1352/2D1352/DD2352/DD2354/2D1354 och SU 2012-12-13

1. (8 p) Är följande påståenden sanna eller falska? För varje deluppgift ger riktigt svar 1 poäng och ett övertygande motiverat riktigt svar 2 poäng.

   a/b) $2^{3\log_2 n} \in \Omega((3n)^2)$.
   Sant. $2^{3\log_2 n} = (2^{\log_2 n})^3 = n^3 \in \Omega(n^2) = \Omega((3n)^2)$.

   a/b) Insättning i en Skipplista är ett exempel på en Monte Carlo-algoritm.
   Falskt. En Monte Carlo-algoritm har garanterad körtid men ger bara rätt svar ibland. Insättning i en Skipplista tar olika lång tid, beroende på hur många nivåer elementen ligger i, men fungerar alltid.

   c/d) Det som skiljer en approximationsalgoritm och en heuristik är att heuristiken säkert returnerar en lösning som är nära den optimala.
   Falskt. Det är tvärtom approximationsalgoritmen som ger en lösning som är garanterat en viss faktor från den optimala medan heuristiken inte ger såna garantier.

   c/d) Memoisering är en metod att lagra data i flödesgrafer.
   Falskt. Memoisering är ett sätt att implementera en dynprogrekursion där funktionen först kollar om den anropats med samma parametrar förut och i så fall slår upp funktionsvärdet i en tabell.

2. (3 p) A, B, C, D och E är beslutsproblem. Anta att B är NP-fullständigt och att det finns polynomiska Karpreduktioner mellan problemen så här:

   $$\begin{array}{ccccc} \mathrm{A} & \leftarrow & \mathrm{B} & \leftarrow & \mathrm{C} \\ & & \downarrow & & \uparrow \\ & & \mathrm{D} & \rightarrow & \mathrm{E} \end{array}$$

   Vad vet man då om komplexiteten för A, C, D och E?

	ligger i NP	är NP-fullständigt	är NP-svårt
A			X
C	X	X	X
D	X	X	X
E	X	X	X

3. (3 p) *Kakelläggarens problem* är ett känt oavgörbart problem där indata består av ett antal olika typer av kvadratiska kakelplattor som inte kan vridas (dvs en specifik sida måste vara nedåt). Frågan är om det för varje positivt heltal m är möjligt att kakla en kvadratisk vägg med $m \times m$ plattors storlek med kakelplattor av dessa typer, så att mönstret stämmer i alla skarvar.

   Visa hur man i *ändlig* tid kan verifiera att en probleminstans är en *nej*-instans givet en lösning. Tala tydligt om vad lösningen består av.

   Om svaret är nej så finns det ett värde på m för vilket det inte går att kakla en $m \times m$-vägg. Låt lösningen vara detta tal m.

   Verifikationen blir att testa alla tänkbara kaklingar av väggen. Om det finns n typer av kakelplattor finns det $n^{(m^2)}$ tänkbara kaklingar. Att kolla mönstret i en kakling tar $O(m^2)$, så en totalsökningsalgoritm tar $O(m^2 n^{(m^2)})$, vilket är ändligt.

4. (6 p) Vi söker i denna uppgift en polynomisk heuristik som ger en så bra lösning till MAX CNFSAT (givet en formel i konjunktiv normalform, satisfiera maximalt antal klausuler) som möjligt. Algoritmen ska också ge ett så bra mått som möjligt på hur långt bort från den optimala lösningen som den hittade lösningen som mest är, uttryckt i procent. Till ditt förfogande finns följande färdiga polynomiska algoritmer:

$\langle \boldsymbol{x}, s\rangle = \text{AsanoApprox}(\phi)$	Approximationsalgoritm med approximationskvot 1,3.
$\langle \boldsymbol{x}, s\rangle = \text{RandomSAT}(\phi)$	Probabilistisk heuristik.
$\langle \boldsymbol{x}_2, s\rangle = \text{LocalSearchOpt}(\phi, \boldsymbol{x}_1)$	Deterministisk heuristik som försöker ge en förbättring av lösningen $\boldsymbol{x}_1$ med hjälp av lokal sökning.

där $\boldsymbol{x}$ är en variabeltilldelning, s är antalet satisfierade klausuler och ϕ är en CNF-formel.

Din algoritm ska ta ϕ som indata och returnera trippeln $\langle \boldsymbol{x}, s, g\rangle$ som utdata, där g anger hur många procent algoritmens lösning som mest är från den optimala.

Om s_A är antalet satisfierade klausuler som AsanoApprox algoritm ger så vet vi att $\text{OPT}/s_A \leq 1,3$. Det betyder att om vi lyckas hitta en ännu bättre lösning med värdet s så gäller $\text{OPT}/s = (s_A/s) \cdot \text{OPT}/s_A \leq 1,3 \cdot (s_A/s)$. Avståndet till optimala värdet uttryckt i procent är $100 \cdot (1 - (s/\text{OPT})) \leq 100 \cdot (1 - s/(1,3s_A))$.

$\langle \boldsymbol{x}_A, s_A\rangle = \text{AsanoApprox}(\phi)$
$\langle \boldsymbol{x}, s\rangle = \text{LocalSearchOpt}(\phi, \boldsymbol{x}_A)$
for $i \leftarrow 1$ **to** 100 **do**
    $\langle \boldsymbol{x}_R, s_R\rangle = \text{RandomSAT}(\phi)$
    $\langle \boldsymbol{x}_t, s_t\rangle = \text{LocalSearchOpt}(\phi, \boldsymbol{x}_R)$
    if $s_t > s$ **then** $\boldsymbol{x} \leftarrow \boldsymbol{x}_t; s \leftarrow s_t$
$g \leftarrow 100 \cdot (1 - s/(1, 3 \cdot s_A))$
return $\langle \boldsymbol{x}, s, g\rangle$

Algoritmen går i polynomisk tid eftersom den bara gör ett konstant antal anrop av hjälpfunktioner som går i polynomisk tid.