Viggo Kann, KTH CSC

Lösningar till teoritenta i Algoritmer (datastrukturer) och komplexitet för KTH DD1352/2D1352/DD2354/2D1354 och SU 2011-12-19

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) Det är lämpligt att använda räknesortering (counting sort) om man ska sortera n objekt i m olika kategorier.

   Sant. Räknesortering tar som indata en array med element och en kategoriseringsfunktion f : element $\rightarrow$ kategori. Den sorterar sedan elementen i dessa kategorier.

   b) I en oriktad graf med n hörn och m kanter gäller alltid att $m \in O(n \log n)$.

   Falskt. I en komplett graf finns $n(n-1)/2 = \Theta(n^2)$ kanter, vilket inte ryms i $O(n \log n)$.

   c) Ett Bloomfilter är till hälften fyllt med ettor. Vid uppslagning i det används tio stycken oberoende och jämnt spridande hashfunktioner. Sannolikheten för att en uppslagning av ett element som inte är inlagt i Bloomfiltret ska ge fel svar (alltså att elementet finns med) är omkring 0,1 %.

   Sant. För att Bloomfiltret ska godkänna ett element krävs att alla tio uppslagna bitar är ett. Sannolikheten för detta är $(1/2)^{10} = 1/1024 \approx 0,001$.

   d) Ford-Fulkersons metod är en dekompositionsalgoritm.

   Falskt. En dekompositionsalgoritm delar upp problemet rekursivt i delproblem och löser dessa var för sig. Ford-Fulkerson utökar iterativt ett flöde med en stig i taget i restflödesgrafen från källan till utloppet. Det är inte dekomposition.

2. (2 p) Nämn fyra realistiska metoder som kan användas för att angripa ett optimeringsproblem som visats vara NP-svårt.

   Konstruera en approximationsalgoritm, konstruera en eller helst flera heuristiker, lös problemet för små indata med en totalsökningsalgoritm, förenkla problemet.

3. (4 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 (en reduktion av A till B tecknas här $A \rightarrow B$):

   $$\begin{array}{ccccccc} \mathrm{A} & \rightarrow & \mathrm{B} & \leftrightarrow & \mathrm{C} & \rightarrow & \mathrm{D} \\ & & \downarrow & & & & \\ & & \mathrm{E} & & & & \end{array}$$

   Vad vet man då om komplexiteten för A, C, D och E? Sätt kryss i tabellen nedan för allt man säkert vet.

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

4. (6 p) I *Kappsäcksproblemet* är indata ett tal W och n prylar, där pryl i har vikten w_i och värdet v_i. Problemet är att bestämma vilka prylar som ska packas ner i kappsäcken så att summan av deras vikt är högst W och summan av deras värde är maximal.

   Vi söker i denna uppgift en heuristik som försöker hitta en bra lösning till Kappsäcksproblemet, ju större sammanlagt värde desto bättre. Heuristiken ska använda sig av *lokal sökning*. Beskriv din algoritm med pseudokod på relativt hög nivå.

```
Extend(K, W, E) = // utvidgar kappsäcken K med saker i E till viktgränsen W
    sw ← 0; sv ← 0
    foreach (w, v) ∈ K do sw ← sw + w; sv ← sv + v
    foreach (w, v) ∈ E do // behandla elementen i E i slumpvis ordning
        if s + w ≤ W then
            K ← K ∪ {(w, v)}; sw ← sw + w; sv ← sv + v
    return (K, sv)

KnapsackLocalSearch({(w_i, v_i)}_1^n, W) =
    (K, s) ← Extend(∅, W, {(w_i, v_i)}_1^n)
    R ← {(w_i, v_i)}_1^n − K // resterande element
    oldval ← 0
    while s > oldval do
            foreach (w, v) ∈ K do // prova att byta ut ett element i kappsäcken
                (newK, news) ← Extend(K − {(w, v)}, W, R)
                if news > s then // bättre lösning funnen
                    K ← newK; s ← news
                    R ← {(w_i, v_i)}_1^n − K
    return (K, s)
```