

### 35.1-1

APPROX-VERTEX-COVER vil aldrig give en optimal løsning til en graf med tre knuder og to kanter.

### 35.1-2

En maksimal matching er en matching, der ikke kan udvides. Hvis mængden af kanter,  $A$ , ikke er en sådan, vil det være i modstrid med, at  $E$  er tom.

### 35.1-4

Vi skal finde en optimal knude-overdækning på et træ i lineær tid: Kanter til blade vil kunne dækkes af både bladene og bladenes forældre; da bladenes forældre også dækker kanterne til bedste-forældrene kan det naturligvis bedst betale sig at vælge dem. Dette gentages indtil alle kanter er overdækket.

### 35.1-5

Antag, at vi for en graf med  $V$  knuder kan finde en approksimations-algoritme med konstant approksimations-faktor,  $C$ , til knude-overdæknings-problemet (VC). Vi skal undersøge, om vi — under hensyn til sammenhængen mellem de optimale løsninger af de to problemer — kan udnytte dette til at finde noget tilsvarende for klike-problemet (Q): Sammenhængen mellem antallet af knuder i de optimale løsninger af de to problemer er som bekendt  $\text{OPT}_{\text{VC}} = V - \text{OPT}_{\text{Q}}$ . Heraf følger

$$C \text{OPT}_{\text{VC}} = C(V - \text{OPT}_{\text{Q}}) \leq \tilde{C} \text{OPT}_{\text{Q}}$$

da  $\tilde{C}$  skal være konstant.

### 35-1

a)

Vi skal se, at optimeringsversionen af “bin-packing”-problemet er NP-hårdt; det kommer ud på, at det tilhørende afgørighedsproblem er NP-fuldstændigt. Vi interesserer os derfor for, om vi kan pakke i  $\leq k$  spande (“bins”).

$$\text{BIN-PACKING} = \left\{ (C, k) \mid \begin{array}{l} C = \{c_1, \dots, c_n\} \\ 0 < c_i < 1 \\ \hline C \text{ kan fordeles i } \leq k \text{ spande} \end{array} \right\}$$

1. BIN-PACKING  $\in$  NP: Et certifikat i form af indholdet af de forskellige spande vil kunne kontrolleres i lineær tid i antallet af elementer i  $C$ .
2. Vi forsøger at reducere fra SUBSET-SUM  $\in$  NPC:

$$\forall L \in \text{NP} : L \leq_{\text{pol}} \text{SUBSET-SUM} \leq_{\text{pol}} \text{BIN-PACKING}$$

3. For en given instans SUBSET-SUM( $C, t$ ) konstruerer vi først elementet  $c_{n+1} = 2t - \sum_{c \in C} c$  og mængden  $C' = C \cup \{c_{n+1}\}$  (hvis  $t > \frac{1}{2} \sum_{c \in C} c$  bliver  $c_{n+1}$  negativ, men så spørger vi i stedet efter  $t = \sum_{c \in C} c - t$ ). Herefter skales alle elementer i  $C'$  ned med  $t$ :  $C'' = \{\frac{c}{t} \mid c \in C'\}$ . Afslutningsvis betragtes instansen BIN-PACKING( $C'', 2$ ).
4. Antag, at der i  $C$  findes en delmængde  $A$ , så  $\sum_{c \in A} c = t$ . Summen af de resterende elementer i  $C''$  vil dermed være givet ved

$$\sum_{c \in C'' \setminus A} \frac{c}{t} = \sum_{c \in C'} \frac{c}{t} - \sum_{c \in A} \frac{c}{t} = 2 - 1 = 1$$

hvoraf det fremgår, at elementerne i  $C''$  kan pakkes i netop to spande.

Uanset hvilken opdeling af  $C''$  i to (fulde) spande, der betragtes, vil  $\frac{c_{n+1}}{t}$  kun kunne ligge i den ene spand. Den anden spand vil dermed udelukkende bestå af oprindelige, nedskalerede elementer fra  $C$ , hvis oprindelige sum tilsammen giver  $t$ .

5. Transformationen af en instans af SUBSET-SUM til en instans af BIN-PACKING består hovedsagelig i konstruktion af elementet  $c_{n+1}$ . Dette kan gøres i lineær tid i antallet af elementer i  $C$ , dvs.  $O(n)$ .

**b)**

Lad nu  $S = \sum_{i=1}^n c_i$ ; der er oplagt brug for mindst  $\lceil S \rceil$  spande, da kapaciteten af en spand er 1.

**c)**

“First-fit”-heuristikken efterlader højst én spand mindre end halvt fuld; flere af sådanne spande ville nemlig kunne samles sammen, hvilket ville være i modstrid med “first-fit”-princippet.

**d)**

Vi skal se, at “first-fit”-heuristikken aldrig bruger mere end  $\lceil 2S \rceil$  spande. Det følger imidlertid direkte af spørgsmål c), idet de alle spande på nær én *altid* vil være *mere end halvt fulde*.

**e)**

Vi skal se, at “first-fit”-heuristikken har en approksimations-faktor på to. Lad  $S^*$  betegne antallet af spande i en optimal løsning, og  $S_F$  løsningen fra heuristikken. Da der er tale om et minimerings-problem, skal vi konkludere, at  $S_F \leq 2S^*$ .

Vi ved fra b), at  $\lceil S \rceil \leq S^*$ . Fra d) følger nu

$$S_F \leq \lceil 2S \rceil = 2\lceil S \rceil \leq 2S^*$$

**f)**

I implementationen skal vi holde styr på, hvor meget plads, der er tilbage i de forskellige spande. Dette kan med fordel gøres ved hjælp af en max-hob: Hvis størrelsen af det aktuelle element er større end den største rest-plads (FIND-MAX) vil der heller ikke være plads i nogen af de andre spande. Elementet lægges derfor i en ny spand (og der tilføjes et nyt element i hoben). Hvis der er plads til elementet reduceres kapaciteten af den aktuelle spand med størrelsen af elementet. Er en spand fuld, kan den tilsvarende knude tages ud af hoben. Køretiden bliver  $O(n \lg n)$ .