**Algoritmer och komplexitet**
**Hösten 2009**
**Mästarprov 2: Komplexitet**

Mästarprovet ska lösas **individuellt** och redovisas både skriftligt och muntligt. Inget samarbete är tillåtet, se vidare hederskodexen.

**Skriftliga lösningar** ska lämnas senast **måndag 7 dec klockan 18.00** i mitt postfack eller i kursens inlämningslåda på studentexpeditionen på Osquars backe 2 plan 2. Det är viktigt att du lämnar in i tid! Se till att spara en kopia av dina lösningar så att du kan läsa på inför den **muntliga redovisningen** som kommer att ske dagarna efter inlämningen.

Mästarprovet är ett obligatoriskt och betygsatt moment i kursen. Det består av fyra uppgifter med olika svårighetsgrad. För betyg E krävs rätt på två av uppgifterna. Sedan är betygssättningen i stort sådan att helt rätt på tre uppgifter ger betyg C och helt rätt på alla uppgifter ger betyg A.

Läs mer om betygskriterier och slutbetyg i kurs-PM eller på kursens webbsida.

## 1. Lösning till PARTITIONERING

Problemet PARTITIONERING är som vi sett på en föreläsning NP -fullständigt. Man kan t.ex. visa det genom att reducera från DELSUMMA. Antag nu att vi ändå vill försöka avgöra problemet. Låt indata vara $x_1, x_2, ..., x_n$. Konstruera en dynamisk programmerings-algoritm som avgör om summan av $x_1, x_2, ..., x_n$ kan partitioneras d.v.s. delas upp i två delsummor av samma storlek. Gör en komplexitetsanalys av din algoritm och visa att den inte är polynomiell i indatas storlek. (Kommer du fram till att den är polynomiell bör du nog tänka efter en gång till.)

## 2. Ett grafspel

Vi tänker oss ett grafspel som går till på följande sätt: Vi har en oriktad graf $G$ men noder $v_1, v_2, ..., v_n$. I varje nod $v_i$ finns ett tal $p_i$ placerat. Vi skall nu vandra runt i grafen från nod till nod genom att gå över kanterna. En sådan väg runt i grafen får använda både noder och kanter flera gånger. Under vandringen runt kommer en poängsumma $S$ hela tiden att beräknas. Vi startar i någon nod $v_i$ och får då $S = p_i$. När vi sedan besöker en nod $v_j$ *första* gången adderas $p_j$ till $S$. Om vi besöker noden igen adderas inget till $S$. $S$ blir alltså summan av $p$-talen för alla noder man besöker. Målet för spelet är att få så hög summa som möjligt med så kort väg (så litet antal kanter) som möjligt. Vi kan ställa upp följande beslutsproblem:

Indata: En oriktad graf $G$. Reella tal $p_1, ..., p_n$ där $n$ är antalet noder i $G$. Ett positivt heltal $M$. Ett reellt tal $K$.

Mål: Finns det en väg $w$ av längd $\leq M$ (antal kanter) som startar i någon nod och som ger en summa $S \geq K$ om vi spelar grafspelet i $G$ med talet $p$-talen placerade i motsvarande noder?

Visa att detta problem är NP-fullständigt.

### 3. Ett konstruktionsproblem

Från föreläsninganteckningarna vet vi att följande variant av 0/1-Programmering som vi kan kalla 0/1-LIN-SAT är NP-fullständigt:

Indata: En uppsättning ekvationer

$$A\bar{x} \leq \bar{b}$$

där $\bar{x} = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix}$ och $A, \bar{b}$ är är en matris och en vektor med reella tal.

Mål: Finns det en vektor $\bar{x}$ med alla element 0 eller 1 som uppfyller olikheterna?

Vi kan formulera en optimeringsvarianten av problemet genom att ändra målet till att finna det maximala antalet olikheter som kan uppfyllas samtidigt. (Varje olikhet motsvarar en rad i $A$-matrisen.) Låt oss anta att vi har en algoritm $F(A, \bar{b})$ som returnerar det maximala antalet olikheter som kan uppfyllas samtidigt under förutsättning att alla $x_i$ är 0 eller 1.
Konstruera en algoritm som hittar en variabeltilldelning för $\bar{x}$ som är en lösning till optimeringsproblemet. Algoritmen skall använda sig av $F$ men får högst göra ett polynomiellt antal anrop. Antag att $F$ har tidskomplexitet $T(m, n)$ där $m, n$ är antalet rader och kolumner i $A$. Ange din algoritms komplexitet som funktion av $m, n$ och $T$.

### 4. Brutopolis tunnelbana

Staden Brutopolis, huvudstaden i Brutopia, har ett enormt tunnelbanesystem. Regeringen i Brutopia vill kunna polisövervaka hela tunnelbanan. Helst skulle man vilja ha en liten polisstation på varje tunnelbanestation men man inser att det skulle bli för dyrt. Därför nöjer man sig med att välja ut vissa stationer till s.k. övervakande stationer. Dessa skall vara valda så att varje tunnelbanestation antingen är en övervakande station eller granne med en övervakande station. (Med granne menar vi att stationerna ligger direkt efter varandra på en linje.) Brutopolis tunnelbanesystem kan tolkas som en godtycklig graf. Brutopias säkerhetspolis ställs inför ett optimeringsproblem: Indata är en graf $G$ som beskriver Brutopolis tunnelbanesystem. Målet är att bestämma det minimala antalet övervakande stationer som krävs. Formulera motsvarande beslutsproblem och avgör om det är NP-fullständigt eller inte.