Föreläsning 9: NP-fullständighet

Olika typer av problem:
1. Beslutsproblem: $A(x) =$ Ja.
2. Optimeringsproblem: $A(x) = m$ Vanligen max/min.
3. Konstruktionsproblem: $A(x) =$ En struktur. Vanligen "lösningen" till problemet.

Några reduktioner:
Lösningen till beslutsproblem ger ofta lösningen till motsvarande optimerings- och konstruktionsproblem.
Ex:
KLICK

Indata: Graf G och ett tal K.
Problem: Finns det en **klick** av storlek K i G?
(Klick = Komplett delgraf.)

KLICK är ett svårt (**NP**-fullständigt) problem.

Motsvarande optimeringsproblem:
MAX-KLICK

Indata: Graf G. Problem: Vad är storleken på en maximal klick i G?

Motsvarande konstruktionsproblem:
KONSTRUERA-MAX-KLICK

Indata: Graf G. Problem: Hitta en klick av maximal storlek i G.

Antag att KLICK har en lösningsalgoritm $A(G, k)$ så att $A(G, k) =$ Ja om och endast om G har en klick av storlek k.
MAX-KLICK har lösningsalgoritmen $B(G)$:

```
(1)   for k ← to n
(2)       if A(G, k) = Ja
(3)           m ← k
(4)   return m
```

KONSTRUERA-MAX-KLICK har lösningsalgoritm $C(G)$:

```
(1)   m ← B(G)
(2)   S ← V
(3)   foreach v ∈ V
(4)       if B(G(S − {v})) = m
(5)           S ← S − {v}
(6)       return S
```

Turingreduktioner

En turingreduktion $Q \leq_T Q'$ är en algoritm som löser Q med hjälp av anrop till en algoritm för Q':

```
Q(x)
(1)     ⋮
(2)     Q'(z)
(3)     ⋮
(4)     return svar
```

Turingreduktionen är liberalare än karpreduktionen; Q' behöver t.ex. inte vara ett beslutsproblem.
Om Q är **NP**-fullständigt och $Q \leq_T Q'$ säger vi ändå att Q' är **NP**-svårt.

Några reduktioner

Två problem: OBEROENDE MÄNGD, HÖRNTÄCKNING
Oberoende mängd: Mängd noder utan kanter mellan dem.
Hörntäckning: Urval av noder som täcker in varje kant.

OBEROENDE MÄNGD
Indata: Graf G, tal K. Problem: Finns det oberoende mängd av storlek K?

HÖRNTÄCKNING
Indata: Graf G, tal K. Problem: Finns hörntäckning av storlek K?

Observation: Om $A \subseteq V$ är en oberoende mängd så är $V - A$ en hörntäckning och omvänt.

Givet instans $x = G, K$ till OBEROENDE MÄNGD skapas $R(x) = G, |V| - K$ till HÖRNTÄCKNING. Detta ger en reduktion OBEROENDE MÄNGD $\leq_P$ HÖRNTÄCKNING

Två andra problem: HANDELSRESANDEPROBLEMET(TSP), HAMILTONKRETS

TSP

Indata: Viktad komplett graf G och ett tal K. Problem: Finns det Hamiltonkrets med vikt $\leq K$?
(Hamiltonkrets = Krets som besöker varje nod exakt en gång.)

HAMILTONKRETS

Indata: Graf G. Problem: Finns det en Hamiltonkrets?

Givet indata $x = G$ till HK skapas komplett graf G' med $w(e) = 1$ om $e \in G$ och $w(e) = 0$ om $e \notin G$. Sätt sedan $K = |V|$. Det ger en korrekt reduktion.

Problem: För att visa att A är **NP**-fullständigt måste man visa att $X \leq_P A$ för alla $X \in$ **NP**. Hur är det möjligt?

Första NP-fullständiga problemet

SAT är följande problem:
Givet en logisk formel Φ, finns det någon variabeltilldelning som gör den sann?
Sats (Cook 1971). *SAT är **NP**-fullständigt.*
(I samma artikel som klassen **NP** definierades för första gången bevisades denna sats.)

Bevisskiss: Varje **NP**-problem motsvarar en icke-deterministisk algoritm A. Problemet är att avgöra om, givet indata x, det finns en beräkning som ger $A(x) = \text{Ja}$. Cook visade att man kan konstruera en logisk formel F så att $A(x)$ har accepterande beräkning $\Leftrightarrow$ F är satisfierbar.

För att visa att A är **NP**-fullständigt räcker det att visa att $SAT \leq_P A$.
Varför: Om $X \in \mathbf{NP}$ gäller $X \leq SAT$. Om $SAT \leq A$ också gäller får vi $X \leq A$!
Praktiskt tillvägagångssätt: Bilda riktad graf där $A \to B$ betyder $A \leq B$. $SAT \to A \to B \to C \to ...$ betyder då att $A, B, C, ...$ är **NP**-fullständiga.

För att visa att A är **NP**-fullständigt räcker det att hitta ett **NP**-fullständigt problem B så att $B \leq A$.
Några varianter på SAT:

SAT:
Formeln får innehålla operatorerna
$\wedge, \vee, \neg, \Rightarrow, \equiv$.

CNF-SAT:
Formeln måste vara på *konjunktiv normalform* d.v.s. bestå av $\wedge$ över ett antal klausuler som bara innehåller $\vee$ och $\neg$.

***k*-CNF-SAT:**
Formeln måste vara på CNF-SAT-form med exakt k literaler i varje klausul.

Om NP-fullständighetsbevis

Alla **NP**-fullständiga problem kan reduceras till varandra, men om man ska visa att ett problem är **NP**-fullständigt kan det ändå löna sig att utgå från rätt problem.
I praktiken finns det ofta enkla reduktioner från något av 5–10 kanoniska **NP**-fullständiga problem.
Det enskilda vanligaste problemet att utgå från är 3-CNF-SAT.

CNF-SAT är NP-fullständigt

1. CNF-SAT $\in$ **NP** ty en lösning kan verifieras genom att kolla att varje klausul innehåller någon satisfierad literal.

2. Att CNF-SAT är **NP**-svårt ser man ur beviset för Cooks sats — den SAT-formel som konstrueras kan lätt skrivas som en CNF-SAT-formel.

   Alternativt kan man ge en reduktion
   SAT $\leq_P$ CNF-SAT.

$\Rightarrow$ CNF-SAT är **NP**-fullständigt

3-CNF-SAT är NP-fullständigt

Att 3-CNF-SAT $\in$ **NP** är klart.
Vi ska reducera CNF-SAT till 3-CNF-SAT:
Givet en CNF-SAT-formel $\Phi = c_1 \wedge \cdots \wedge c_k$ bildar vi en ekvivalent 3-CNF-SAT-formel Φ_3 genom att ersätta varje klausul i Φ med en eller flera 3-CNF-SAT-klausuler.
Antag att c_i innehåller j literaler $l_1, \ldots l_j$. Klausuler som läggs till Φ_3:

$j = 3$ $\quad l_1 \vee l_2 \vee l_3$

$j = 2$ $\quad (l_1 \vee l_2 \vee y_i) \wedge (l_1 \vee l_2 \vee \neg y_i)$

$j = 1$ $\quad (l_1 \vee y_i \vee z_i) \wedge (l_1 \vee y_i \vee \neg z_i) \wedge$
$\quad (l_1 \vee \neg y_i \vee z_i) \wedge (l_1 \vee \neg y_i \vee \neg z_i)$

$j > 3$ $\quad (l_1 \vee l_2 \vee y_i^1) \wedge (\neg y_i^1 \vee l_3 \vee y_i^2) \wedge$
$\quad (\neg y_i^2 \vee l_4 \vee y_i^3) \wedge \cdots \wedge (\neg y_i^{j-3} \vee l_{j-1} \vee l_j)$

Φ_3 är per konstruktion satisfierbar precis då Φ är satisfierbar.

Några NP-fullständiga grafproblem

Hamilton cycle
Har grafen G en hamiltoncykel?
(D.v.s. en sluten stig som passerar alla hörn exakt en gång.)

Clique
Innehåller grafen G en klick av storlek K?
(En klick är en komplett delgraf.)

Vertex cover
Givet en graf G och ett tal K, går det att välja ut K hörn i G så att alla kanter innehåller minst ett av de K hörnen?

Graph coloring
Givet en graf G och ett heltal K, går det att färga hörnen i G med K färger så att ingen kant blir monokromatisk?

Andra NP-fullständiga problem

Exact cover
Givet en uppsättning delmängder av en basmängd M. Går det att välja ut en mängd av delmängder så att varje element i M ligger i exakt en mängd?

Subset sum
Givet en mängd P av positiva tal och ett tal K. Finns det någon delmängd av talen i P vars summa är K?

Integer programming
Givet en $m \times n$-matris A, en m-vektor b, en n-vektor c och ett tal K. Finns det en n-vektor x med heltal så att $Ax \leq b$ och $c \cdot x \geq K$?

Om man i sista exemplet släpper kravet att komponenterna i x är heltal blir problemet (linjärprogrammering) polynomiskt lösbart.

Ytterligare NP-fullständiga problem

Bin packing
: Givet en mängd vikter och k lådor som var och en klarar högst vikt B. Går det att lagra vikterna i lådorna?

Knapsack
: Givet en mängd S med element som har vikter w och värde v definierade. Går det att hitta en delmängd S' så att summan av vikterna $\leq W$ och summan av värdena $\geq B$?

TSP
: Handelsresandeproblemet.

Reduktioner för de flesta av dessa problem finns beskrivna CLR 36.4-36.5.

Delgrafsisomorfi är NP-fullständigt

Givet två oriktade grafer G_1 och G_2, är G_1 en delgraf till G_2?
Problemet tillhör uppenbarligen **NP**.
Bevis för att problemet är **NP**-svårt:
Reducera från Hamilton Cycle.
En graf $G = (V, E)$ innehåller nämligen en hamiltoncykel om och endast om den innehåller en delgraf som är isomorf med cykeln $C_{|V|}$ som har lika många noder som G.

Partition är NP-fullständigt

Givet en mängd S av positiva tal.
Kan S delas upp i två disjunkta delar S_1 och S_2 så att $\sum_{x \in S_1} x = \sum_{x \in S_2} x$?
Problemet ligger i **NP** ty givet en disjunkt uppdelning går det snabbt att kolla om den uppfyller villkoren.
Vi reducerar från Subset sum:
En instans där är tal $p_1, p_2, \ldots, p_n$ och K; ur instansen skapar vi partitionsinstansen

$$p_1, p_2, \ldots p_n, P - 2K$$

(där $P = \sum p_i$) om $K \leq P/2$ och

$$p_1, p_2, \ldots p_n, 2K - P$$

annars. En jämn partitionering av denna instans finns precis då Subset sum-instansen är lösbar.

0/1-programmering är NP-fullständigt

Givet en $m \times n$-matris A och en m-vektor b.
Finns det en n-vektor x med heltal $\in \{0, 1\}$ så att $Ax \leq b$?
Problemet tillhör **NP** ty givet en lösning x tar det $O(n^2)$ tid att kolla att $Ax \leq b$.
För att bevisa att problemet är **NP**-svårt
reducerar vi 3-CNF-SAT till det:
En instans av 3-CNF-SAT är en formel Φ över n variabler. Till x_i i Φ låter vi svara $y_i \in \{0, 1\}$ där 1 är sant och 0 falskt.
För varje klausul $c_j = l_1 \vee l_2 \vee l_3$ inför vi en olikhet

$$T(l_1) + T(l_2) + T(l_3) \geq 1$$

där $T(x_i) = y_i$ och $T(\neg x_i) = (1 - y_i)$.

0/1-programmering, forts.

Exempel:
Klausulen $x_1 \vee \neg x_2 \vee x_3$ ger olikheten
$y_1 + (1 - y_2) + y_3 \geq 1$.
Tolkning: Vänsterledet är sant (d.v.s. minst 1) om minst en av literalerna är sann (d.v.s. har värdet 1).
Formeln Φ omvandlas alltså till en uppsättning linjära olikheter som kan uppfyllas om och endast om Φ är satisfierbar.
$\Rightarrow$ 0/1-programmering är **NP**-fullständigt.
(Ofta vill man i tillämpningar maximera $c \cdot x$ givet att $Ax \leq b$ där x är en $\{0, 1\}$-vektor. Det är också **NP**-svårt.)