# Dagens föreläsning

- Repetition av klasserna P och NP
- Vad vet vi egentligen om relationen mellan dem?
- Därefter: räknestuga / Gruppövning 5

# Klasserna P och NP

P är klassen av alla språk som kan avgöras i polynomial tid med en deterministisk TM.
Hit hör problem som PATH, $\mathsf{E}_{\mathsf{DFA}}$, $\mathsf{A}_{\mathsf{CFG}}$.

NP är klassen av alla språk som kan...

- avgöras i polynomial tid med en ickedeterministisk TM.
- verifieras i polynomial tid med en deterministisk TM.

Hit hör problem som SAT, HAMPATH, QAP, och KNAPSACK.

# Relationen mellan P och NP

Det är fortfarande en öppen fråga om P och NP är samma klass.

Ett viktigt resultat som rör den här frågan lades fram av Stephen Cook och Leonid Levin i början på 1970-talet:

> Några av problemen in NP är sådan att om det fanns en polynomialtids algoritm för något av dem, då skulle samtliga problem i NP kunna lösas i polynomial tid.

Problem som har den här egenskapen kallas för NP-kompletta.

# Satisfierbarhet (SAT)

Givet: En boolsk formel, till exempel

$$\varphi = (\neg A \vee B) \wedge (\neg B \vee C) \wedge (\neg C \vee \neg B) \wedge A \ .$$

Frågan är: Kan de ingående variablerna (i det här fallet $A$, $B$, och $C$ tilldelas sanningsvärden (dvs. sant eller falskt) så att formeln som helhet blir sann? Som ett språk:

$$\mathsf{SAT} = \{\langle\varphi\rangle \mid \text{Det finns ett sätt att tilldela sanningsvärden till de ingående variablerna ...}\}$$

# SAT och frågan om NP = P

**Cook-Levin teoremet** $\mathsf{SAT} \in \mathsf{P} \iff \mathsf{P} = \mathsf{NP}$

Metoden som Cook och Levin använde för att bevisa det här resultatet kallas för *polynomialtids reduktion*.

# Polynomialtids reduktioner

- När vi arbetade med beräkningsbarhet kom vi också i kontakt med reduktioner.
- Då handlade det bara om problem var avgörbara eller ej, med andra ord, om dom gick att lösa.
- Nu vill vi även vet hur fort dom går att lösa.
- När vi säger att om ett problem $A$ kan reduceras effektivt till ett problem $B$, så betyder det att en effektiv lösning för $B$ även ger oss en effektiv lösning för $A$.
- "Effektiv" ska tolkas som "i polynomial tid".

# Polynomialtids reduktioner

**Definition 7.28** En funktion $f : \Sigma^* \mapsto \Sigma^*$ sägs vara *beräkningsbar i polynomial tid* om det finns en Turing maskin $M$ som givet $w$ som input stannar med exakt $f(w)$ på sitt band inom polynomial tid.

Minns ni varför vi ville ha exakt $f(w)$ på bandet och inget annat?

# Polynomialtids reduktioner

**Definition 7.29** Ett språk $A$ är polynomialtids reducerbart till ett språk $B$, och vi skriver $A \leq_P B$, om det finns en funktion $f : \Sigma^* \mapsto \Sigma^*$ sådan att

- den kan beräknas i polynomial tid, och
- för varje $w$ gäller att

$$w \in A \iff f(w) \in B \ .$$

Funktionen $f$ kallas då för en polynomialtidsreduktion av $A$ till $B$.

# Nyttan av polynomialtids reduktioner

**Teorem 7.31** Om $A \leq_P B$ och $B$ är i klassen P, då är $A$ också i P.

*Bevis.* Låt $M$ vara en polynomialtids TM som avgör $B$, och låt $f$ vara polynomialtidsreduktionen från $A$ till $B$. Då kan vi skapa en polynomialtids TM $N$ som avgör $A$ så här:

$N$ = "På input w:

1. Beräkna $f(w)$
2. Kör $M$ på $f(w)$ och svara som $M$ svarar."

# 3SAT

Vi tittar nu på en specialfall av SAT kallat 3SAT:

- En *litteral* är en (möjligtvis negerad) boolsk variabel. T.ex. $A$ eller $\neg A$.
- En *klausul* är en eller flera litteraler som är sammanfogade med $\vee$. T.ex. $(A \vee B \vee \neg C)$.
- En logisk formel är i *konjunktiv normalform*, även kallat CNF, om den innehåller en eller flera klausuler sammanfogade med $\wedge$. T.e.x

$$(A \vee B \vee \neg C) \wedge (\neg B \vee D) \wedge (A \vee C \vee \neg D) \ .$$

- En logisk formel är i 3CNF om samtliga klausuler har tre litteraler.

Språket 3SAT är då $\{\langle\varphi\rangle \mid \varphi$ är en satisfierbar formel i 3CNF form$\}$.

# 3SAT och CLIQUE

**Teorem 7.32** Problemet 3SAT är polynomialtids reducerbart till CLIQUE

*Bevisskiss.* Vi tittar på en polynomialtids reduktion $f$ från 3SAT till CLIQUE som konverterar formler till grafer. I den konstruerade grafen motsvarar en clique med en viss storlek en satisfierande tilldelning av de boolska variablerna i ursprungsformeln.

# Definition av NP-kompletta språk

**Definition 7.34** Ett språk $B$ är *NP-komplett* om det uppfyller följande två villkor

1. $B$ är i NP, och
2. varje $A$ i NP är kan reduceras i polynomial tid till $B$.

**Teorem 7.35** Om $B$ är NP-komplett, och $B$ är i P, då är P = NP.

# Hur NP-kompletthet "smittar"

**Teorem 7.36** Om $B$ är ett NP-komplett språk, och $B \leq_P C$ för något $C$ i NP, då är även $C$ ett NP-komplett språk.

*Bevis.* Vi vet redan att $C$ är i NP, så det som är kvar är att visa att varje språk $A$ i NP kan reduceras i polynomial tid till $C$. Eftersom varje språk kan reduceras till $B$ i polynomial tid, och $B$ kan reduceras till $C$ i polynomial tid, så har vi det önskade resultatet.

**Teorem 7.37** SAT är NP-komplett.