Avgjørelsesproblemer, P og NP

IN2010 – Algoritmer og Datastrukturer

Uke 45, 2020

Institutt for Informatikk

Avgjørelsesproblemer

- Et *avgjørelsesproblem* hvor svaret er JA/NEI
  "Sorted: Er en liste sortert?"
  "Reachability: Finnes det en sti mellom to par av noder i en graf?"
- En *instans* av et problem er inputtet

Sorted	
Instans:	En liste L
Spørsmål:	Er L sortert?

Reachability	
Instans:	En graf G og to noder v, w
Spørsmål:	Finnes det en sti fra v til w i G?

- JA-instans: instans som fører til output JA
  $1, 2, 5, 7, 8, 13$: JA-instans av Sorted
- NEI-instans: instans som fører til output NEI
  $1, 2, 5, 4$: NEI-instans av Sorted

Avgjørelsesproblemer

Men problemene vi har sett på hadde ikke JA/NEI svar!

- "Finn et element i en liste"
- "Hvor mange sterkt sammenhengende komponenter har en graf?"

Vi kan oversette disse problemene til relaterte avgjørelsesproblemer:

Contains	
Instans:	En liste L og et element e
Spørsmål:	Inneholder L elementet e?

SCC-k	
Instans:	En graf G og et tall k
Spørsmål:	Har G minst k sterkt sammenhengende komponenter?

Kompleksitetsklasser

- man ønsker å klassifisere problemer etter hvor vanskelige de er å løse
- vi skal se på to klasser avgjørelsesproblemer: P og NP
- P: løsninger kan bli effektivt beregnet
- NP: løsninger kan bli effektivt verifisert
- med "effektivt" mener vi i polynomisk tid

Kompleksitetsklassen P

- Problemer som kan *løses* i polynomisk tid
- med andre ord: problemer der vi vet at det finnes en algoritme som er i $O(n^k)$ for et tall $k > 0$.
- alle problemer vi har sett algoritmer for er i P
  - sorteringsalgoritmene løser

Sort	
Instans:	To lister L_1 og L_2
Spørsmål:	Består L_2 av elementene fra L_1 i sortert rekkefølge?

  - minimale spenntrær:

MST-k	
Instans:	En graf G, et tall k
Spørsmål:	Finnes det et spenntre i G som koster mindre enn k?

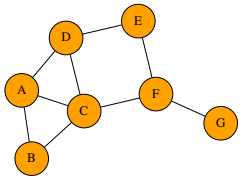
Løsning vs. verifisering

Her er tanken at å løse et problem er minst like vanskelig som å verifisere en løsning.

Men hva betyr verifisering for avgjørelsesproblemer?

- En algoritme som verifiserer et problem tar instansen og et *sertifikat* som input. Output blir JA hvis instansen er en JA-instans.

Reachability	
Instans:	En graf G og to noder v, w
Spørsmål:	Finnes det en sti fra v til w i G?



- Instans: graf til høyre og to noder A og G
- mulig sertifikat: en sti fra A til G: A, C, F, G
- verifiseringsalgoritme: sjekk om sertifikatet er en sti fra A til G i grafen

Kompleksitetsklassen NP

- Problemer som kan *verifiseres* i polynomisk tid
- Ekvivalent definisjon: Problemer som kan løses av *ikke-deterministiske* algoritmer i polynomisk tid.[1]
- Så et problem L er i NP hvis det finnes en algoritme V som tar som en instans av L og et sertifikat som input, og outputter JA for JA-instanser innen polynomisk tid ($O(n^k)$ for et tall $k > 0$).

[1] mer om dette i en senere video, ikke eksamensrelevant

Kompleksitetsklassen NP: Eksempler

- Reachability er i NP, siden verifikasjonen (sjekk om sertifikatet er en sti fra v til w i G) kan gjøres i polynomisk tid

Algorithm 1: Reachability Verifier

Input: En graf G, to noder v, w, og en liste L av noder
Output: JA hvis L er en sti fra v til w i G

```
1 Procedure Reachability-Verifier(G, v, w, L)
2     n = L.length
3     if L[0] ≠ v or L[n − 1] ≠ w then
4         return NEI
5     for i from 1 to n − 1 do
6         if (L[i − 1], L[i]) not edge in G then
7             return NEI
8     return JA
```

Reachability	
Instans:	En graf G og to noder v, w
Spørsmål:	Finnes det en sti fra v til w i G?

P er i NP

- Hvis vi kan løse i polynomisk tid, så kan vi også verifisere i polynomisk tid!
- Verifiseringsalgoritmen kan da bare løse problemet, og trenger ingen sertifikat

Algorithm 2: Verifying by solving

Input: En graf G, to noder v, w, og en liste L av noder
Output: JA hvis L er en sti fra v til w i G

```
1 Procedure Reachability-Verifier2(G, v, w, L)
2     BFS(G,v)
3     return w.visited
```

- Men finnes det problemer i NP som ikke er i P?
- Vi vet ikke! (men de fleste antar P$\neq$NP)
- Clay Mathematics Institute gir ut \$1mill til den som løser problemet!

"Vanskelige" problemer i NP

- vi vet av noen (mange) problemer at de er de vanskeligste problemene i NP
- disse heter NP-komplette problemer

Hamiltonsykel	
Instans:	En graf G
Spørsmål:	Finnes det en sykel i G som besøker alle noder nøyaktig en gang?

Knapsack	
Instans:	En mengde objekter med hver sin vekt og verdi, og to tall s og t
Spørsmål:	Finnes det en mengde objekter som som tilsammen er verdt mer enn t og veier mindre enn s?

Sudoku	
Instans:	Et ufullstendig fylt ut $n \times n$ Sudoku brett
Spørsmål:	Har inputbrettet en gyldig løsning?