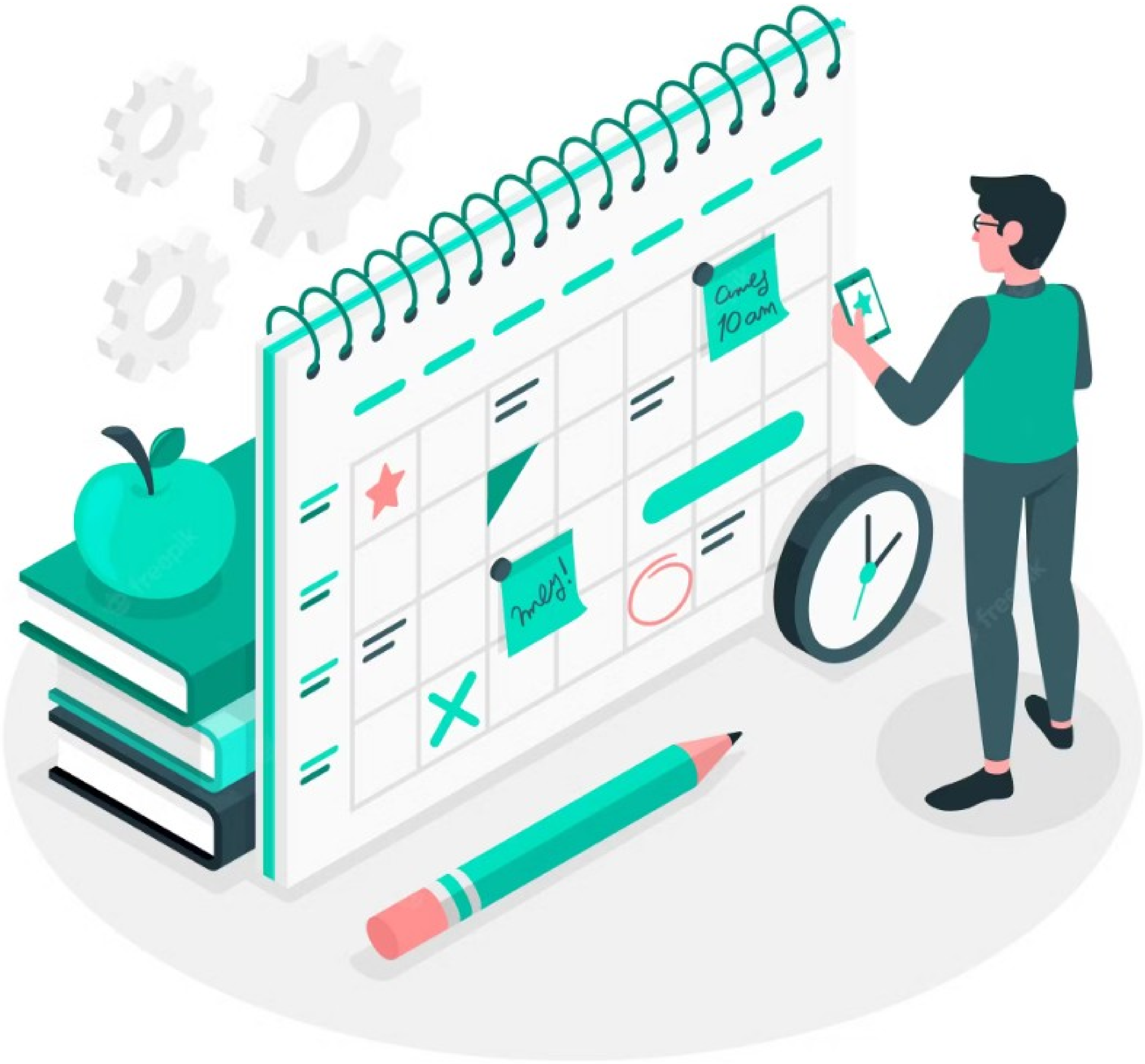
IN2010 Gruppe 4

Repitisjon: O-Notasjon og Kjøretidsanalyse



Dagens Plan

- Info
- Gjennomgang(O notasjon)
- Felles øvelser
- Pause
- Gjennomgang(Kompleksitet)
- Gruppeoppgaver

Info

Prøveeksamen

Gjennomgås på mandag

Gjennomgang

Hva er vanskelig med O-notasjone?

beregne kompleksitet	Analysere en algoritme	hvor mange iterasjoner de ulike delene bruker	hvordan telle steg
Hva er while-løkke kjøretid?	Beregne kompleksitet ved å se på en kode	kjenne igjen doble løkker	Antall iterasjoner

3

20

Mentimeter

Hva er vanskelig med O-notasjone?

Beregne, telle steg	beregne	analysere	skal vi alltid tenke worst-case
Nuh uh	Grafer	alt	med grafer blir det fort litt forvirrende, når det er mye rekursjon osv

3

20

Mentimeter

Hva er vanskelig med O-notasjone?

bruken i grafer

Beregne kompleksitet

finne hvilke mengder kompleksiteten er avhengig av (for eksempel |V|, |E|)

Regne kompleksiteten til en algoritme

hvordan to algoritmer med samme kjøretid kan føre til at den ene algoritmen fortsatt er raskere enn den andre?

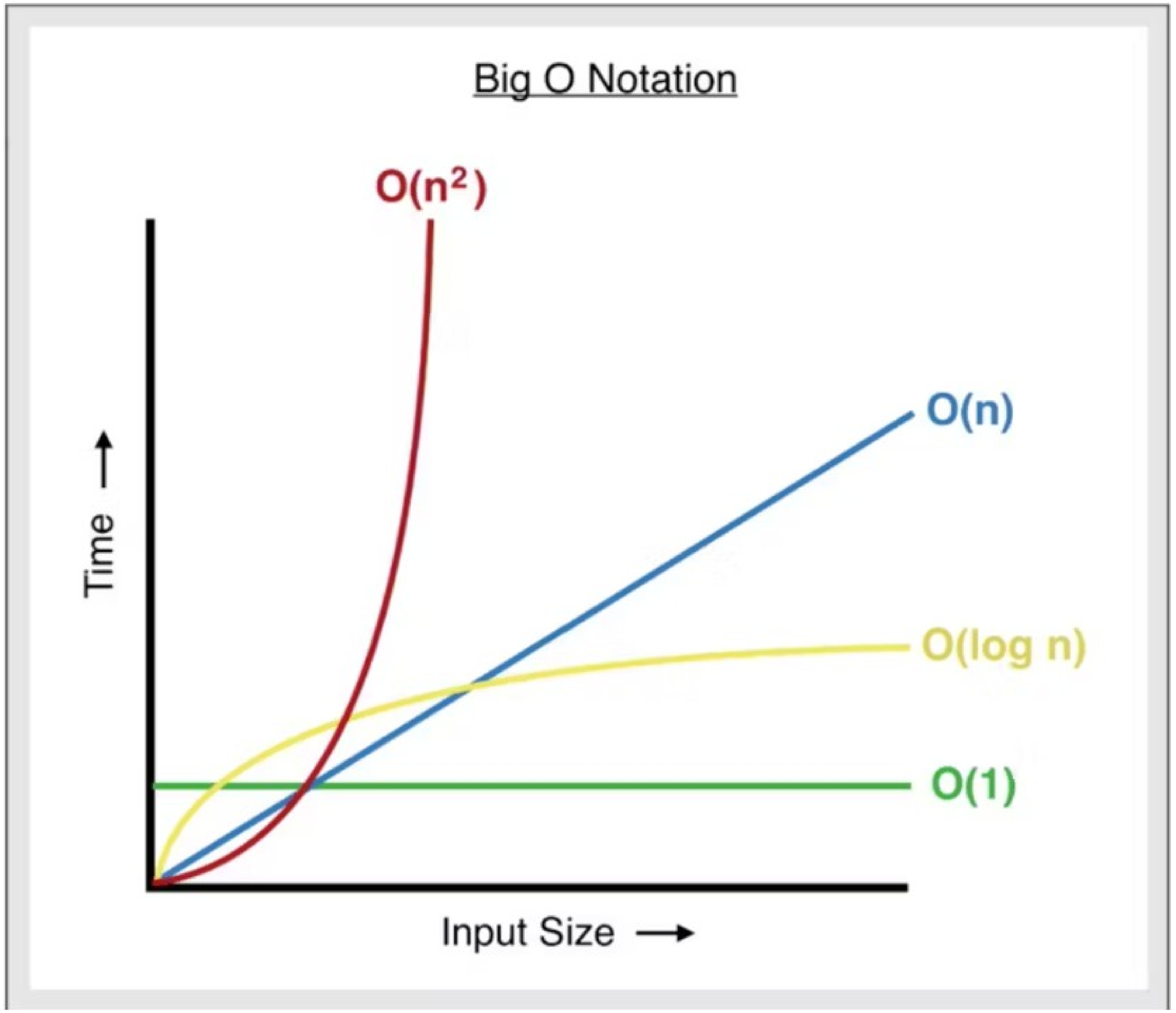
Flere funksjoner kaller hverandre!

i hvilke tilfeller kan konstanter være av betydning?

å vite når stegene er avhengig av input size

3

20



O-Notasjon

- Hvor vanskelig er et problem/algoritme å løse?
- Gitt en algoritme: Hvor mange steg tar den for å fullføre
- Er disse stegene avhengig av input størrelse? Isåfall, hvordan?

Noen regler

Når vi gjør kjørtidanalyse kan vi abstrahere vekk mange detaljer!

Eks: hardware, programmeringspråk osv

2

Noen regler(cont.)

- Primitive steg(Tilordning, Indeks aksessering, aritmetiske operasjoner og returnering) er konstant tid
- Metode kall er avhengig av kjøretiden til metoden som blir kalt på
- Ting som ikke avhenger av input størrelse(n) kan ses på som konstant tid

Big O

- Worst case kjøretid
- N kan bli uendelig stor
- Grenseverdien til N fører til at vi kan se bort fra konstanter

4

Eksempler

```javascript
const smallNumber = 1000;
const biggerNumber = 10000;

function countOperations(n) {
  let operations = 0;
  let i = 1;

  while (i < n) {
    i = i * 2;
    operations++;
  }

  return operations;
}
```

Hva er kjøretiden på CountOperations?

8

Mentimeter

Hva er kjøretiden på CountOperations?

log n

log n

O(logn)

logggi

logN

O(log(n)) siden mengden av trekk blir halvert hver gang en iterasjon i while løkken kjører

O(log(n))

O(n-1)

1

19

Hva er kjøretiden på CountOperations?

log n	logn	log(n)	log(n)
Log(N)	O(log(n))	O(log n)	lineær

1

19

Mentimeter

Hva er kjøretiden på CountOperations?

O(log n)

O(log(n))

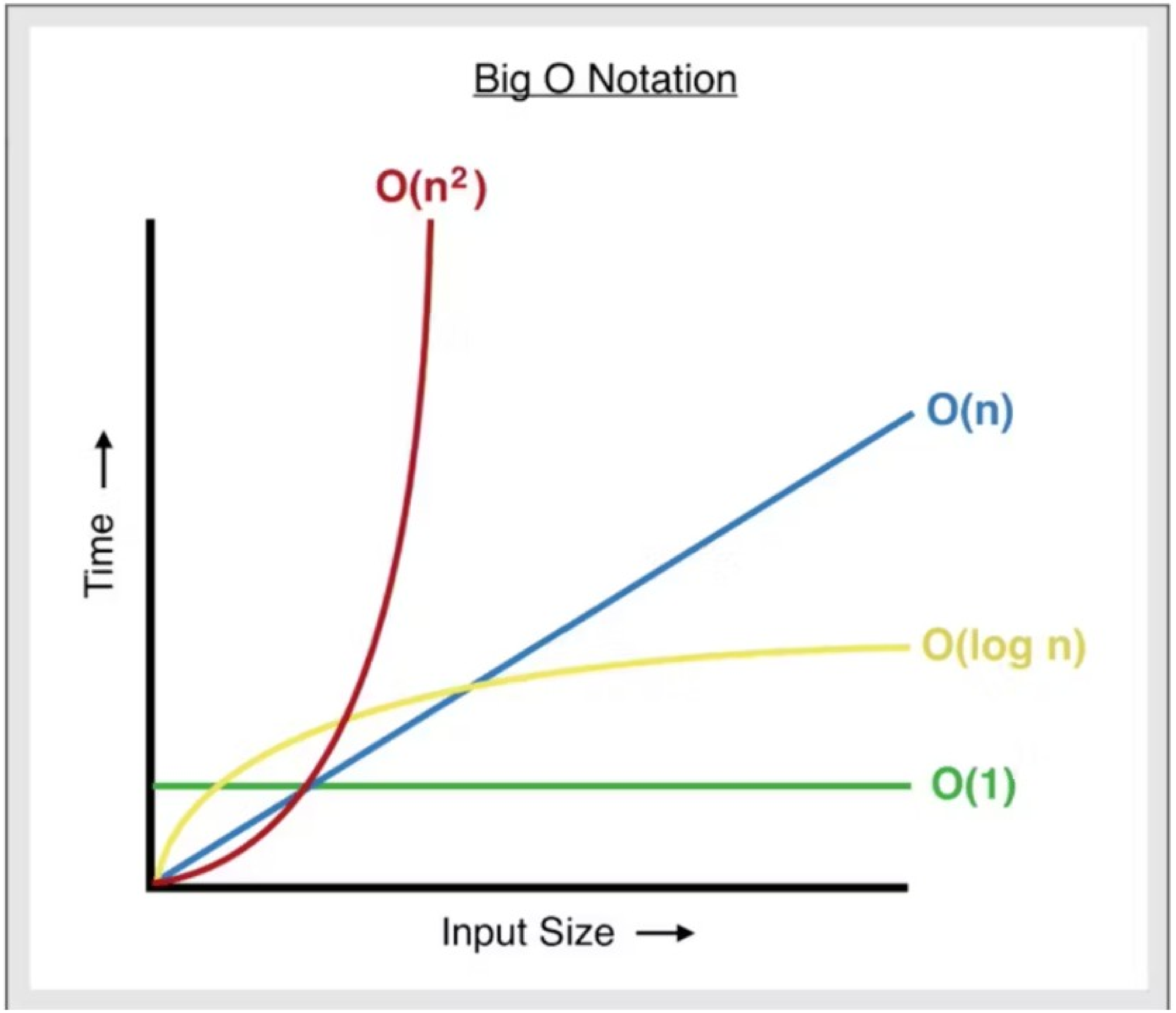
log n

1

19

Tips

- Se på ledd og kartlegge det største leddet
- Telle steg i løkker: Hvor mange ganger kjører en løkke i forhold til n
- Avhengigheter: Blir funksjoner kalt på i andre funksjoner eller løkker?



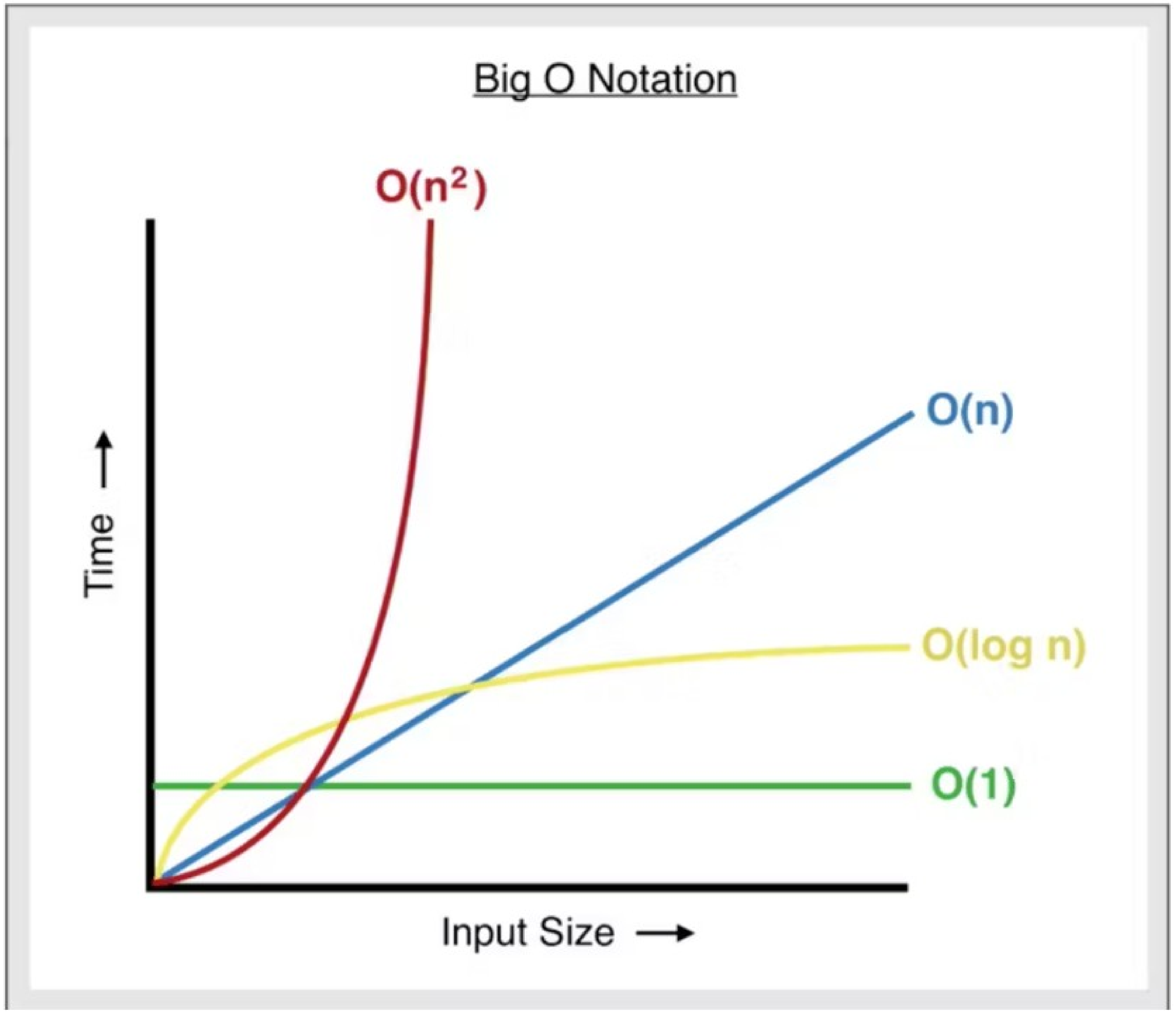
Bestemme O

- På eksamen, og generelt så ønsker vi ikke konkret etter antall steg
- Eks: (4 + 2 +3n + 10n)
- Mye av dette kan abstraheres

Lifehack regler

- Når det er ledd mellom plusstegn, så kan man ta hensyn til leddet som vokser raskets
- Konstanter og koffesienter kan man se bort ifra, eller redusere ned til 1
- Variabler kan faktoriseres for å gjøre det enklere å se hva som vokser rasketst
- Om algoritmen er avhengig av flere inputstørrelser så må man skille de ved bruk av n og m
- NB! Bare for at m og n begge grenser mot uendelig, så kan ikke O(m+n) forkortes til O(n)

Mentimeter



Bestemme O

- O(n^3 + 50n^2 + 10000)
- O((n + 30) * (n + 5))
- O(n*log(n) + log(n)*log(n))
- O(n + n + n + n + n + n)

Bestemme O

- → $O(n^3 + 50n^2 + 10000) = O(n^3)$
- → $O((n + 30) * (n + 5)) = O(n^2)$
- → $O(n\mathit{log(n)} + \mathit{log(n)}\log(n)) = O(n*\log(n))$
- → $O(n + n + n + n + n + n) = O(6n) = O(n)$

Felles Oppgaver

3

```
int proc(int n) {
  int x = 1;

  if (n > 1) {
    for (int i = 1; i <= n − 1; i++) {
      x = x + proc(i);
    }
  }
  return x;
}
```

EKSAMEN H2013

Hva er kjøretiden til proc?

4

Mentimeter

```
int x = 0;
for (int i = 0; i < n; i++){
    for (j = 0; j < i*i; j++){
        x = x + j;
    }
}
```

EKSAMEN H2015

Hva er kjøretiden?

1

```
int x = 0;
for (int i = 1; i <= log n; i ++) {
 for (int j = 1; j < = i; j++) {
 x = x + 1;
 }
}
```

EKSAMEN H2016

Hva er kjøretiden?

2

```
// assume positive input values

int[] sorted(int[] input){

    int[] sort = new int[input.length];

    int max = 0;

    for( int i = 0; i < input.length; i++ ){
        if(input[i] > max){
            max = input[i];
        }
    }

    int[] b = new int[max + 1]; // all values are 0

    for( int i = 0; i < input.length; i++ ){
        b[ input[i] ]++;
    }

    int counter = 0;

    for( int i = 0; i < b.length; i++ ){
        while( b[i] > 0 ){
            sort[counter] = i;
            b[i]--;
            counter++;
        }
    }

    return sort;
}
```

EKSAMEN H2010

Hva er worst case tidskompleksitet til denne metoden, gitt som O(n, m) hvor: n = input.length og m = største verdi i input array

Mentimeter

Noe uklart?

hjelp

Hadde man hjelpemidler tilgjengelig under de eksamensoppgavene vi gjennomgikk???????

:(

sheeeeeeeet

Er et np-hardt problem utenfor np?

er det noen eksamens oppgaver hvor ingen fikk poeng på oppgaven?

hva bruker man avgjørelsesproblemer p, np osv til i praksis?

2

6

Pause

Beregnbarhet og Kompleksitet

Denne delen handler ikke om å "LØSE" problemer, men heller det å "AVGJØRE/BESTEMME" hvorvidt et problem er løselig eller ikke

Avgjørelsesproblemer

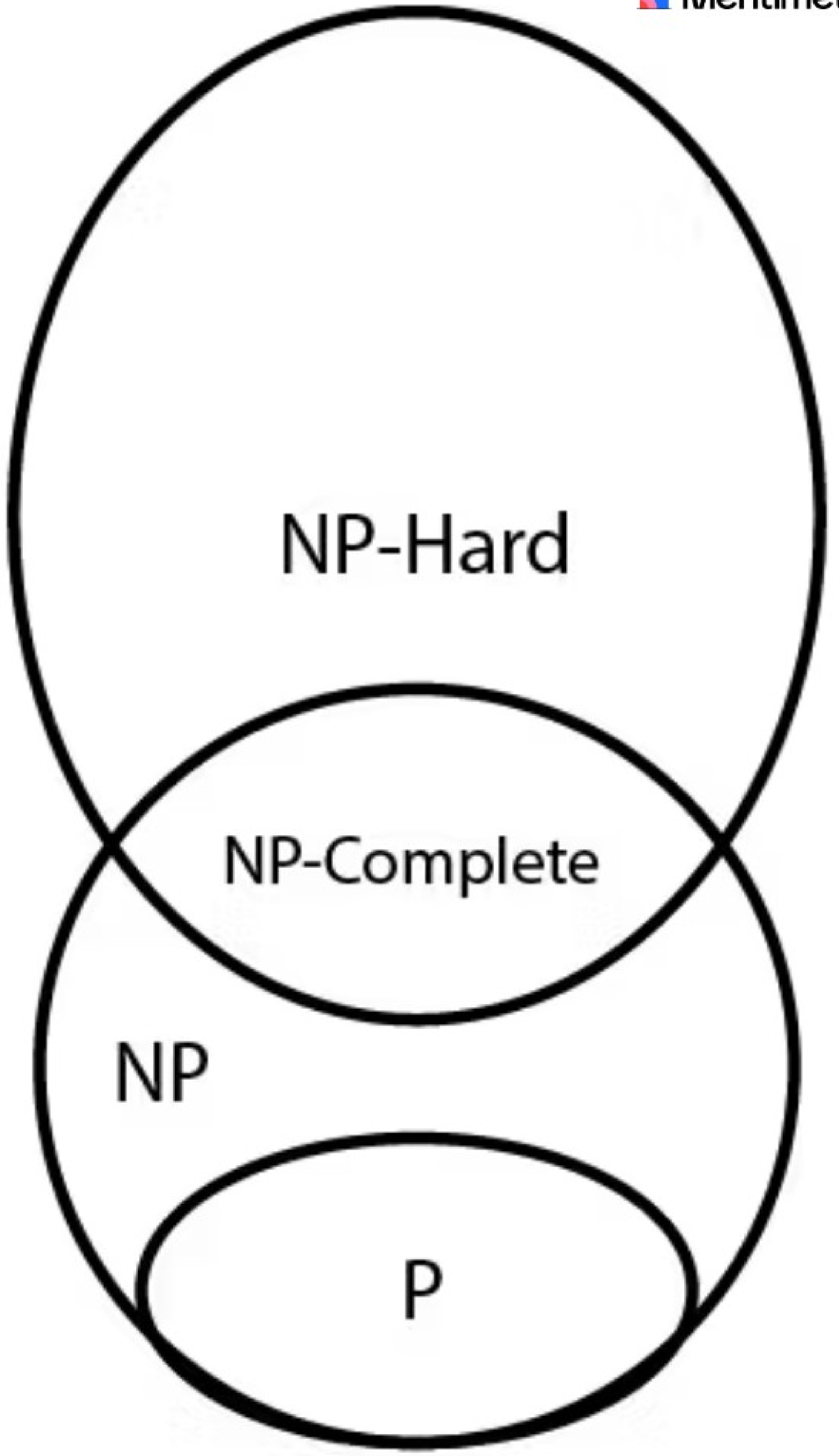
- Et avgjørelsesproblem er basically et problem/spørsmål som vi kan avgjøre om det er mulig å løse eller ikke.
- Svaret/outputten på avgjørelsesproblemer er enten ja eller nei
- Eksempel: Er det mulig å telle alle i klasserommet?
- Eksempel: Er det mulig sortere en liste?

Kompleksitetklassen P:

- Klassen P er en samling av alle avgjørelsesproblemer som kan løses i polynomiell tid
- Polynomiell tid: O(n^p), der p er et polynom
- Nøkkelordet her er at problemet LØSES i polynomisk tid
- Dette gjelder også for alt som kjører raskere

Kompleksitetsklassen NP

- Et problem som kan verifisere(løses av en ikke deterministisk algoritme)
- Den ikke deterministiske algoritmen må kjøre i polynomiell tid.
- NP = Non-Determenistic polynomial time



2

Sudoku

7	2	9	1	4	3	6	8	5
3	6	1	7	5	8	9	2	4
5	8	4	9	6	2	7	1	3
9	4	2	5	1	7	8	3	6
8	7	5	3	9	6	2	4	1
1	3	6	2	8	4	5	9	7
4	1	7	8	2	5	3	6	9
6	5	8	4	3	9	1	7	2
2	9	3	6	7	1	4	5	8

Sudoku

P og NP

- Om et problem kan løses i polynomiell tid, så kan den også verifiseres i polynomiell tid.
- Alt i P er også i NP
- Men alt i NP er ikke i P
- Det er ikke bevist at P=NP
- Men det er også ikke bevist at P≠NP

3

Fellesoppgaver

1(f)

Kodeanalyse

```
boolean loops(int n) {
    boolean output = true;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            for (int k = 0; k < n; k++) {
                for (int l = 0; l < n; l++) {
                    for (int m = 0; m < n; m++) {
                        output = !output;
                    }
                }
            }
        }
    }
    return output;
}
```

Gi kjøretiden til kodesnutten i O-notasjon:

Eksamen 2019

1(g) **5-CLIQUE**

En k-klikk, eller en klikk av størrelse k, i en graf $G = \langle V, E \rangle$ er en delmengde $C \subseteq V$ av k noder som utgjør en komplett graf. Det vil si at hvis $u, v \in C$ er to forskjellige noder i klikken, så må $\{u, v\} \in E$.
I problemet 5-CLIQUE skal du avgjøre om en graf inneholder en klikk av størrelse 5.

5-CLIQUE
INSTANS: En graf G
SPØRSMÅL: Inneholder G en 5-klikk?

Hint: Se forrige oppgave.

Velg ett alternativ:

- 5-CLIQUE er NP-komplett
- 5-CLIQUE er i P

Eksamen 2019

1

Gruppeoppgaver

Spørsmål?

13 questions
1 upvote