Intro + Sekvensielle datastrukturer



IN2010 – Uke 2

Om meg!

Spørsmål? marikram@uio.no

- Maria (Svarer også om du sier ‘Marie’ eller ‘Mari’)
- 3. år med språkteknologi (40-gruppe i lingvistikk!)
- Tok IN2010 høst 2025
- Første år som gruppelærer





Smalltalk guide:



- «Sett noen gode filmer i det siste?»
- «Hva er det beste Shakespeare stykket?»
- «Hvordan gikk kampen til Arsenal sitt kvinnelag denne helgen?»



- «Hvordan fungerer egentlig the block chain?»
- «Hvilke temaer kommer på eksamen i år?»
- «Hvordan gikk kampen til Strømsgodset denne helgen?»

Navnerunde (3-4 min)

- Navn
- Linje
- (Frivillig) Et tema du gjerne prater om / gjerne ikke prater om

Litt tips for IN2010

- Gjør så mange oppgaver som mulig!
  - Gjør den samme oppgaven flere ganger (prøv å optimaliser algoritmen din)
- Forklar algoritmer høyt til andre + visualiser dem
  - Helst med pen og papir
- Ikke bli stressa om alt ikke klikker med en gang!
  - IN2010 er vanskelig. Det krever mengdetrening
  - Få overordnet forståelse → Forstå kode → Gjenskap kode → Forstå kjøretid

Kjøretidskompleksitet + Stor O



Stor O

- Antall steg i forhold til hvor mye input den tar inn (f.eks. størrelsen på et array.)
- «Om jeg øker lengden på dette arrayet, hvor mye treigere blir programmet?»

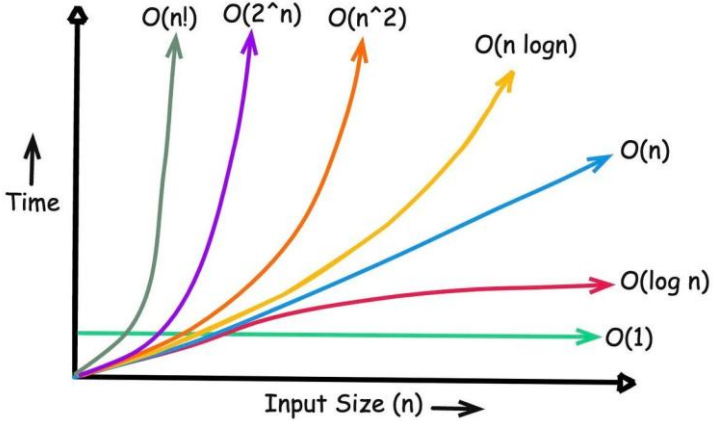
F.eks.:

Konstant tid O(1) – Programmet blir ikke noe treigere om du øker input

Lineær tid O(n) – Om du dobler mengden n, dobler du kjøretiden.

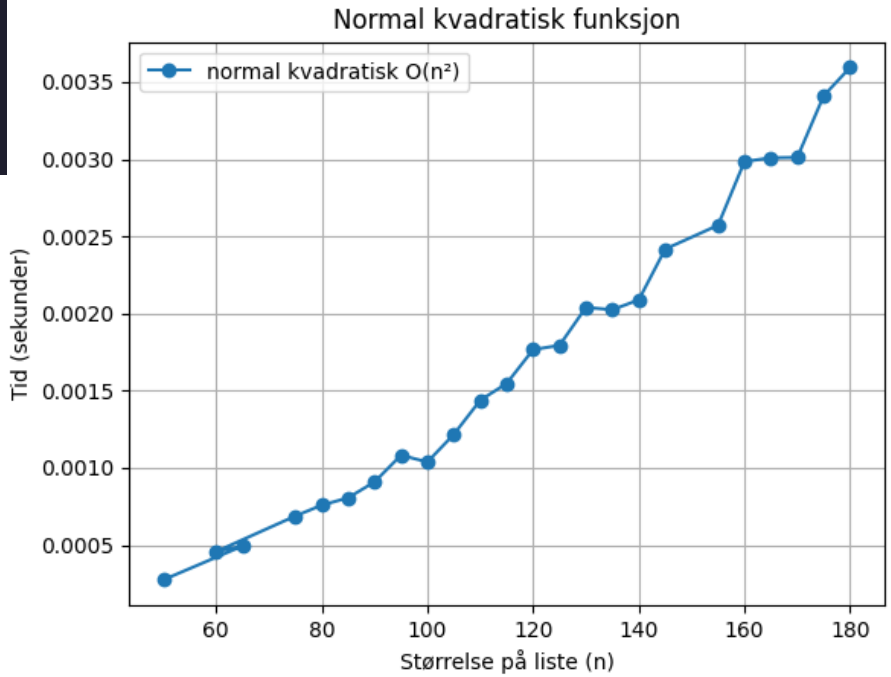
Stor O

- Vi definerer kjøretiden til et program ut ifra ‘formen’ på grafen dens.
- Vi kan hovedsakelig ignorere de faktiske verdiene til y og x aksen.
- https://youtu.be/RGuJga2Gl_k



Stor O

```python
def normal_kvadratisk_funksjon(liste):
    tall = 0
    for i in liste:
        for j in liste:
            tall += i * j
    return tall
```

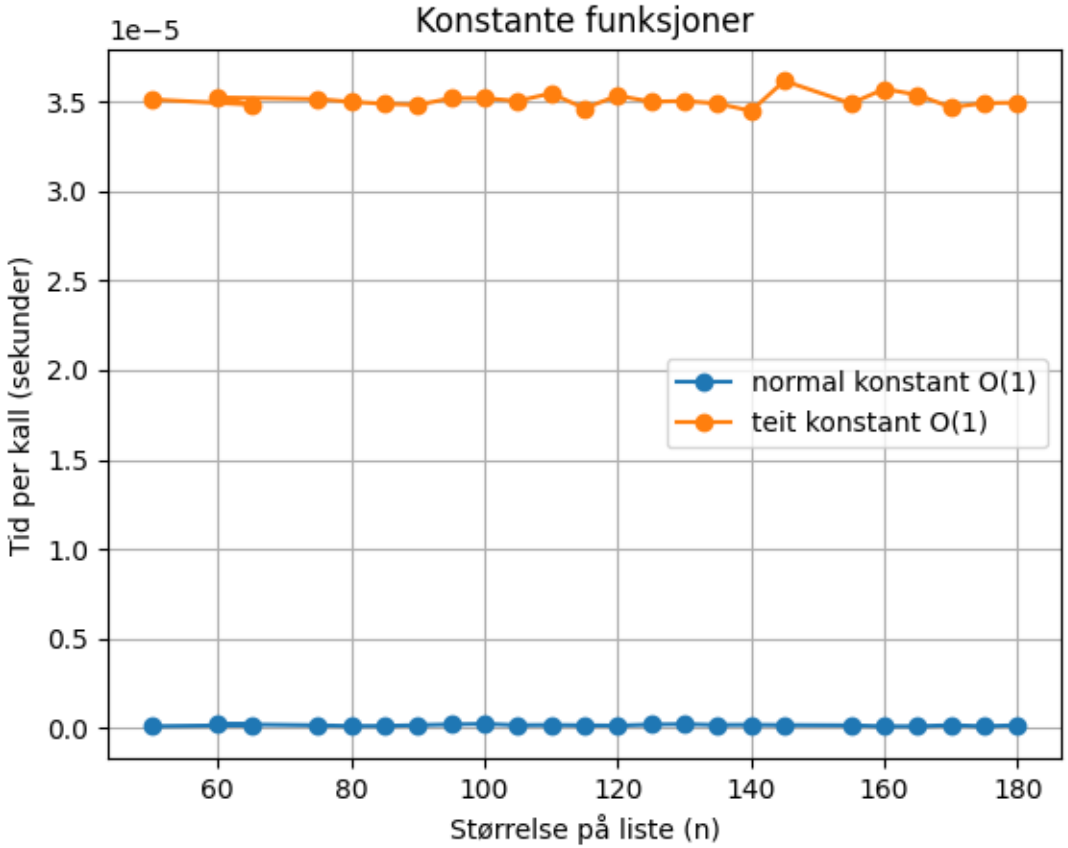


```python
def normal_konstant_funksjon(liste):
    if len(liste) > 0:
        return liste[0]
```

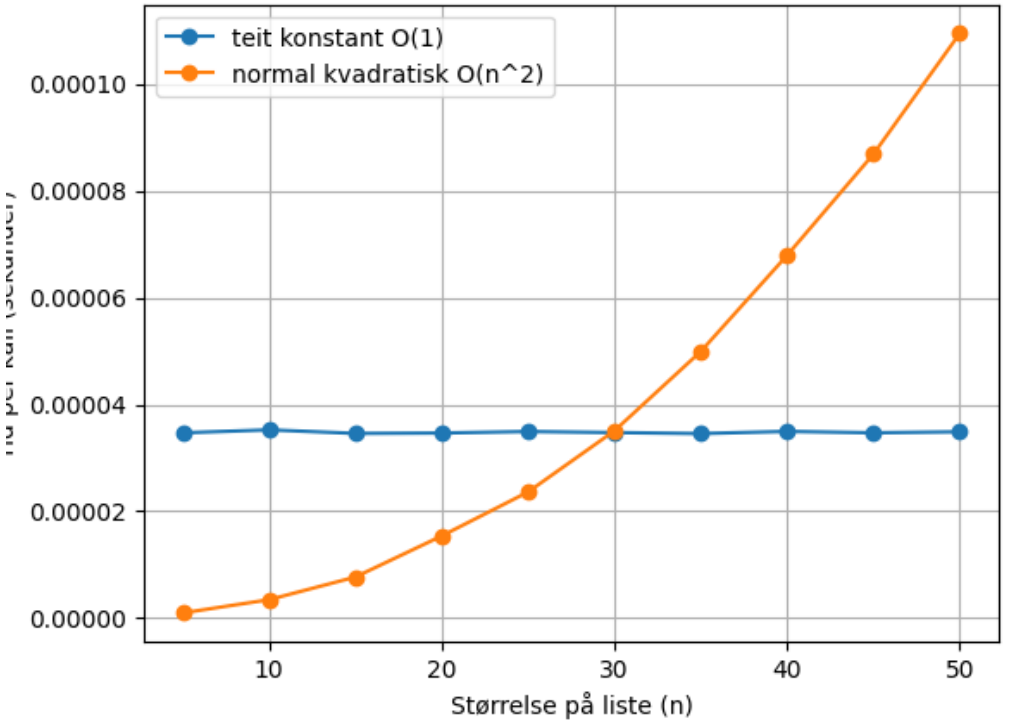
```python
def teit_konstant_funksjon(liste):
    if len(liste) > 0:
        variabel = liste[0]
    else:
        return
    teller = 0
    while teller < 200:
        teller += 1
        variabel += 3

        if teller % 2 == 0:
            variabel = variabel * 1
        if teller % 5 == 0:
            variabel = variabel / 2

    return variabel
```


Konstante funksjoner
1e−5
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Tid per kall (sekunder)
60
80
100
120
140
160
180
Størrelse på liste (n)
normal konstant O(1)
teit konstant O(1)

- Bedre stor O betyr ikke nødvendigvis bedre tid (…men den gjør ofte det)
- Noen algoritmer har ‘dårligere’ tid i stor O men fungerer raskere på lite input



Ukas video!

- Kjøretidskompleksitet, en liten sneak peak på sorteringsalgoritmer, og litt generell utviklervisdom

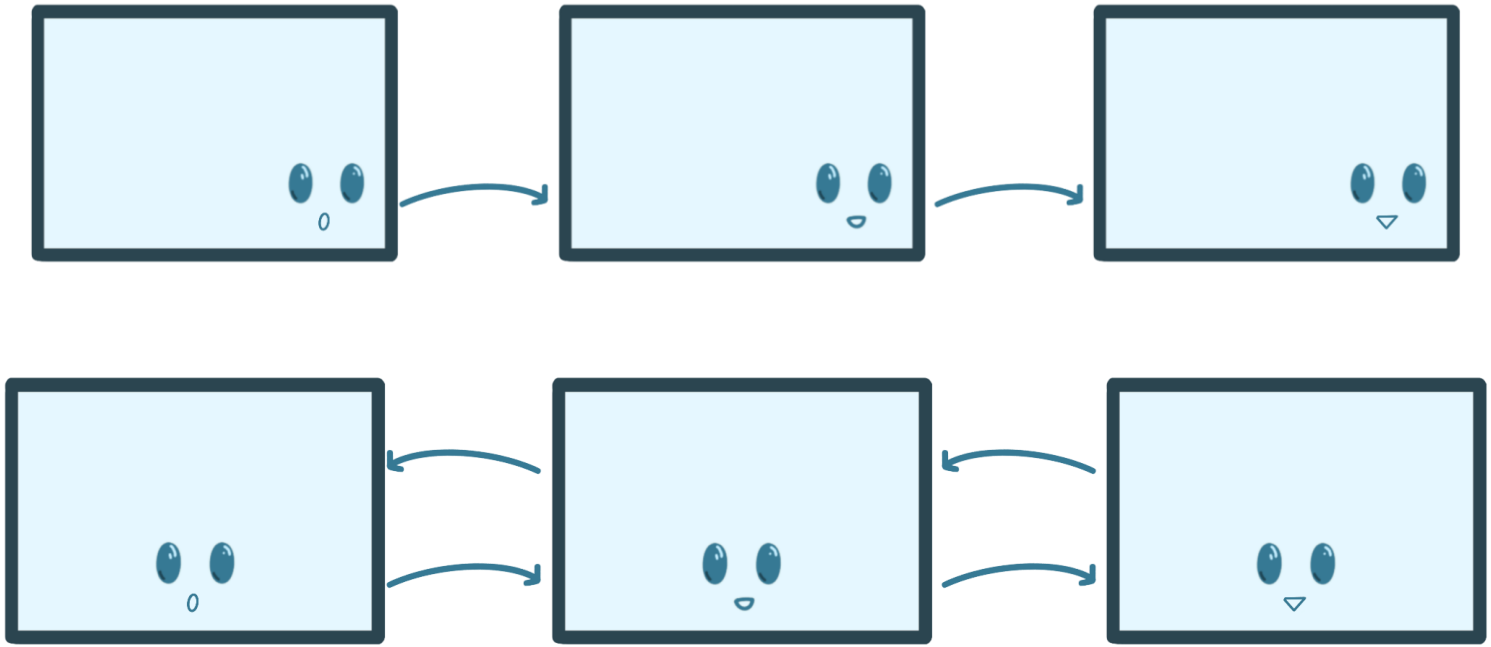
https://youtu.be/RGuJga2Gl_k

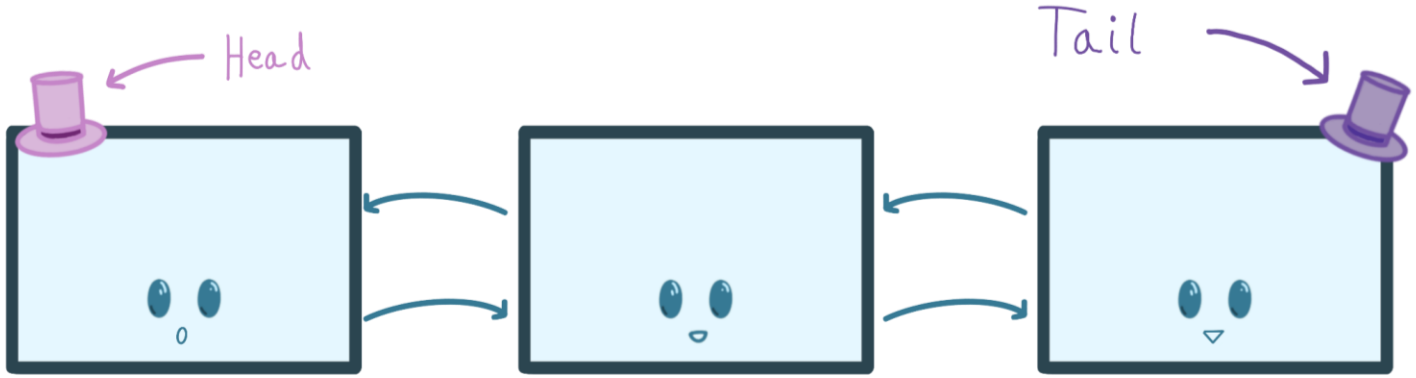


Sekvensielle datastrukturer



Enkeltlenket vs dobbellenket







← Kan også løses som en sirkel, med et felles hode

```
class SinglyLinkedList

    first, last <-- null
    // ...

    method removeLast()
        pointer <-- first

        if pointer = null then
            error "Cannot remove from an empty list"

        if pointer.next = null then
            first, last <-- pointer.next
            return

        while pointer.next.next != null do
            pointer <-- pointer.next

        pointer.next <-- null
        last <-- pointer
```

```
class DoublyLinkedList

    first, last <-- null
    // ...

    method removeLast()
        if last != null then
            last <-- last.prev

        if last = null then
            first <-- null
        else
            last.next <-- null
```

a) Gi en kort forklaring (med naturlig språk) av hvordan algoritmene fungerer. Hvilken er mest effektiv?

Tenk først selv 3 min.

Prat i grupper 4-5 min.

ALGORITHM: DOBLE KAPASITETEN TIL ET DYNAMISK ARRAY VED BEHOV

Input: Et dynamisk array D
Output: Et dynamisk array med plass til minst ett element til

```
1 Procedure EnsureCapacity(D)
2     if D.n < |D.A| then
3         return
4     A ← new array of size |D.A| · 2
5     for i ← 0 to D.n − 1 do
6         A[i] ← D.A[i]
7     D.A ← A
```

ALGORITHM: SETT INN I ET DYNAMISK ARRAY PÅ INDEKS

Input: Et dynamisk array D, en indeks i og et element e
Output: Et dynamisk array med e lagt til på indeks i

```
1 Procedure add(D, i, e)
2     EnsureCapacity(D)
3     for j ← D.n − 1 down to i do
4         D.A[j + 1] ← D.A[j]
5     D.A[i] ← e
6     D.n ← D.n + 1
```

b) Gi en kort forklaring (med naturlig språk) av hvordan algoritmene fungerer. Hva er kjøretidskompleksiteten?

Tenk først selv 3 min.
Prat i grupper 4-5 min.

Oppgaver!

Psst, jeg anbefaler å gjøre dem i grupper

Bonusoppgave

Implementer en dobbellenket liste med metodene get(L,i), set(L,i,e), add(L,i,e) og remove(L,i).

Hint: Du bør i så fall også lage en enkel node-klasse + hjelpemetodene NodeAt(L,i) og InsertBefore(v,e)