ICA.05

Gruppe 12
Guro Kalland
Lene Odde
Martin Nenseth
Robert Berntsen
Sindre Grønstøl Haugeland
Thomas Berntsen

Martin - ASUS ZENBOOK UX305FA

Type	**Navn**	**Klokkefrekvens**	**Cache**
CPU	Intel Core M3 - 5Y10	0,8GHz (2,0GHz - turbo)	L2 kombinert: 0,5MB L3 kombinert: 4MB
RAM	LPDDR3 (8GB)	1600MHz	
GPU	Integrated Intel HD graphics 5300	850MHz	
OS	Windows 10		
HDD	SanDisk SD7SN3Q (128GB)	Read/write: ~420/300MB/s	

Lene - MacBook Pro

Type	**Navn**	**Klokkefrekvens**	**Cache**		
CPU	Intel Core i5	2,5GHz			
RAM	4 GB DDR3	1600 MHz			
GPU	Intel HD Graphics 400 1024 MB	800 MHz			
OS	OS X Yosemite				
HDD	SATA	5400 RPM			

Guro - HP ENVY 15-j019so Notebook

Type	**Navn**	**Klokkefrekvens**	**Cache**
CPU	Intel Core i7-4700MQ	2,4 GHz	L1 256 KB L2 1024 KB L3 6144 KB
RAM	12 GB (3x4)DDR3	1600MHz	
GPU	NVIDIA GeForce GT 740M	810 MHz	
OS	Windows 10		
HDD	Toshiba mq01abd100 1 TB 5400 rpm		8 MB

Thomas - MacBook (Retina, 12-inch, Early 2015)

Komponent	**Beskrivelse**	**Klokkefrekvens**	**Cache**
CPU	Intel Core M	1,1 (2,4) GHz	4 MB
RAM	8 GB DDR3	1600 MHz	
GPU	Intel HD Graphics 5300 1536 MB		
Harddisk	256 GB PCIe-basert integrert flashlagring		
Operativsystem	OS X El Capitan Versjon: 10.11.3		

Robert - UX305F Asus Zenbook pc

Type	**Navn**	**Klokkefrekvens**	**Cache**
CPU	Intel core M3 -5Y10c	0.80 GHz max speed 2.0Ghz	L1 :128 kb L2: 512kb L3: 4096kb
RAM	LPDDR3 SDRAM (8GB)	1600 Mhz	
GPU	Integrated intel HD graphics 5300	850 Mhz	
Operativsystem	windows 10		

Harddisk	Micron_M600	Read 560MB/s Write 510MB/s	

Sindre

CPU	Intel i5 5200u	2.2 -> 2.6 GHz	3MB kombinert cache	
Ram	8gb	1600 MHz		
Operativsystem	Windows 10			
Disk	Samsung 128 gb SSD	Read/Write: 480/98 MB/s		

Beskrivelse av komponentene

Central processing unti(CPU) er hovedregne og prosesseringsenheten i datamaskin. Den utfører instruksjoner gitt av programmer man kjører. Prosessoren har ansvar for alt som skal utføres, og ved behov vil delegere bort det ansvaret til andre komponenter.

CPU og RAM samarbeider for å gjennomføre alle slags oppgaver, for at disse komponentene skal yte 100% er det viktig at de ikke holder hverandre igjen. F.eks. vil en pc ha problemer med å ha mange dokumenter, faner i nettleser og andre programmer oppe samtidig hvis man kun har 2GB minne, til tross for at man har den nyeste og beste prosessoren på markedet.

Grafikkprosessor eller GPU (Graphics Processing unit) er mikroprosessoren som behandler grafikk. GPU fungerer på mange måter som CPU-en, men er mer dedikert til grafikkdata. GPU-en er som oftest plassert på et eget kort eller enhet i datamaskinen (skjermkort) og får ordre fra data og hovedkortet.

Harddisk er en permanent lagringsenhet som lagrer informasjon frem til den blir slettet av en bruker eller et program. Kapasiteten på harddisken bør oppgis som lagringskapasitet og ikke minne.

Et operativsystem (os) er programvaren på datamaskinen som tildeler de forskjellige ressursene i datamaskinen til andre programmer.

TimeIt test1.0

- Lista inneholder tall fra 0 til 10000.
- I testen kjøres begge funksjonene 100 ganger, og testen selv kjøres 100 ganger.

Navn:	**Tid: search_slow**	**Tid: search_fast**
Sindre	2.145288	0.1154730
Lene	0.276646	0.029681
Robert	0.606931	0.007416
Thomas	0.033358	0.001385
Guro	1.802864	0.306595
Martin	3.076967	0.128483

Gjennomsnitt slow: 1,323675666
Gjennomsnitt fast: 0,09817216666

TimeIt test1.1

- Liste inneholder 1000 tilfeldige tall.
- I testen kjøres begge funksjonene 100 ganger, og testen selv kjøres 100 ganger.

Navn:	**Tid: search_slow**	**Tid: search_fast**
Sindre	0.211480	0.012783
Lene	0.027365	0.003976
Robert	0.060384	0.002894
Thomas	0.033359	0.015614
Guro	0.250204	0.087521
Martin	0.030948	0.005479

Gjennomsnitt slow: 0,192866
Gjennomsnitt fast:0,02137783333

TimeIt test1.2

- Liste inneholder 100 tilfeldige tall.
- I testen kjøres begge funksjonene 100 ganger, og testen selv kjøres 100 ganger.

Navn:	**Tid: search_slow**	**Tid: search_fast**
Sindre	0.021271	0.006075
Lene	0.002965	0.000644
Robert	0.007743	0.000139
Thomas	0.033776	0.003508
Guro	0.034457	0.004798
Martin	0.030948	0.005479

Gjennomsnitt slow: 0,02186
Gjennomsnitt fast: 0,0034405

TimeIt test1. 3

- I testen kjøres begge funksjonene 100 ganger, og testen selv kjøres 100 ganger.
- Denne gangen legger vi 50 string elementer i lista.

Navn:	**Tid: search_slow**	**Tid: search_fast**
Sindre		
Lene		
Robert		
Thomas		
Guro		
Martin		

Hvorfor er det slik?

Når du bruker Python trenger du ikke bekymre deg for arrays, hvordan arrangere minnet, eller i hvilken rekkefølge det blir sendt til CPU. Det gjør at du kan fokusere på algoritmene som blir iverksatt, men ulempen er at det tar opp stor ytelse. I kjernen til Python, kjøres det et sett med svært optimaliserte instruksjoner. Trikset er midlertidig å få pyton til å utføre dem i riktig sekvens for å oppnå bedre ytelse.

```python
def search_fast(haystack, needle):
    for item in haystack:
        if item == needle:
            return True
    return False
```

```python
def search_slow(haystack, needle):
    return_value = False
    for item in haystack:
        if item == needle:
            return_value = True
    return return_value
```

Selvom begge har runtime O løsninger, vil search_fast kjører fortere enn search_slow fordi den ikke har unødvendige beregninger som følge av å ikke avslutte loop'en tidlig.

Python vectorization er når en CPU mottar mye data på en gang, og klarer å operere alle på engang. Denne type CPU instruksjon er kjent som SIMD (single instruction, Multiple data)
I search slow versjonen så starter ikke vectorization med engang, som betyr at det tar lengre tid før PC-en kan regne ut kalkulasjonen. Derfor er search_slow funksjonen tregere en search_fast.

Det å finne mer effektive måter å gjøre de sammen beregningene, og finne unyttige operasjoner og fjerne dem vil gi samme sluttresultat men antall beregninger og dataoverføringer reduseres drastisk.

Grafene

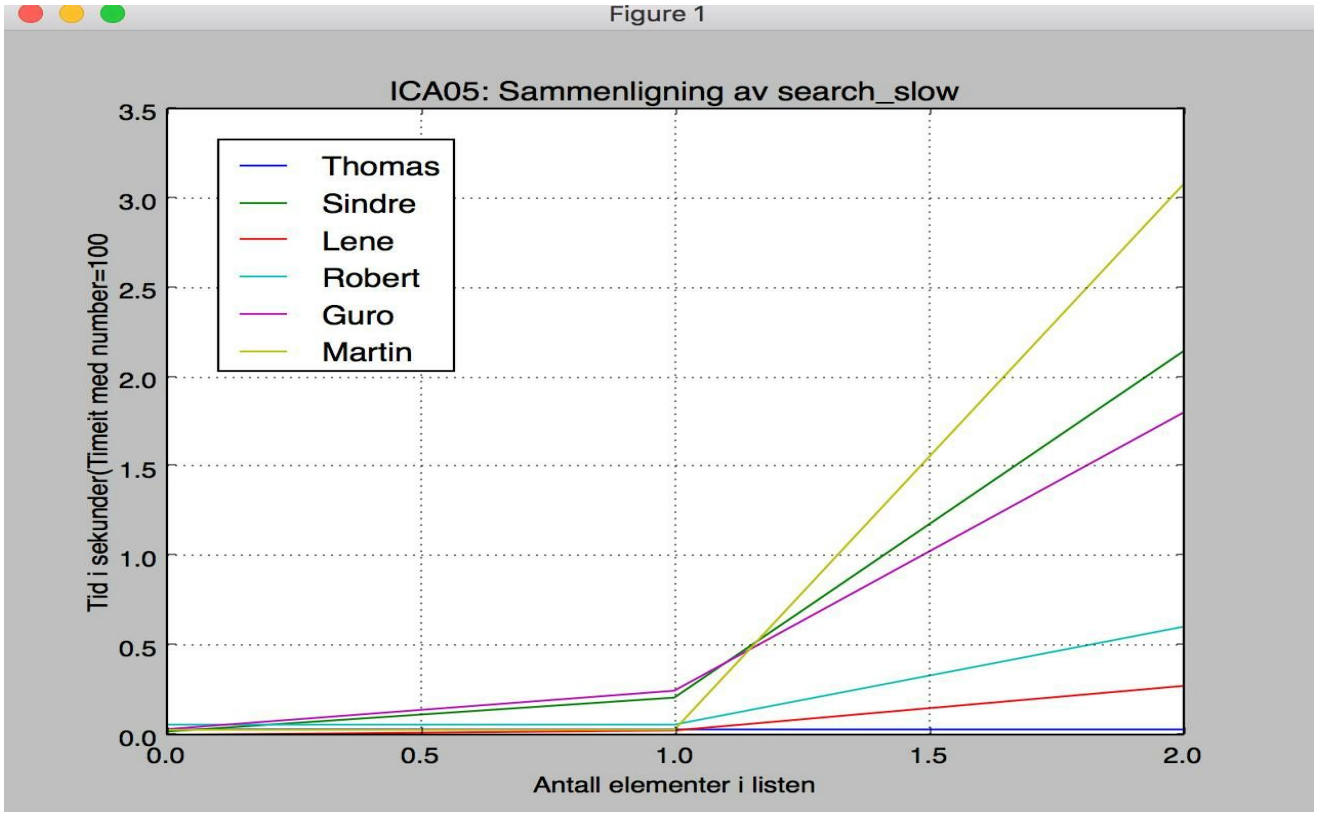
Figure 1
ICA05: Sammenligning av search_slow
Thomas
Sindre
Lene
Robert
Guro
Martin
Tid i sekunder(Timeit med number=100
Antall elementer i listen
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5

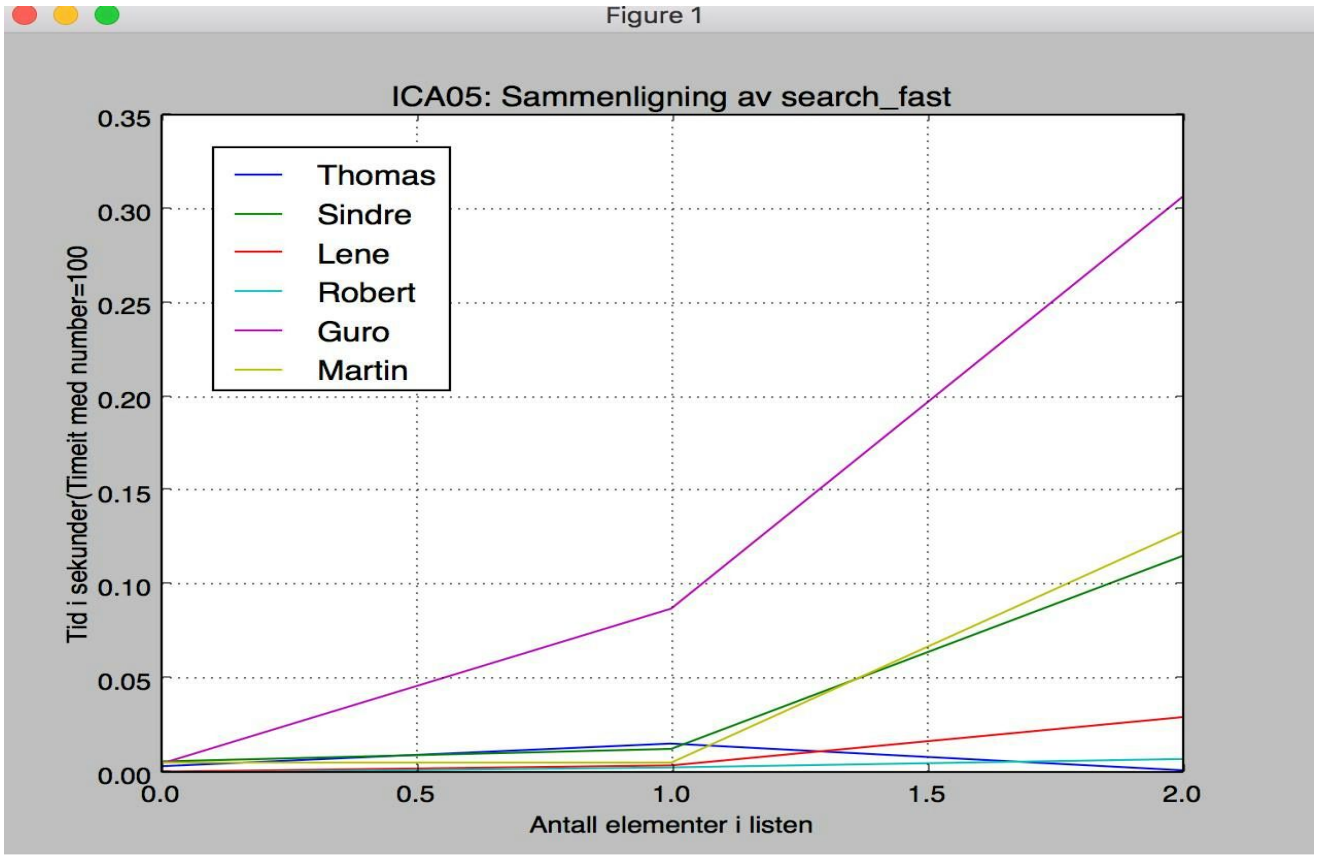
Figure 1
ICA05: Sammenligning av search_fast
Thomas
Sindre
Lene
Robert
Guro
Martin
Tid i sekunder(Timeit med number=100
Antall elementer i listen
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.0
0.5
1.0
1.5
2.0

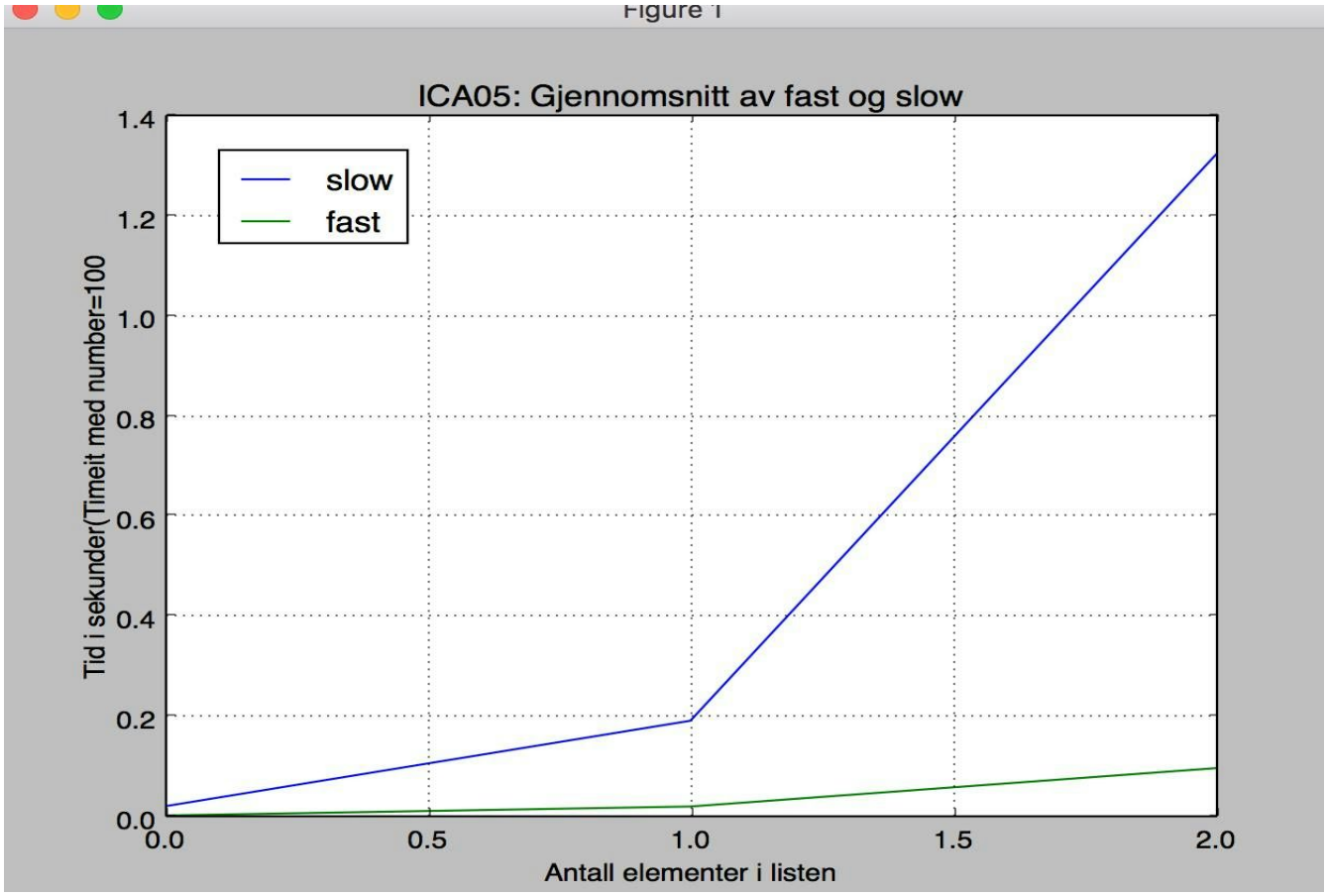
Figure 1
ICA05: Gjennomsnitt av fast og slow
slow
fast
Tid i sekunder(Timeit med number=100
Antall elementer i listen
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
0.0
0.5
1.0
1.5
2.0