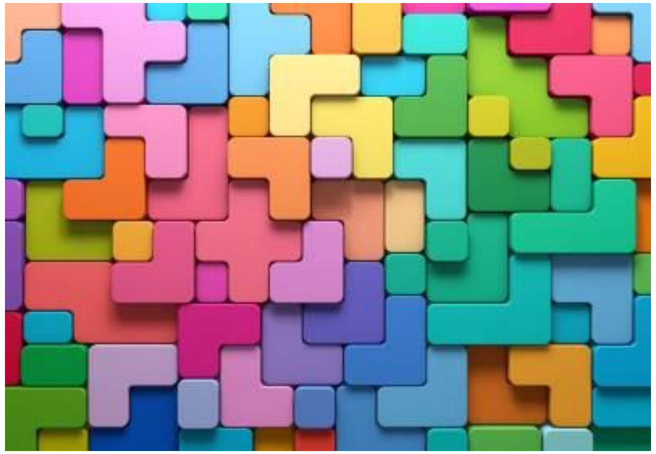
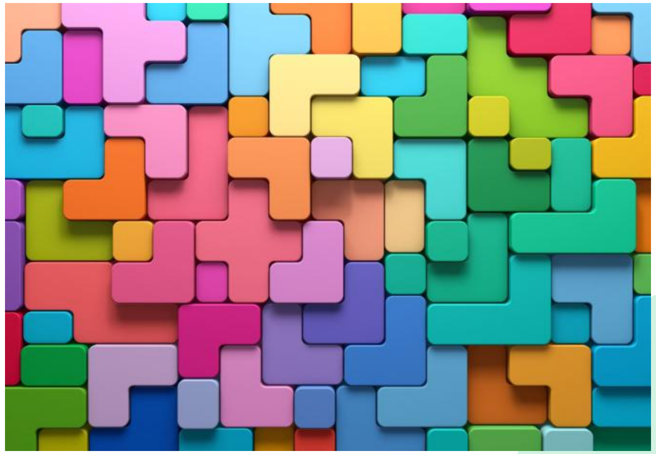
ALGORITMEANALYSE OG SMARTE PEKERE

FORELESNING 15

MANDAG 13/10





(bilder generert av bing image creator)

LIVEKODING:

SINGLY LINKED LIST

(FORTS.)

Dokumentasjon: Doxygen

- Lastes ned her
- (Hvordan bruke? Livekodes i forelesning)

Algoritmeanalyse

```
def find_biggest(numbers: list[int]) -> int:
    biggest = numbers[0]
    n = len(numbers)

    i = 1
    while i < n:
        if numbers[i] > biggest:
            biggest = numbers[i]
        i = i + 1

    return biggest
```

- Hvordan skalerer algoritmen med antall elementer?
- Se på antall operasjoner:
  - Indeksering […]
  - Tilordning =
  - Sammenligning >, <
  - Addisjon +
  - len(…)
  - Returverdi

n = 1 element

```python
def find_biggest(numbers: list[int]) -> int:
    biggest = numbers[0]
    n = len(numbers)

    i = 1
    while i < n:
        if numbers[i] > biggest:
            biggest = numbers[i]
        i = i + 1

    return biggest
```

- biggest = numbers[0]
  2 operasjoner totalt
- n = len(numbers)
  2 operasjon totalt
- i = 1
  1 operasjon totalt
- while i < n:
  1 operasjon totalt
- if numbers[i] > biggest:
  0 operasjoner totalt
- biggest = numbers[i]
  0 operasjoner totalt
- i = i + 1
  0 operasjoner totalt
- return biggest
  1 operasjon totalt
- Sum: **7**

n = 2 elementer

```python
def find_biggest(numbers: list[int]) -> int:
    biggest = numbers[0]
    n = len(numbers)

    i = 1
    while i < n:
        if numbers[i] > biggest:
            biggest = numbers[i]
        i = i + 1

    return biggest
```

- biggest = numbers[0]
  2 operasjoner totalt
- n = len(numbers)
  2 operasjoner totalt
- i = 1
  1 operasjon totalt
- while i < n:
  2 operasjoner totalt (kjører 2 ganger)
- if numbers[i] > biggest:
  2 operasjoner totalt (kjører 1 gang)
- biggest = numbers[i]
  0 - 2 operasjoner totalt (kjører 0 - 1 gang)
- i = i + 1
  2 operasjoner totalt (kjører 1 gang)
- return biggest
  1 operasjon totalt
- Sum: **12 - 14**

n = 3 elementer

```python
def find_biggest(numbers: list[int]) -> int:
    biggest = numbers[0]
    n = len(numbers)

    i = 1
    while i < n:
        if numbers[i] > biggest:
            biggest = numbers[i]
        i = i + 1

    return biggest
```

- biggest = numbers[0]
  2 operasjoner totalt
- n = len(numbers)
  2 operasjoner totalt
- i = 1
  1 operasjon totalt
- while i < n:
  3 operasjoner totalt (kjører 3 ganger)
- if numbers[i] > biggest:
  4 operasjoner totalt (kjører 2 ganger)
- biggest = numbers[i]
  0 - 4 operasjoner totalt (kjører 0 - 2 ganger)
- i = i + 1
  4 operasjoner totalt (kjører 2 ganger)
- return biggest
  1 operasjon totalt
- Sum: **17 - 21**

Hvordan scaler dette med n?

- n = 1 → 7 - 7
- n = 2 → 12 - 14
- n = 3 → 17 - 21
- n = 4 → 22 - 28

```python
def find_biggest(numbers: list[int]) -> int:
    biggest = numbers[0]
    n = len(numbers)

    i = 1
    while i < n:
        if numbers[i] > biggest:
            biggest = numbers[i]
        i = i + 1

    return biggest
```

- "Best case": $7 + 5(n - 1) = 5n + 2$
- "Worst case": $7 + 7(n - 1) = 7n$

Det går an å lese dette mer direkte

```python
def find_biggest(numbers: list[int]) -> int:
    biggest = numbers[0]
    n = len(numbers)

    i = 1
    while i < n:
        if numbers[i] > biggest:
            biggest = numbers[i]
        i = i + 1

    return biggest
```

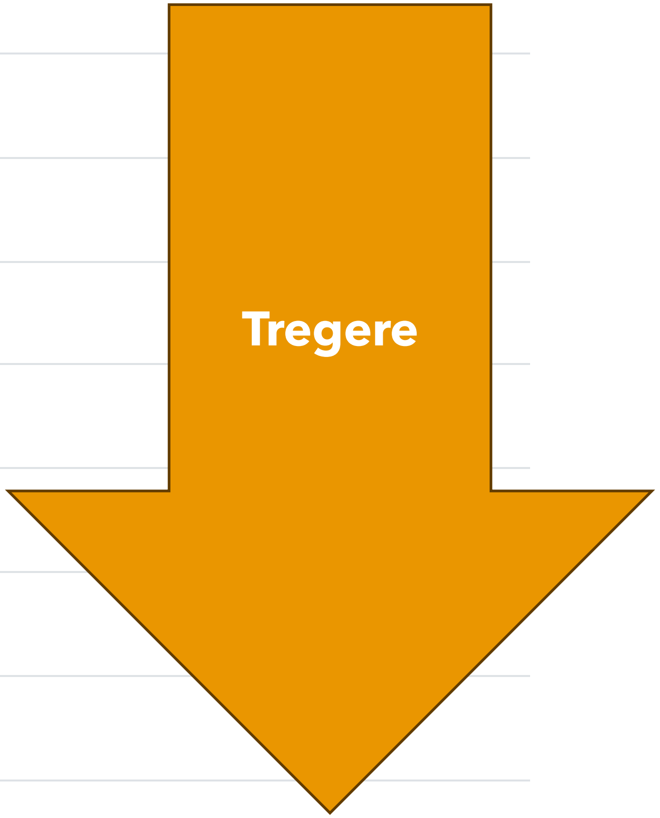
- 2
- 2
- 1
- n
- 2(n-1)
- 0 – 2(n-1)
- 2(n-1)
- 1

- "Best case": $6 + n + 4(n-1) = 5n + 2$
- "Worst case": $6 + n + 6(n-1) = 7n$

Stor-O-notasjon

- Når kjøretiden er $an + b$ har vi **lineær** scaling med n
- Leddet som dominerer når n blir stort er det vi fokuserer på
- Vi sier at skaleringen er av orden $O(n)$ for denne algoritmen

Big O	Name
$\mathcal{O}(1)$	constant
$\mathcal{O}(\log n)$	logarithmic
$\mathcal{O}(n)$	linear
$\mathcal{O}(n \log n)$	loglinear/quasilinear
$\mathcal{O}(n^2)$	Quadratic
$\mathcal{O}(n^3)$	Cubic
$\mathcal{O}(n^k)$	Polynomial
$\mathcal{O}(e^n)$	Exponential
$\mathcal{O}(n!)$	Factorial

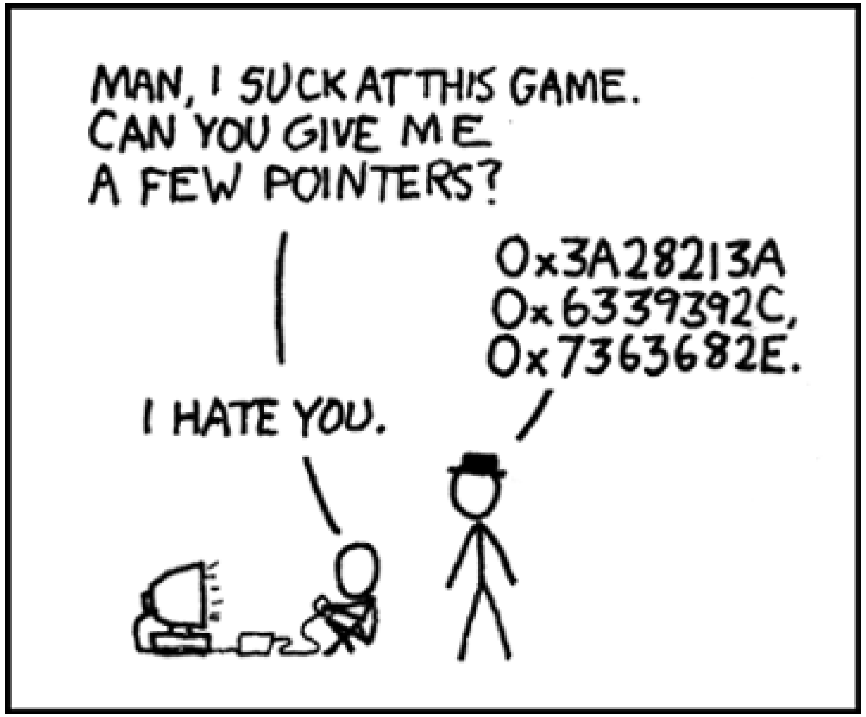


Note that an algorithm can also have fractional exponent scaling, such as $\mathcal{O}(n^{1.5})$, but this is rare.

LIVEKODING: SMARTE PEKERE

Etter forelesningen

- Fristen for å melde inn eksamensdato er ute
- Du får vite eksamensdato snart (enten av administrasjonen eller oss)
- Fortsett med gode spørsmål på Mattermost og i gruppetimene



Bilde: xkcd.com