

Algoritmer og datastrukturer

Kondisjonstest

Repetisjonsoppgaver

Onsdag 6. oktober 2004

Dette oppgavesettet er for det meste ment å drille på konsepter. Det er ikke representativt for *vanskelighetsgraden* på eksamen, men klarer du hele dette settet stiller du i alle fall sterkt! :-)

Disse repetisjonsoppgavene er ganske mange. Det er derfor mye å lese gjennom, men vi håper du blir tilsvarende drillet i dette stoffet.

Hvis du ikke rekker gjennom alle, er det jo bare å kose seg en annen gang :-)

PS! Bakerst er det med formelark.

Oppgave 1 - Designmetoder

Deloppgave 1a

Alles favoritt, tegn streker! Dra streker mellom designmetode og algoritmer som hovedsaklig benytter seg av designmetoden. Det skal være to streker fra hver designmetode, med mindre du velger å se på algoritmene på en annen måte en vi har valgt i LF. Det er ikke nødvendigvis åpenbart hvilken designmetode som passer best til å beskrive en algoritme.

Rå makt (brute force)	Flettesortering (merge sort)
Splitt-og-hersk (divide and conquer)	Sekvensielt søk (sequential search)
Krymp-og-hersk (decrease and conquer)	Dybde-først-søk (depth-first search)
	Boblesortering (bubble sort)
	Topologisk sortering (topologic sort)
	Binærsøk

Deloppgave 1b

Stryk det som ikke er en designmetode. Tenk kort gjennom hva som kjennetegner hver designmetode. (Du får velge selv om du vil tenke på de designmetodene som ikke er gjennomgått ennå)

- Forgren-og-begrens (branch-and-bound)
- Splitt-og-hersk (divide and conquer)
- Kombinér eksponensvekst (combine exponentiation)
- Endre-og-hersk (transform and conquer)
- Tilbakesporing (backtracking)
- Grådighet (greedy technique)
- Rå makt (brute force)
- Krymp-og-hersk (decrease and conquer)
- Dynamisk programmering (dynamic programming)
- Tid-plass-avveining (space-time tradeoffs)

Deloppgave 1c

Hvilken av designteknikkene rå makt, splitt-og-hersk og krymp-og-hersk brukes i denne koden?

```
A = [2, 5, 1, 7, 9, 20, 10, 3]

def finn(A, tall):
    print A
    if len(A) == 0:
        return None
    elif len(A) == 1:
        if A[0] == tall:
            return 0
        else:
            return None
    else:
        midt = len(A) // 2
        res1 = finn(A[:midt], tall)
        res2 = finn(A[midt:], tall)

        if res1 != None:
            return res1
        else:
            if res2 != None:
                return midt + res2
            else:
                return None

i = finn(A, 7)
if i == None:
    print "Fant ikke tallet"
else:
    print "Funnet i posisjon", i
```

Deloppgave 1d

Enn denne? Her er det ikke et entydig svar.

```
A = [12, 3, 2, 1, 0, 8, 4]
finn = 2
for i in range(len(A)):
    if A[i] == finn:
        print "Funnet i posisjon", i
```

Deloppgave 1e

Og denne?

```
A = [1, 5, 3, 0, 8, 6]
B = []
for X in A:
    if len(B) == 0:
        B = [X]
    else:
        i = 0
        for Y in B:
            if X < Y:
                B.insert(i, X)
                X = None
```

```

        break
    i += 1
    if X != None:
        B.append(X)

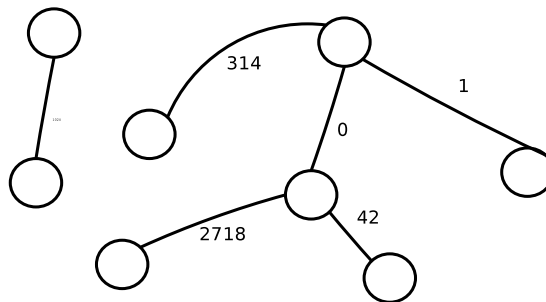
print B

```

Oppgave 2 - Datastrukturer

I denne oppgaven skal du finne navnet på forskjellige datastrukturer. En del av strukturene har også en del lett gjenkjennelige egenskaper. Se om du kan finne alle som nevnes i LFet :)

Deloppgave 2a



Deloppgave 2b

```

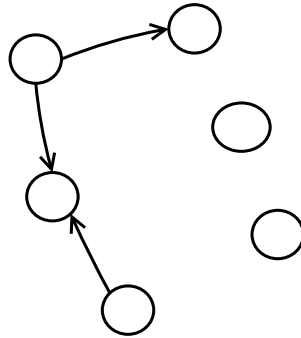
class struct:
    def __init__(self):
        self.values = [None]
        self.i = -1
        self.n = 1

    def add(self, object):
        self.i += 1
        if self.i == self.n:
            self.n *= 2
            new_values = [None] * self.n
            for j in range(0, self.i):
                new_values[j] = self.values[j]
            self.values = new_values
        self.values[self.i] = object

    def remove(self):
        object = self.values[self.i]
        self.i -= 1
        if self.i < self.n // 4:
            self.n //= 2
            new_values = [None] * self.n
            for j in range(0, self.i + 1):
                new_values[j] = self.values[j]
            self.values = new_values
        return object

```

Deloppgave 2c



Deloppgave 2d

```
size = 23

class struct:
    def __init__(self):
        self.keys = [None] * size
        self.values = [None] * size

    def add(self, key, object):
        i = calc(key)
        if i >= 0:
            if self.keys[i] == None:
                self.keys[i] = [key]
                self.values[i] = [object]
            else:
                for j in range(0, len(self.keys[i])):
                    if self.keys[i][j] == key:
                        self.values[i][j] = object
                        break
                else:
                    self.keys[i].append(key)
                    self.values[i].append(object)

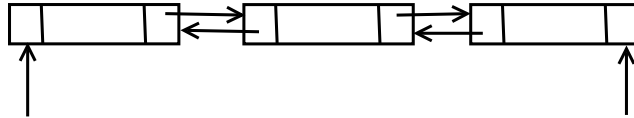
    def retrieve(self, key):
        i = calc(key)
        if i >= 0:
            if self.keys[i] == None:
                return None
            else:
                for j in range(0, len(self.keys[i])):
                    if self.keys[i][j] == key:
                        return self.values[i][j]

    def calc(key):
        if key.__class__ == (1).__class__:
            i = key % size
        elif key.__class__ == "a".__class__:
            i = 0
            c = 1
            for d in key:
                i += ord(d) # ord('a') = 97, chr(ord(d)) = d, chr(97) = 'a'
            i %= size
        else:
            print "Unsupported type"
            return -1
        return i
```

Deloppgave 2e

`[[0, 1, 0, 1], [1, 1, 1, 0], [0, 1, 1, 0], [0, 0, 0, 1]]`

Deloppgave 2f



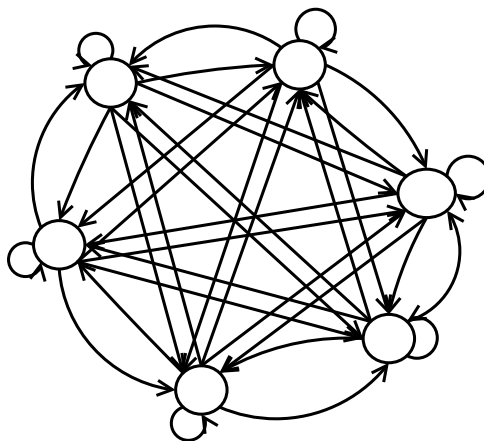
Deloppgave 2g

```
class struct:
    def __init__(self):
        self.values = []

    def add(self, object):
        self.values.append(object)

    def remove(self):
        object = self.values[0]
        self.values = self.values[1:]
        return object
```

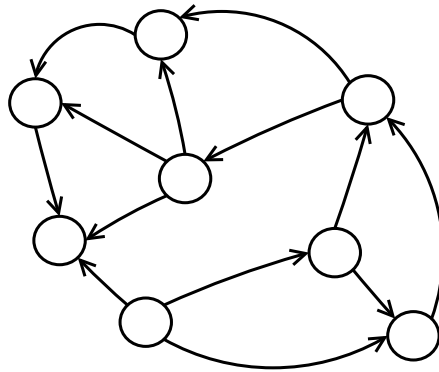
Deloppgave 2h



Deloppgave 2i

`[1, 2, 0, 13, 26]`

Deloppgave 2j



Oppgave 3 - Algoritmer

Deloppgave 3a

Fyll ut tabellen. Angi kompleksiteten (Kompl.) for worst case, i asymptotisk notasjon. Bruk den beste worst case-kompleksiteten fra boka, dersom det finnes flere versjoner av algoritmen.

I problem-kolonnen skal du fylle ut hva slags type problemer algoritmen løser. Denne typen er en eller flere av:

- Sortering (sorting)
- Søking (searching)
- Strengprosessering (string processing)
- Grafproblemer (graph problems)
- Kombinatorisk problem (combinatorial problem)
- Geometrisk problem (geometric problem)
- Numerisk problem (numerical problem)

Algoritme	Designmetode	Kompl.	Problem
Quicksort			
SequentialSearch2			
Binary search			
BruteForceStringMatch	Brute force		
Selection sort			
BFS			
Mergesort			
BruteForceClosestPoints	Brute force		
Bubble sort			
Insertion sort			
DFS			

Deloppgave 3b

Hvilken algoritme er dette?

```
def algoritme(G):
    (V, E) = G
    Vt = [V[0]]
    Et = []
    E.sort()
    for i in range(1, len(V)):
        for (w, (v, u)) in E:
            if v in Vt and u not in Vt:
                Vt.append(u)
                Et.append((w, (v, u)))
                break
    return Et
```

Deloppgave 3c

Hvilken algoritme er dette?

```
count = 0

def algoritme(G):
    global count
    count = 0
    (V, E) = G
    for v in V:
        v.m = 0
    for v in V:
        if v.m == 0:
            funksjon(V, E, v)

def funksjon(V, E, v):
    global count
    count += 1
    v.m = count
    q = [v]
    while len(q) > 0:
        v = q.pop(0)
        for w in V:
            if w.m == 0 and (v, w) in E:
                count += 1
```

```
w.m = count
q.append(w)
```

Oppgave 4 - Analyse

Deloppgave 4a

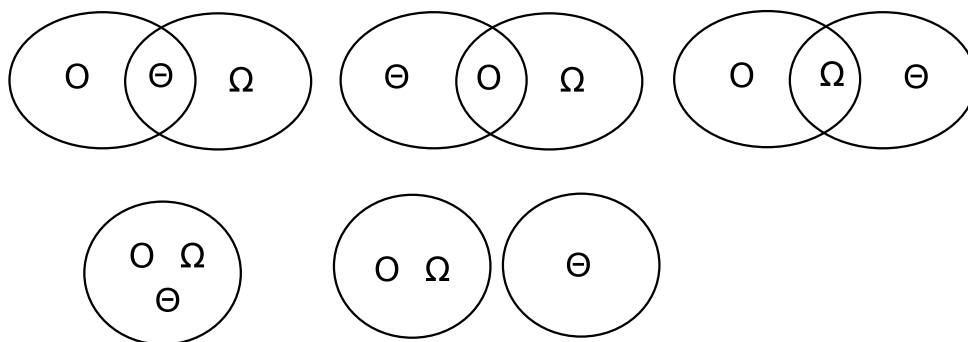
Beskriv 1000 forskjellige funksjoner som er med i $\Omega(n)$.

Deloppgave 4b

Beskriv 1000 forskjellige funksjoner som er med i $\Theta(n)$.

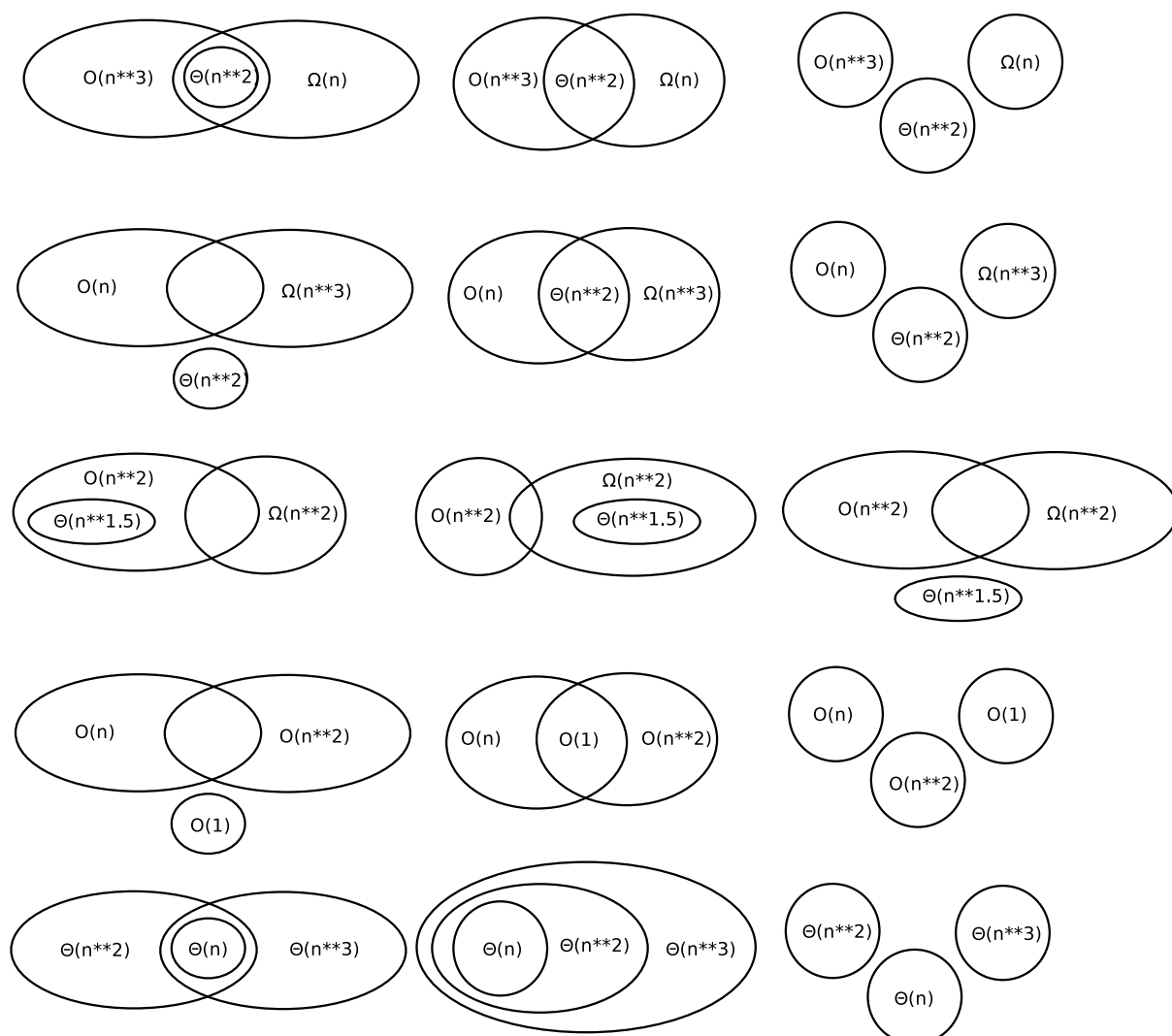
Deloppgave 4c

Hvilken figur er riktig?



Deloppgave 4d

Hvilke figurer er riktige?

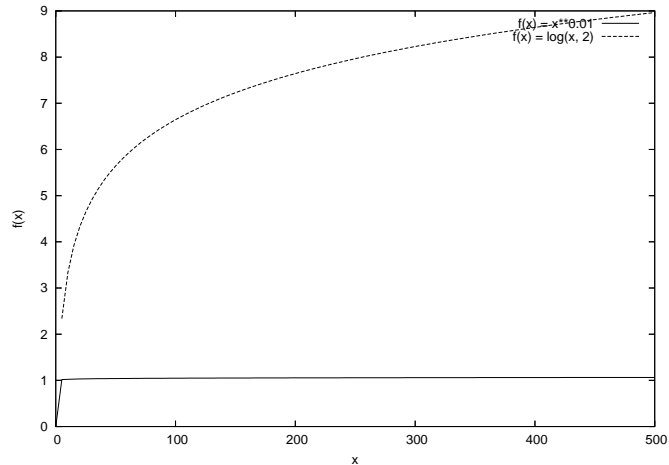


Deloppgave 4e

Noen kjappe ja/nei-spørsmål:

Er $O(n^{1.000000001}) \subseteq O(n)$?

Er $O(n^{0.01}) \subseteq O(\log n)$? Merk at nå er n opphøyd i noe som er mindre enn 1. Dette er grafen for små n :



Deloppgave 4f

Gi eksempel på en funksjon som er...

- konstant (constant)
- logaritmisk (logarithmic)
- lineær (linear)
- polynomisk (polynomial)
- n-log-n (n-log-n)
- kvadratisk (quadratic)
- kubisk (cubic)
- eksponensiell (exponential)
- faktoriell (factorial)

Deloppgave 4g

Beskriv uendelig mange polynomiske funksjoner vha. 4 tegn.

Oppgave 5 - Rekurrenser

Dette er en blåkopi av rekurrensoppgavene fra oppgave 2 på øving 4. Eneste unntak er deloppgave c, som er gjort litt vanskeligere. Finn nøyaktige svar av rekurrensene i hver deloppgave ved hjelp av backward substitution og noen av formlene fra formelarket.

Hint: Ikke løs flere oppgaver når backward substitution og de tilhørende matematiske triksene sitter i fingrene :)

Deloppgave 5a

$$T(n) = T(n/2) + 1, T(1) = 0, \text{ der } n = 2^i, i \in \{0, 1, 2, \dots\}$$

Deloppgave 5b

$$T(n) = 2T(n/2) + 1, T(1) = 0, \text{ der } n = 2^i, i \in \{0, 1, 2, \dots\}$$

Deloppgave 5c

$$T(n) = 2T(n/3) + 1, T(1) = 1, \text{ der } n = 3^i, i \in \{0, 1, 2, \dots\}$$

Deloppgave 5d

$$T(n) = 3T(n/3) + 1, T(1) = 0, \text{ der } n = 3^i, i \in \{0, 1, 2, \dots\}$$

Deloppgave 5e

$$T(n) = T(n/2) + n, T(1) = 0, \text{ der } n = 2^i, i \in \{0, 1, 2, \dots\}$$

Deloppgave 5f

$$T(n) = 2T(n/2) + n, T(1) = 0, \text{ der } n = 2^i, i \in \{0, 1, 2, \dots\}$$

Oppgave 6 - Master-teoremet

Her kommer en del av de samme rekurensene en gang til. Nå skal oppgavene løses vha. *Master-teoremet*. Dersom Master-teoremets krav ikke er oppfylt, skal du svare at rekurrensen ikke kan løses vha. Master-teoremet, og løse rekurrensen ved hjelp av backward substitution.

Deloppgavene som ikke lar seg løse med masterteoremet, krever ganske høy konsentrasjon for å løses med backward substitution. Det er godt mulig du kan bruke tiden på mer fornuftige ting enn å finne den nøyaktige løsningen på disse.

Deloppgave 6a

$$T(n) = T(n/2) + 1, T(1) = 0, \text{ der } n = 2^i, i \in \{0, 1, 2, \dots\}$$

Deloppgave 6b

$$T(n) = 2T(n/2) + 1, T(1) = 0, \text{ der } n = 2^i, i \in \{0, 1, 2, \dots\}$$

Deloppgave 6c

$$T(n) = 2T(n/3) + 1, T(1) = 1, \text{ der } n = 3^i, i \in \{0, 1, 2, \dots\}$$

Deloppgave 6d

$$T(n) = 2T(n/2) + n \log n, T(1) = 0, \text{ der } n = 2^i, i \in \{0, 1, 2, \dots\}$$

Deloppgave 6e

$$T(n) = 3T(n/3) + 1, T(1) = 0, \text{ der } n = 3^i, i \in \{0, 1, 2, \dots\}$$

Deloppgave 6f

$$T(n) = T(n/2) + n, T(1) = 0, \text{ der } n = 2^i, i \in \{0, 1, 2, \dots\}$$

Deloppgave 6g

$$T(n) = 2T(n/2) + n, T(1) = 0, \text{ der } n = 2^i, i \in \{0, 1, 2, \dots\}$$

Deloppgave 6h

$$T(n) = 2T(n-2) + n, T(0) = 0, \text{ der } n \in \{0, 1, 2, \dots\}$$

Oppgave 7 - Kjøretid - Finn formel

I denne oppgaven skal vi se på den asymptotiske oppførselen til noen enkle funksjoner. Du skal gi rekurensen (eller en degenerert rekurens) for funksjonen `funksjonX` av n med hensyn på basisoperasjonen, `gjorKonst()` eller `gjorNoe()`. Gi et så presist svar som mulig, men ikke løs ut evt. rekurrenser.

I neste oppgave skal du løse evt. rekurrenser, så da får du se om du har gjort riktig på noen av oppgavene her. Ikke tjuvtitt :-)

Anta at funksjonen `gjorKonst()` har kjøretid $\Theta(1)$ og at funksjonen `gjorNoe()` har kjøretid $O(n)$.

For å gjøre ting pent, anta at $n = 2^i, i \in 1, 2, 3, \dots$

Deloppgave 7a

```
def funksjonA(n):
    for i in range(0, n):
        for j in range(0, n):
            gjorKonst()
```

Deloppgave 7b

```
def funksjonB(n):
    sum = 0
    if n > 1:
        sum += funksjonB(n//2)
        sum += funksjonB(n//2)
        for i in range(n):
            sum += i
            gjorKonst()

    return sum
```

Deloppgave 7c

```
from random import randint

def funksjonC(n):
    tall = [0] * n
    for i in range(0, n):
        tall[i] = randint(1, n)
        gjorKonst()
```

```

for i in range(0, n):
    for j in range(0, n):
        maks = 0
        for k in tall:
            tmp = abs(tall[j] - k)
            if tmp > maks:
                maks = tmp
            gjorKonst()
        tall[j] = maks

```

Deloppgave 7d

```

def funksjonD(n):
    if n > 0:
        funksjonD(n-1)
    gjorNoe()

```

Deloppgave 7e

Hint: Bare sett opp rekurensen for worst case-kjøretid. Hvis denne kan løses og gi et svar med Θ -notasjon, er det bare å endre dette svaret til å bruke O -notasjon for å få en generell kjøretid.

```

from random import randint

def funksjonE(n):
    tall = randint(0, n - 1)
    print finn(tall, 0, n - 1)

def finn(tall, nedre, ovre):
    gjorKonst()
    gjett = (ovre + nedre) // 2
    if gjett == tall:
        return gjett
    elif gjett > tall:
        return finn(tall, nedre, gjett - 1)
    else:
        return finn(tall, gjett + 1, ovre)

```

Oppgave 8 - Kjøretid - Løs rekurens

I denne oppgaven skal vi løse rekurensene fra hhv. deloppgaver b, d og e i forrige oppgave. Bruk den metoden du liker best :)

Deloppgave 8a

$$T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n), T(1) = 0$$

Deloppgave 8b

$$T(n) = T(n - 1) + O(n), T(1) = O(n)$$

Deloppgave 8c

$$T(n) \leq T(n/2) + \Theta(1), T(1) = O(1)$$

(endret i forhold til utleverte oppgaver).

Oppgave 9 - Ja/Nei

Er den asymptotiske kjøretiden til en algoritme avhengig av datastrukturen som brukes?

Brukes fysiske datastrukturer til å implementere abstrakte datatyper?

Kan en graf inneholde flere trær?

Kan et binærtre være ei lenket liste?

Kan en graf ha flere kanter enn noder?

Kan et tre ha flere kanter enn noder?

Er $n^2 \in O(n)$?

Er $n^2 \in \Omega(n)$?

Er $n^2 \in \Theta(n)$?

Er $3^n \in O(2^n)$?

Er $n! \in O(2^n)$?

Formler

Her følger noen formler dere kan få bruk for på denne kondisjonstesten og ellers i livet. Du kan anse dette som en sketch på hva som forventes på eksamen, men vi ønsker *ikke* å binde oss ved f.eks. å love at den anbefalte kunnskapen vil bli oppgitt ved behov.

Forventet kunnskap

NB! Denne listen er ikke nødvendigvis¹ komplett.

$$\log x^y = y \log x$$

$$\log xy = \log x + \log y$$

$$\log \frac{x}{y} = \log x - \log y$$

$$\log_b x = \frac{\log_a x}{\log_a b} \text{ (eks. } \log_2 x = \frac{\log_e x}{\log_e 2} \text{)}$$

$$a^{\log_a x} = x^{\log_a a} \text{ (eks. } 2^{\log_2 n} = n \text{)}$$

$$\frac{a^n}{b^n} = \left(\frac{a}{b}\right)^n \text{ (eks. } \frac{1}{2^n} = \left(\frac{1}{2}\right)^n \text{)}$$

$$\sqrt{a \cdot b} = \sqrt{a} \cdot \sqrt{b} \text{ (eks. } x \cdot \sqrt{y} = \sqrt{x^2 \cdot y} \text{)}$$

$$\begin{aligned} \sum_{i=1}^n i \cdot x &= x \cdot \sum_{i=1}^n i \\ \sum_{i=1}^n (a_i + b_i) &= \sum_{i=1}^n a_i + \sum_{i=1}^n b_i \\ \sum_{i=1}^n i &= \sum_{i=1}^k i + \sum_{i=k+1}^n i, \text{ der } i \leq k < n \\ \text{(eks. } \sum_{i=1}^n (n-i) &= \sum_{i=0}^n (n-i) + n \text{)} \\ \sum_{i=k}^{l-1} c &= (l-k) \cdot c \end{aligned}$$

$$\begin{aligned} \sum_{i=1}^n i &= \frac{n(n+1)}{2} \approx \frac{1}{2}n^2 \\ \sum_{i=0}^n 2^i &= 2^{n+1} - 1 \end{aligned}$$

Master-teoremet:

Dersom

$$T(n) = aT(n/b) + \Theta(n^d), \quad n = b^k, \quad k = 1, 2, \dots,$$

$$T(1) = c,$$

der $a \geq 1$, $b \geq 2$, $c > 0$, $d \geq 0$ så

$$T(n) \in \begin{cases} \Theta(n^d) & \text{hvis } a < b^d \\ \Theta(n^d \log n) & \text{hvis } a = b^d \\ \Theta(n^{\log_b a}) & \text{hvis } a > b^d \end{cases}$$

Anbefalt kunnskap

$$a^{\log_b x} = x^{\log_b a}$$

$$\sum_{i=1}^n i^2 = \frac{n(n+1)(2n+1)}{6} \approx \frac{1}{3}n^3$$

$$\sum_{i=0}^n a^i = \frac{a^{n+1}-1}{a-1}, \text{ der } a \neq 1$$

Kursorisk:

$$\sum_{i=1}^n i2^i = (n-1)2^{n+1} + 2$$

¹Bonuskunnskap: "ikke nødvendigvis" \neq "nødvendigvis ikke" (spør Trygve om en eventuell utdyping av dette).