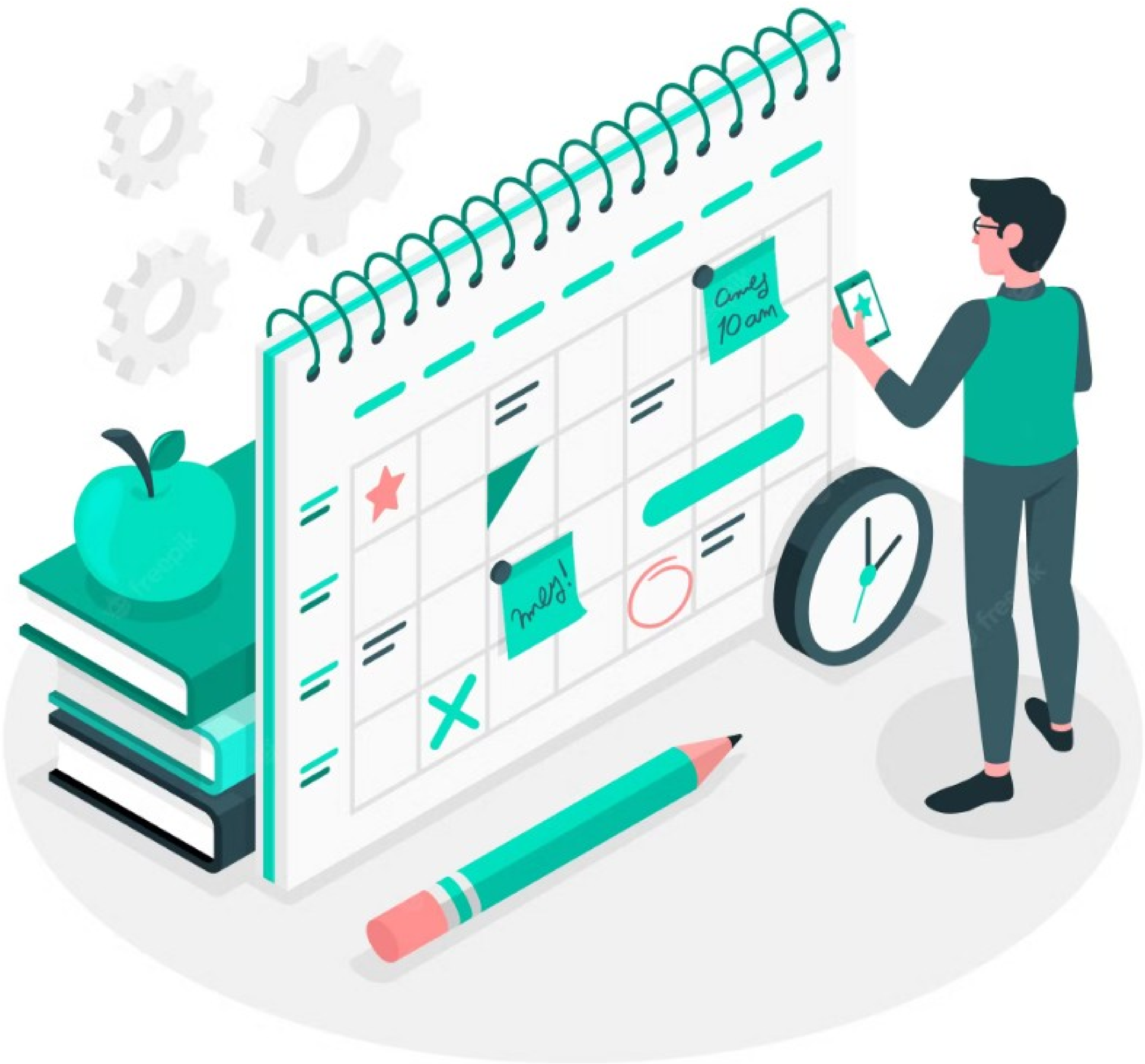
IN2010 Gruppe 5

Amadu Swaray

Bli Med!

19

Mentimeter



Dagens plan

- Praktisk info
- Introduksjon
- Bli kjent
- Pensumgjennomgang
- Gruppe Oppgaver

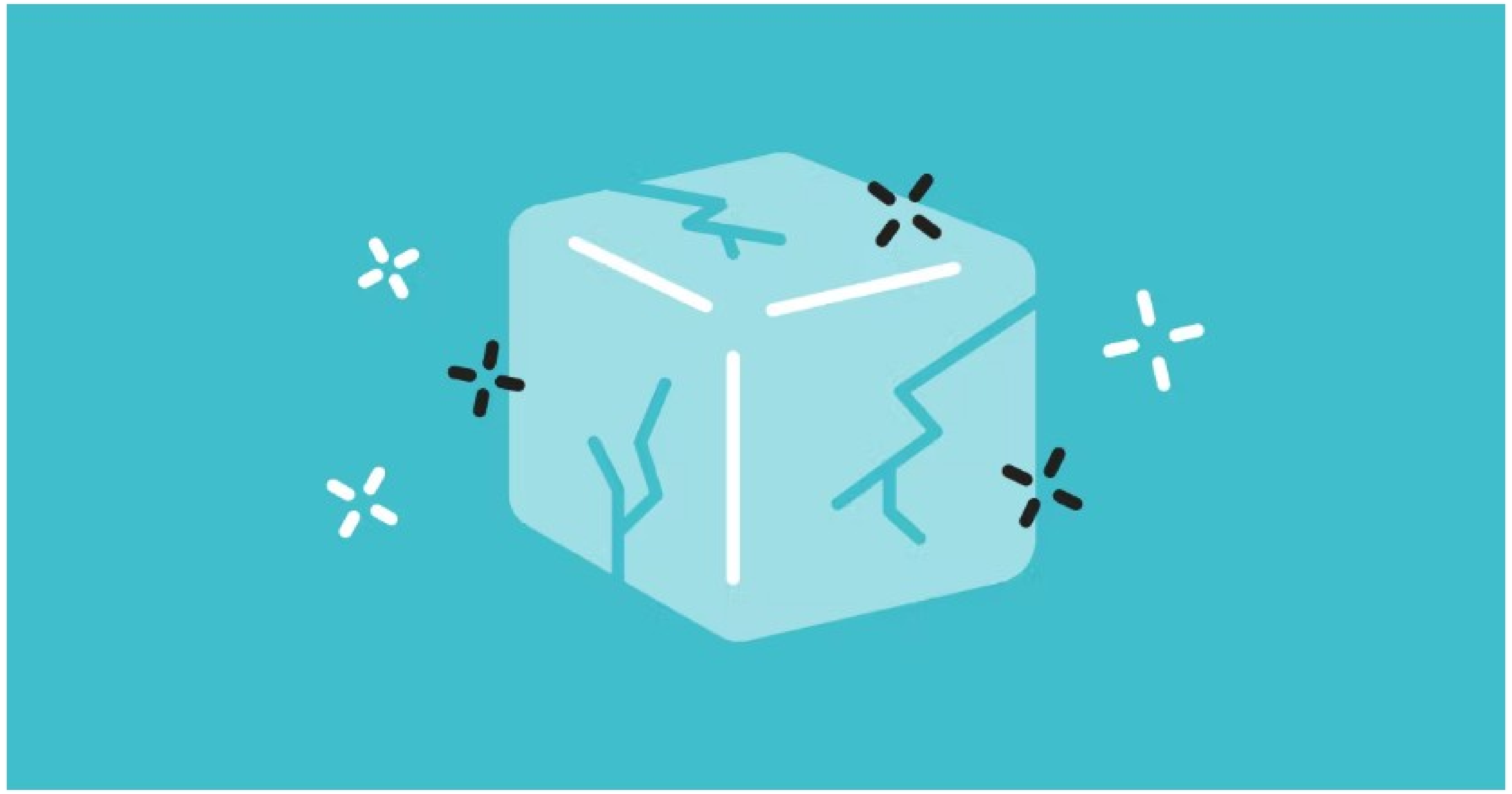
2

Praktisk Info

- E-Post: amadus@ifi.uio.no
- Forum: Astro-Discourse
- 3 Obligatoriske oppgaver
- Frist: fredag 16. september kl. 23:59
- Frist: fredag 14. oktober kl. 23:59
- Frist: fredag 4. november kl. 23:59

Hvem er jeg?

- → 4. Året prosa
- → Trønder
- → Musikk og Mat



Ice Breaker

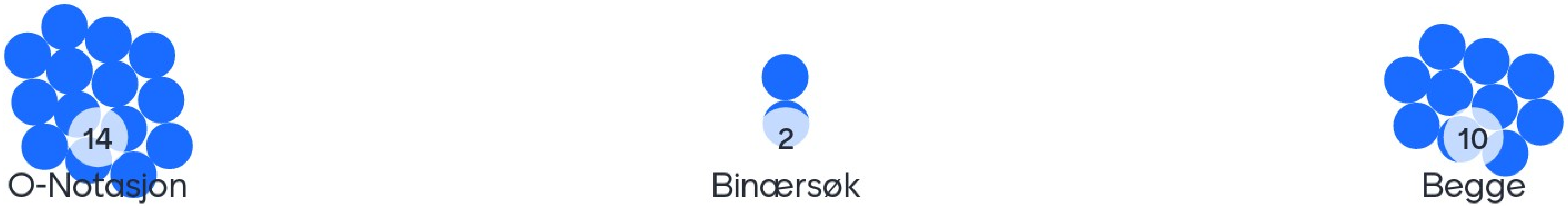
Gå sammen i grupper på 3-4 pers
Introdusere dere til hverandre ved disse spørsmålene:
Navn?
Studie?
Favoritt rett?
Favoritt film/serie?
2 min per pers.

Mentimeter

Hva skal vi gjøre i gruppetimene

1. Diskutere
2. Jobbe med oppgaver
3. Lære

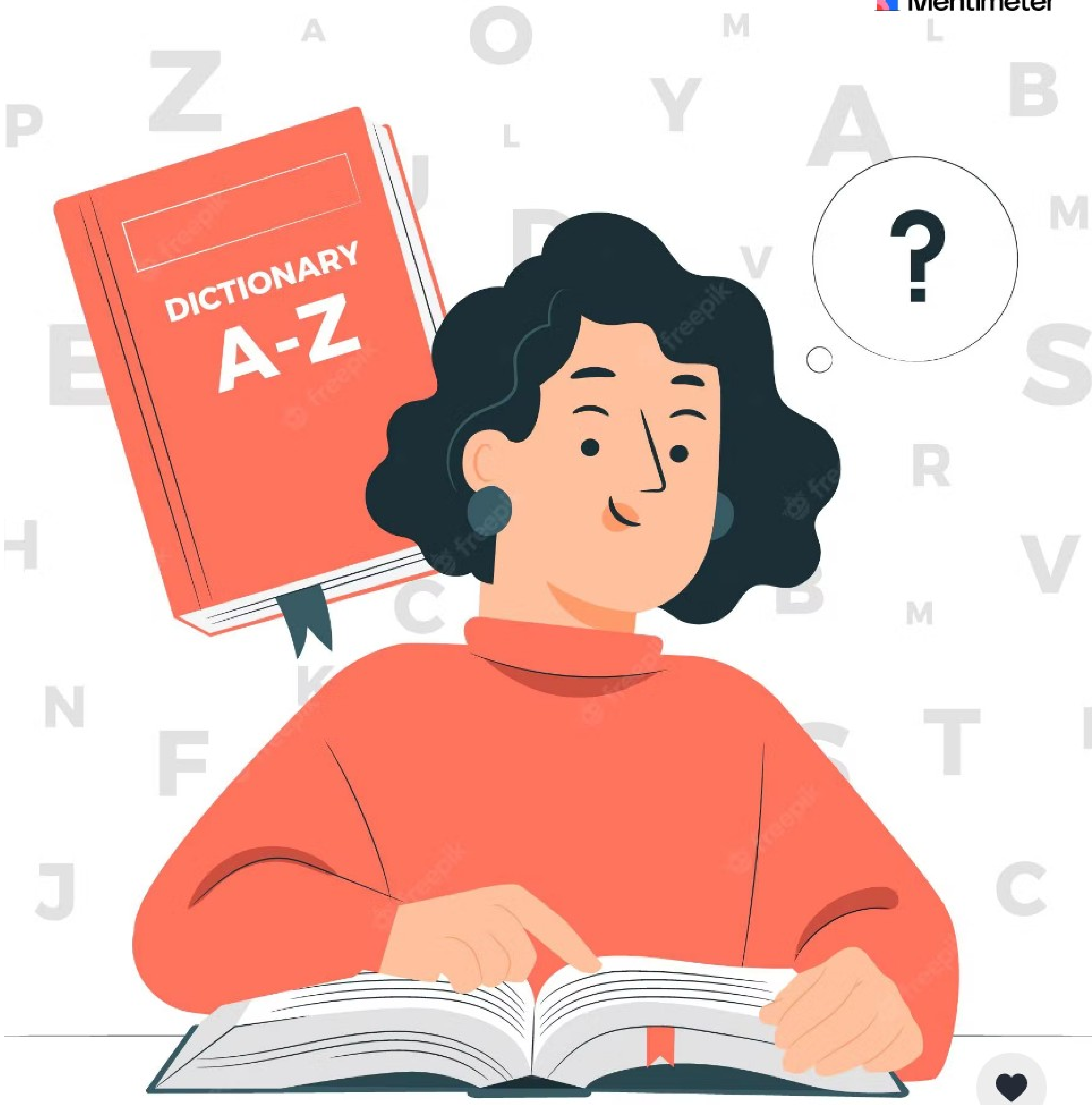
Hva er det mest behov å gjennomgå?

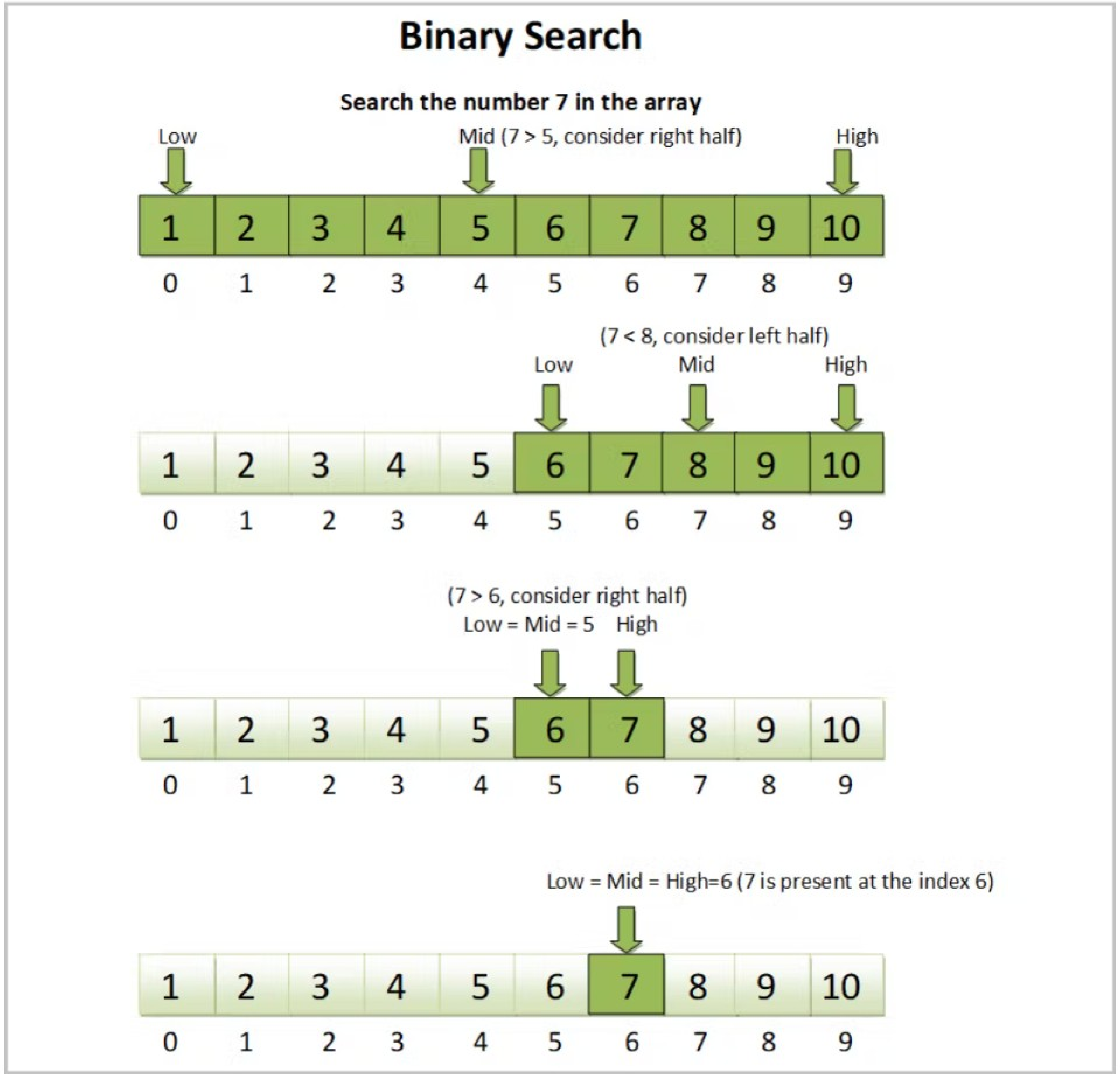


26

Binærsøk

Ide: Slå opp ord i en ordbok



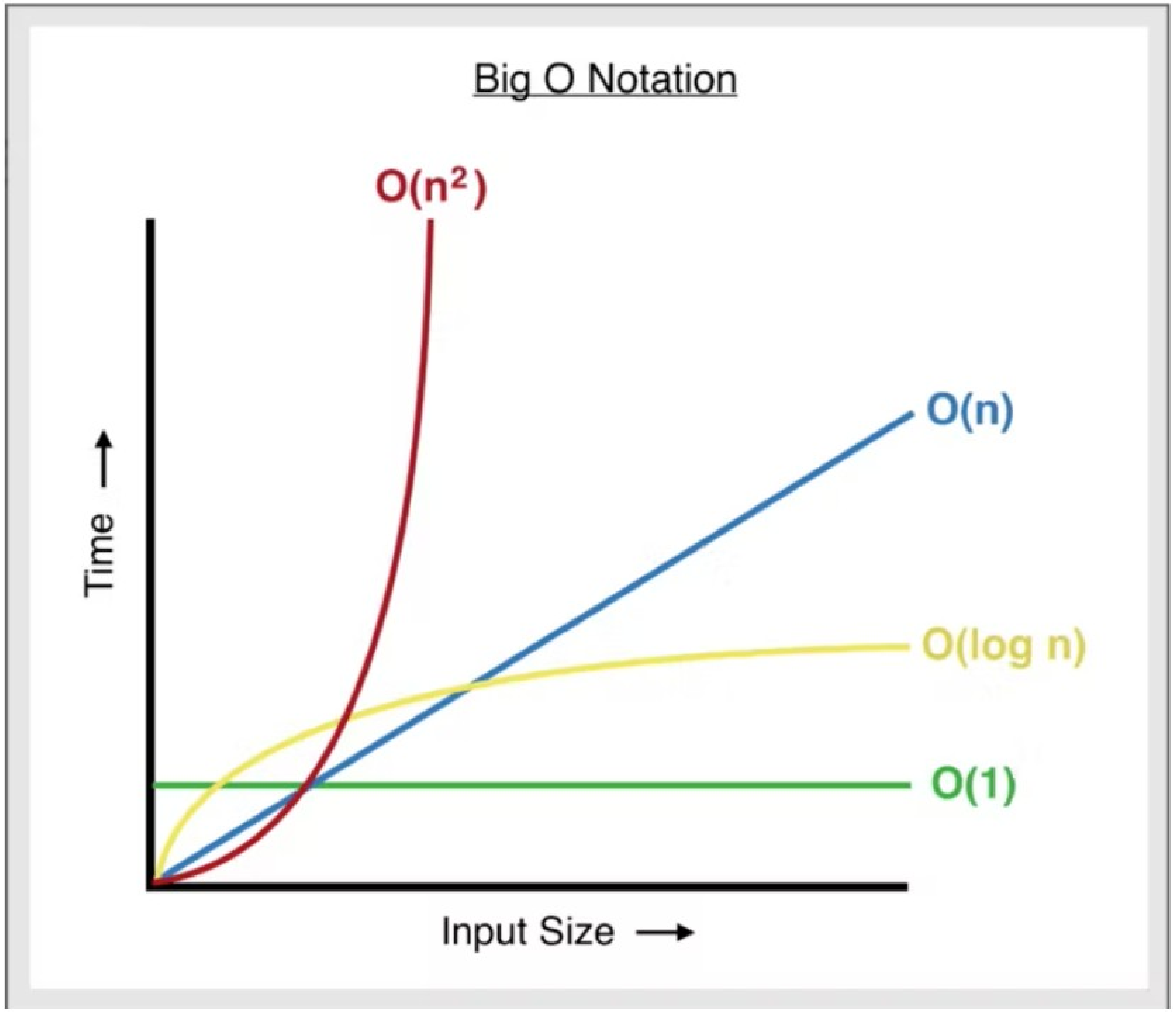


Binary Search
Search the number 7 in the array
Low
Mid (7 > 5, consider right half)
High
1 2 3 4 5 6 7 8 9 10
0 1 2 3 4 5 6 7 8 9
(7 < 8, consider left half)
Low
Mid
High
1 2 3 4 5 6 7 8 9 10
0 1 2 3 4 5 6 7 8 9
(7 > 6, consider right half)
Low = Mid = 5
High
1 2 3 4 5 6 7 8 9 10
0 1 2 3 4 5 6 7 8 9
Low = Mid = High=6 (7 is present at the index 6)
1 2 3 4 5 6 7 8 9 10
0 1 2 3 4 5 6 7 8 9

ALGORITHM: BINÆRSØK

Input: Et ordnet array A og et element x
Output: Hvis x er i arrayet A, returner `true` ellers `false`

```
Procedure BinarySearch(A, x)
    low ← 0
    high ← |A| − 1
    while low ≤ high do
        i ← ⌊(low+high)/2⌋
        if A[i] = x then
            return true
        else if A[i] < x then
            low ← i + 1
        else if A[i] > x then
            high ← i − 1
    return false
```



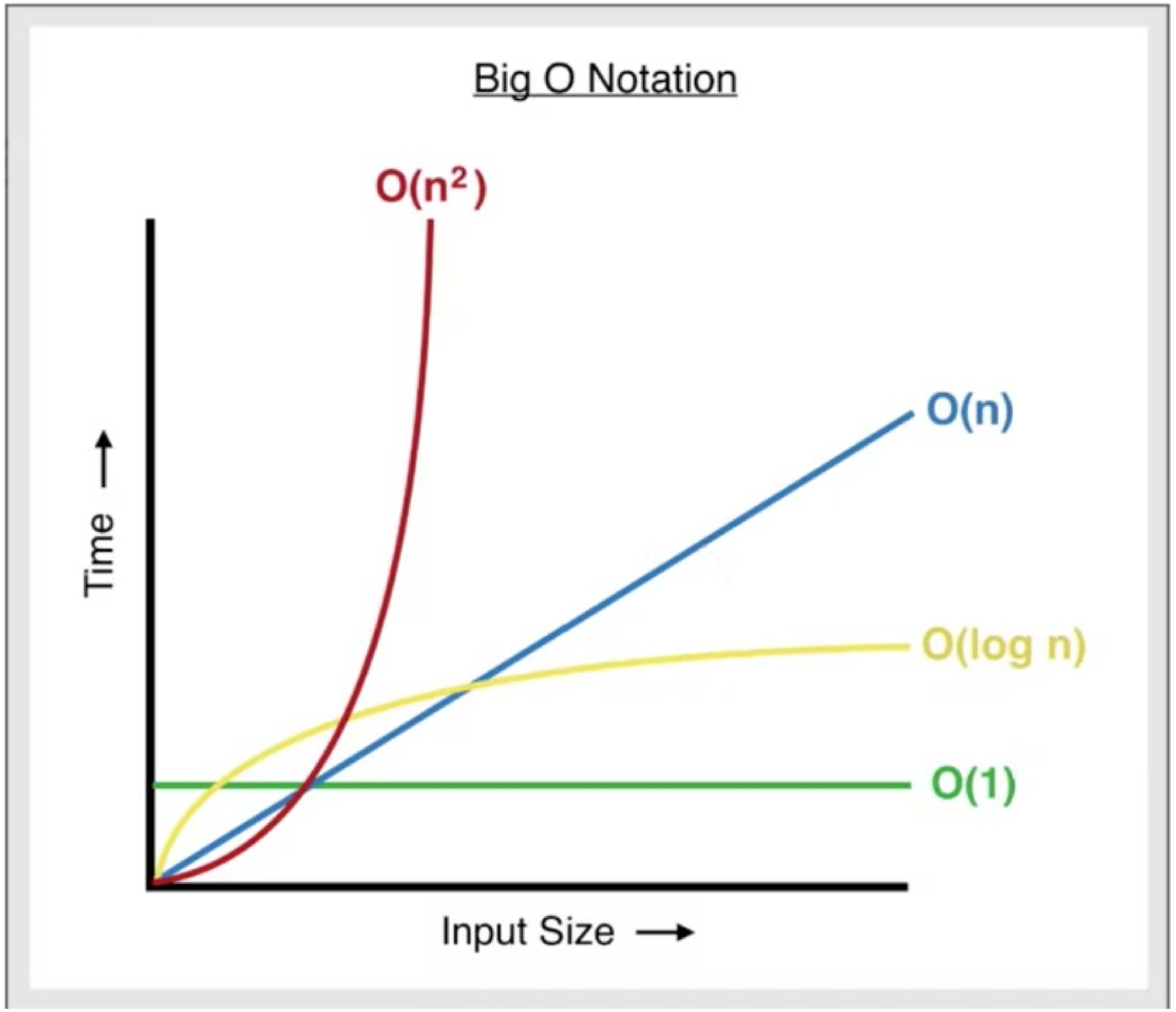
O-Notasjon

- Hvor vanskelig er et problem/algoritme å løse?
- AKA: Hvor mange steg trenger man for å løse problemet?
- Eksempel: Hvor mange steg må man gjøre for å finne en vilkårlig person i klassen?(Rett-Frem søk)

Mentimeter

$O(n)$

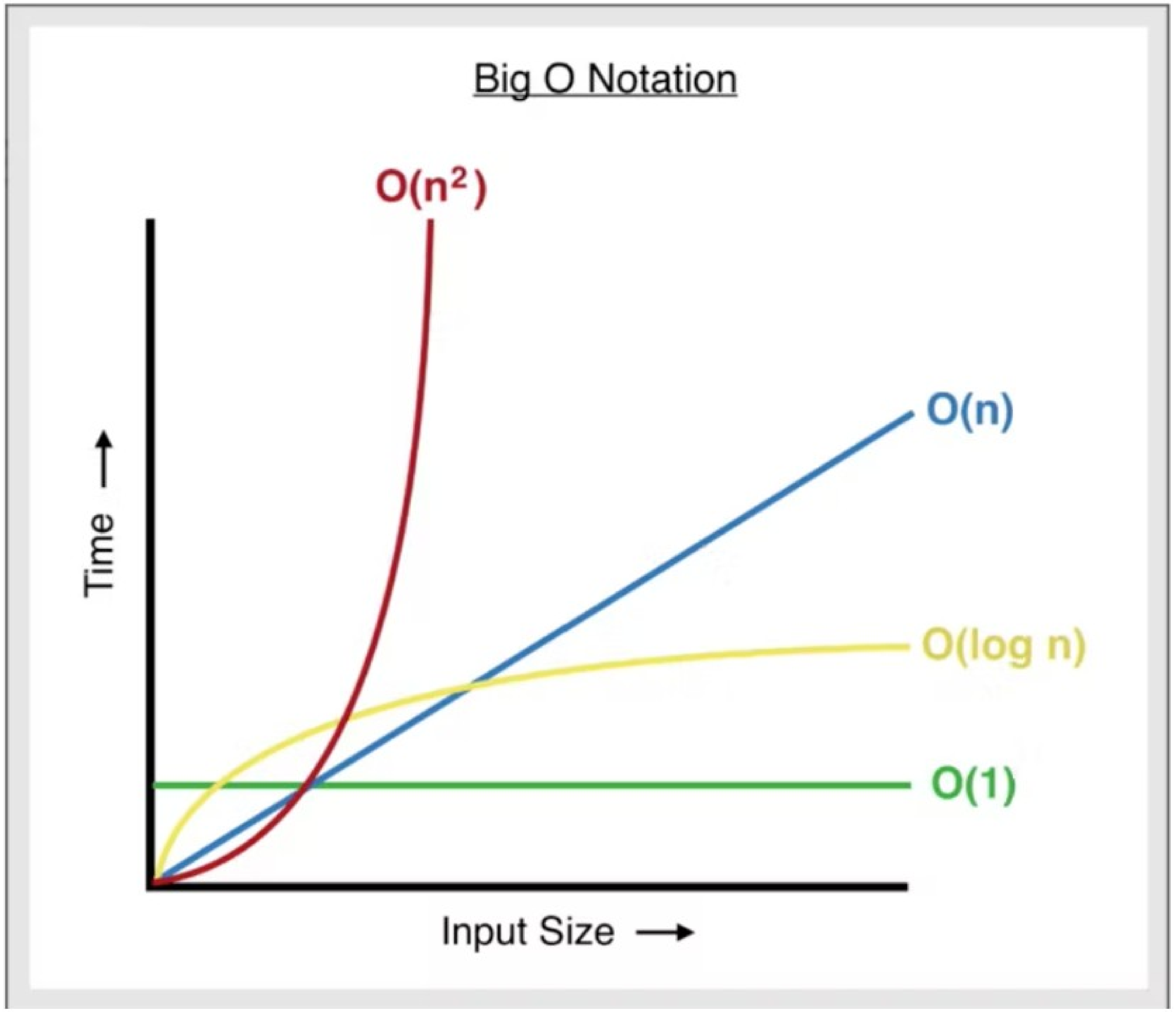
Mentimeter



Samme eksempel, men med binærsøk

- → Hvor stort er søkeromet?
- → Hva er tidskompleksiteten?

$O(\log(n))$



Bestemme O

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

Mentimeter

Bestemme O

- O(n^3 + 50n^2 + 10000) = O(n^3)
- O((n + 30) * (n + 5)) = O(n^2)
- O(n*log(n) + log(n)*log(n)) = O(n*log(n))
- O(n + n + n + n + n + n) = O(6n) = O(n)

Gruppeoppgaver

Kode Oppgave 1: Implementer binærsøk iterativt(loops).

Kode Oppgave 2: Implementer binærsøk ved bruk av rekursjon.

Oppgave 1

```
int product(int a, int b) {
    int sum = 0;
    for (int I = 0; I < b; I++) {
        sum += a;
    }
    return sum;
}
```

Oppgave 2

```
int mod(int a, int b) {
    if (b <=a) return -1;
    int div = a / b;
    return a - div * b;
}
```

Oppgave 3

```
static int power(int a, int b) {
    if (b < 0) return a;
    if (b == 0) return 1;
    int sum = a;
    for (int I = 0; I < b - 1; I++) {
        sum *= a;
    }
    return sum;
}
```

2

Mentimeter

Oppgave 1

```
int product(int a, int b) {
    int sum = 0;
    for (int I = 0; I < b; I++) {
        sum += a;
    }
    return sum;
}
```

Oppgave 2

```
int mod(int a, int b) {
    if (b <=a) return -1;
    int div = a / b;
    return a - div * b;
}
```

Oppgave 3

```
static int power(int a, int b) {
    if (b < 0) return a;
    if (b == 0) return 1;
    int sum = a;
    for (int I = 0; I < b - 1; I++) {
        sum *= a;
    }
    return sum;
}
```

Ukesoppgaver

R-1.9, R-1.11–1.15

C-1.8

C-1.19 (ignorer minnerestriksjonen)

C.1.24

Mentimeter

R-1.9 Bill has an algorithm, find2D, to find an element x in an $n \times n$ array A. The algorithm find2D iterates over the rows of A and calls the algorithm arrayFind, of Algorithm 1.12, on each one, until x is found or it has searched all rows of A. What is the worst-case running time of find2D in terms of n? Is this a linear-time algorithm? Why or why not?

O-Notasjon

Algorithm Loop1(n):
  $s \leftarrow 0$
  for $i \leftarrow 1$ **to** n **do**
    $s \leftarrow s + i$

Algorithm Loop2(n):
  $p \leftarrow 1$
  for $i \leftarrow 1$ **to** $2n$ **do**
    $p \leftarrow p \cdot i$

Algorithm Loop3(n):
  $p \leftarrow 1$
  for $i \leftarrow 1$ **to** n^2 **do**
    $p \leftarrow p \cdot i$

Algorithm Loop4(n):
  $s \leftarrow 0$
  for $i \leftarrow 1$ **to** $2n$ **do**
    for $j \leftarrow 1$ **to** i **do**
      $s \leftarrow s + i$

Algorithm Loop5(n):
  $s \leftarrow 0$
  for $i \leftarrow 1$ **to** n^2 **do**
    for $j \leftarrow 1$ **to** i **do**
      $s \leftarrow s + i$

C-1.8 Al and Bill are arguing about the performance of their sorting algorithms. Al claims that his $O(n \log n)$-time algorithm is *always* faster than Bill's $O(n^2)$-time algorithm. To settle the issue, they implement and run the two algorithms on many randomly generated data sets. To Al's dismay, they find that if $n < 100$, the $O(n^2)$-time algorithm actually runs faster, and only when $n \geq 100$ is the $O(n \log n)$-time algorithm better. Explain why this scenario is possible. You may give numerical examples.

C-1.19 An array A contains $n - 1$ unique integers in the range $[0, n - 1]$; that is, there is one number from this range that is not in A. Design an $O(n)$-time algorithm for finding that number. You are allowed to use only $O(1)$ additional space besides the array A itself.

C-1.24 Suppose that each row of an $n \times n$ array A consists of 1's and 0's such that, in any row of A, all the 1's come before any 0's in that row. Assuming A is already in memory, describe a method running in $O(n)$ time (not $O(n^2)$ time) for finding the row of A that contains the most 1's.

Spørsmål

Mentimeter

0 questions
0 upvotes