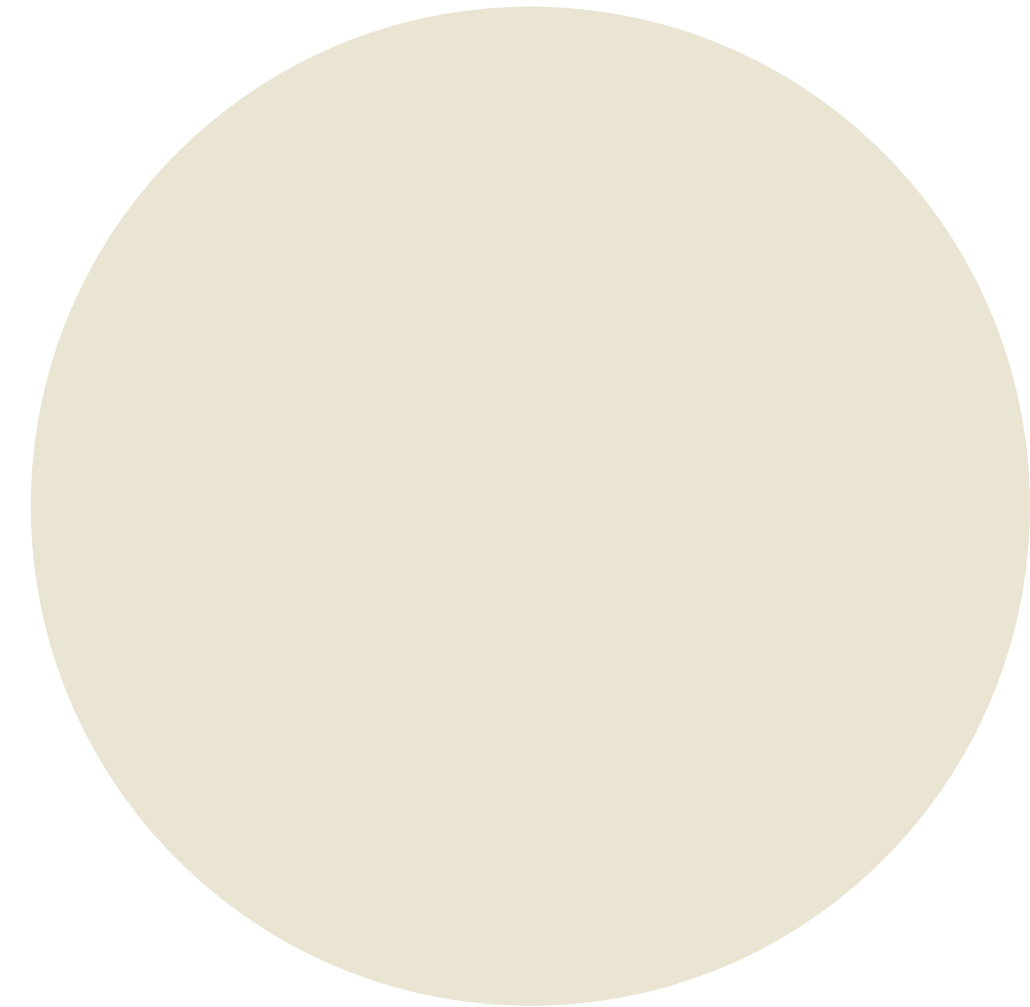


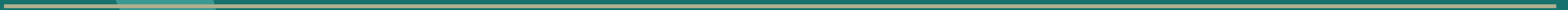


ASA 1

● KOMPLEKSITAS DAN SEQUENCE SEARCH



KOMPLEKSITAS



KOMPLEKSITAS

Waktu

- Jumlah langkah yang diperlukan algoritma untuk menyelesaikan masalah.
- Biasanya Dinotasikan dalam bentuk Big-O (O-notation).
- Big-O, merupakan batas langkah eksekusi terburuk pada algoritma. Contoh $O(n)$ berarti pada kasus terburuk program akan jalan sebanyak n kali
- Pada Python dalam 1 detik program dapat menjalankan kurang lebih 2.000.000 langkah


Ruang

- Jumlah memori yang diperlukan algoritma untuk menyelesaikan masalah.
- Biasanya Dinotasikan dalam bentuk $S(n)$.
- n pada kompleksitas ruang dapat dihitung secara sederhana dari banyaknya inputan
- algoritma yang memerlukan struktur data kompleks dapat memperbesar kompleksitas ruang seperti graph $S(n^2)$
- Pada python maksimal besar ruang kurang lebih 500.000.000 elemen

KOMPLEKSITAS

TRY IT YOURSELF!

Menghitung berapa kali program berjalan dalam 1 detik menggunakan python



```
1  import time
2
3  start = time.time()
4  count = 0
5  while time.time() - start < 1:
6      count += 1
7
8  print(f"Iterasi dalam 1 detik: {count}")
9
```

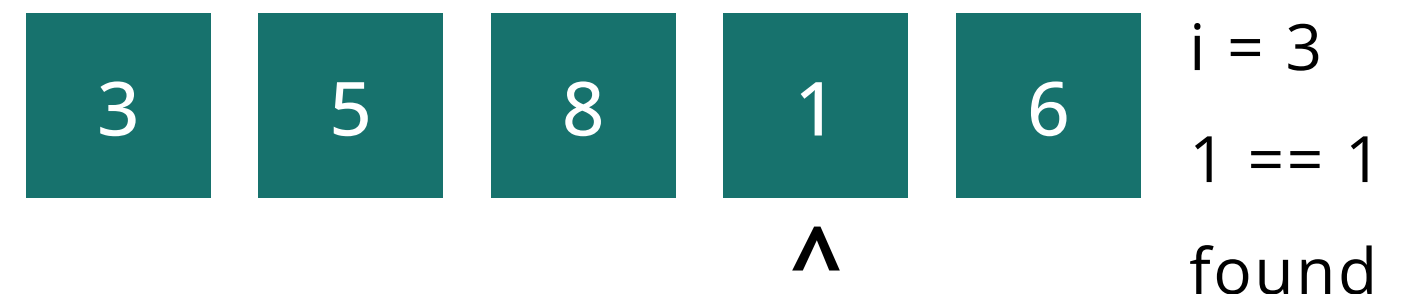
SEQUENTIAL SEARCH



SEQUENTIAL SEARCH

Sequential Search merupakan algoritma pencarian yang mencari target dengan membandingkan semua elemen dalam list dengan target

Target = 1



Target ditemukan pada index 3

KOMPLEKSITAS SEQUENTIAL SEARCH

```
1 def sequential_search(arr, x):
2     length = len(arr)
3
4     for i in range (length):
5         if arr[i] == x:
6             return i
7     return -1
8
9
10 arr = [1, 2, 3, 4, 5] #input
11 x = 3
12 ans = sequential_search(arr, x)
13 print(f'integer {x} berada di index: {ans}')
```

Kompleksitas waktu pada algoritma tersebut adalah $O(\text{length} + 4)$ dimana 4 merepresentasikan :

- $\text{length} = \text{len}(\text{arr})$
- $\text{arr} = [1, 2, 3, 4, 5]$ #input
- $x = 3$
- $\text{print}(\text{'integer \{x\} berada di index: \{ans\}'})$

Pada notasi Big-O angka dapat tidak di anggap sehingga kompleksitas waktunya adalah $O(\text{length})$

KOMPLEKSITAS SEQUENTIAL SEARCH

```
1 def sequential_search(arr, x):
2     length = len(arr)
3
4     for i in range (length):
5         if arr[i] == x:
6             return i
7     return -1
8
9
10 arr = [1, 2, 3, 4, 5] #input
11 x = 3
12 ans = sequential_search(arr, x)
13 print(f'integer {x} berada di index: {ans}')
```

Kompleksitas ruang pada algoritma tersebut adalah $S(\text{lenght}+1)$ dimana `length` merepresentasikan jumlah emelen pada **list** **arr** dan 1 merepresentasikan **variable** **x**.

seperti kompleksitas waktu, angkat dapat tidal dianggap sehingga kompleksitasnya menjadi $S(\text{lenght})$

SEQUENTIAL SEARCH PADA MATRIKS N*M

Kompleksitas waktu = $O(n*m)$

kompleksitas ruang = $S(n*m)$

```
1 def sequential_search(matrix, x):
2     rows = len(matrix)
3     cols = len(matrix[0]) if rows > 0 else 0
4
5     for i in range(rows):
6         for j in range(cols):
7             if matrix[i][j] == x:
8                 return (i, j)
9     return None
10
11 matrix = [
12     [1, 2, 3],
13     [4, 5, 6],
14     [7, 8, 9]
15 ]
16 x = 5
17 result = sequential_search(matrix, x)
18 if result:
19     print(f"Index elemen : {result}")
20 else:
21     print("Elemen tidak ditemukan")
```

SEQUENTIAL SEARCH (X-TH LARGEST ELEMENT)

LATIHAN

Diberikan sebuah list of integer arr dan integer x, dimana kalian harus mencari elemen terbesar ke-x, buatlah program untuk mencari nilai terbesar ke-x.

Contoh nilai x dan arr :

arr = [8, 3, 1, 5, 4]

x = 2

Output : 3

SEQUENTIAL SEARCH (X-TH LARGEST ELEMENT)

```
1 def find_xth_largest_element(arr, x):
2     if x > len(arr) or x <= 0:
3         return None
4     lenght = len(arr)
5     dum = 0
6     arr1 = arr
7     for i in range(x):
8         for j in range(lenght):
9             if arr1[j] > arr1[dum]:
10                 dum = j
11             arr1[dum] = 0
12     return arr[dum]
13
14
15 arr = [10, 4, 3, 50, 23, 90]
16 x = 3
17 ans = find_xth_largest_element(arr, x)
18 if ans is not None:
19     print(f"Hasil = {ans}")
```

Pada masalah ini, algoritma akan mengulang pencarian elemen terbesar sebanyak 5 kali, dan jika sudah didapatkan, elemen terbesarnya akan dieliminasi (dijadikan 0)

THANK YOU



● FOR YOUR NICE ATTENTION

link canva: https://www.canva.com/design/DAGfdfBfJ3k/jlbz-zgFoo8-TdKexGM5WA/edit?utm_content=DAGfdfBfJ3k&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton