

Nama : Abdi Setiawan

NIM : 2200016103

Kelas : A

TUGAS ALGORITMA PEMROGRAMAN 2



NIM : 2200016103

NAMA : ABDI SETIAWAN

KELAS : A

**PROGRAM STUDI SISTEM INFORMASI
FAKULTAS SAINS DAN TEKNOLOGI TERAPAN
UNIVERSITAS AHMAD DAHLAN
YOGYAKARTA
TAHUN AJARAN 2022/2023**

Nama : Abdi Setiawan

NIM : 2200016103

Kelas : A

A. Input Process Output (IPO) dan Kode Pemrograman.

1. Sequential Search

```
def sqsearch(data, key):  
    pos = []  
    hitung = 0  
    for i in range(len(data)):  
        hitung += 1  
        if data[i] == key:  
            pos.append(i+1)  
    if len(pos)>0:  
        print("Data", key, "Sebanyak", len(pos), "ditemukan di posisi", pos)  
    else:  
        print("Data tidak ditemukan")  
    print("Jumlah iterasi yang diperlukan:", hitung)  
    return pos
```

Input:

- **data**: Sebuah list yang berisi data untuk pencarian.
- **key**: Data yang akan dicari di dalam **data**.

Proses:

1. Menginisialisasi list **pos** sebagai list kosong untuk menyimpan posisi kemunculan **key**.
2. Menginisialisasi variabel **hitung** dengan nilai 0 untuk melacak jumlah iterasi.
3. Melakukan iterasi sebanyak **len(data)** kali menggunakan loop **for**.
4. Di setiap iterasi, menambahkan 1 ke variabel **hitung**.
5. Jika data pada indeks ke-i dalam **data** == **key**, maka:
 - Menambahkan posisi (**i+1**) ke dalam list **pos**.
6. Jika panjang list **pos** lebih dari 0, maka:
 - Mencetak pesan yang menyatakan bahwa data **key** ditemukan sebanyak **len(pos)** di posisi-posisi **pos**.
7. Jika panjang list **pos** sama dengan 0, maka:
 - Mencetak pesan yang menyatakan bahwa data **key** tidak ditemukan.
8. Mencetak jumlah iterasi yang diperlukan.
9. Mengembalikan list **pos** yang berisi posisi-posisi kemunculan **key** dalam **data**.

Output:

- Jika data **key** ditemukan di dalam **data**, maka outputnya adalah pesan yang menyatakan data **key** ditemukan sebanyak **len(pos)** kali di posisi-posisi **pos**.
- Jika data **key** tidak ditemukan di dalam **data**, maka outputnya adalah pesan yang menyatakan bahwa data **key** tidak ditemukan.
- Jumlah iterasi yang diperlukan juga dicetak sebagai output.
- Nilai yang dikembalikan adalah list **pos** yang berisi posisi-posisi kemunculan **key** dalam **data**.

Nama : Abdi Setiawan

NIM : 2200016103

Kelas : A

2. Binary Search

```
def binsearch(data,key):  
    awal = 1  
    akhir = len(data)+1  
    ketemu = False  
    hitung = 0  
    while (awal <= akhir) and not ketemu:  
        hitung += 1  
        tengah = int((awal+akhir)/2)  
        if key == data[tengah]:  
            ketemu = True  
            print("data", key,"ditemukan di indeks ke", tengah)  
        elif (key<data[tengah]):  
            akhir = tengah - 1  
        else:  
            awal = tengah + 1  
    if ketemu == False:  
        print("data tidak ditemukan")  
    print("Jumlah iterasi yang diperlukan:", hitung)
```

Input:

- **data**: Daftar elemen yang telah diurutkan.
- **key**: Elemen yang akan dicari dalam daftar **data**.

Proses:

- Inisialisasikan **awal** dengan nilai 1, yang menunjukkan indeks awal dari daftar.
- Inisialisasikan **akhir** dengan nilai **len(data) + 1**, yang menunjukkan indeks akhir dari daftar.
- Inisialisasikan **ketemu** sebagai False, yang menunjukkan apakah **key** ditemukan atau tidak.
- Inisialisasikan **hitung** dengan nilai 0, yang merepresentasikan jumlah iterasi.
- Selama **awal** kurang dari atau sama dengan **akhir** dan **ketemu** adalah False:
 - Tambahkan **hitung** dengan 1.
 - Hitung indeks **tengah** sebagai titik tengah antara **awal** dan **akhir**.
 - Jika **key** sama dengan **data[tengah]**:
 - Set **ketemu** menjadi True.
 - Cetak bahwa **key** ditemukan pada indeks **tengah**.
 - Jika **key** lebih kecil dari **data[tengah]**, perbarui **akhir** menjadi **tengah - 1**.
 - Jika **key** lebih besar dari **data[tengah]**, perbarui **awal** menjadi **tengah + 1**.
- Jika **ketemu** masih False, cetak bahwa **key** tidak ditemukan.
- Cetak jumlah total iterasi (**hitung**).

Output:

- Jika **key** ditemukan dalam daftar **data**, akan mencetak indeks di mana **key** ditemukan.

Nama : Abdi Setiawan

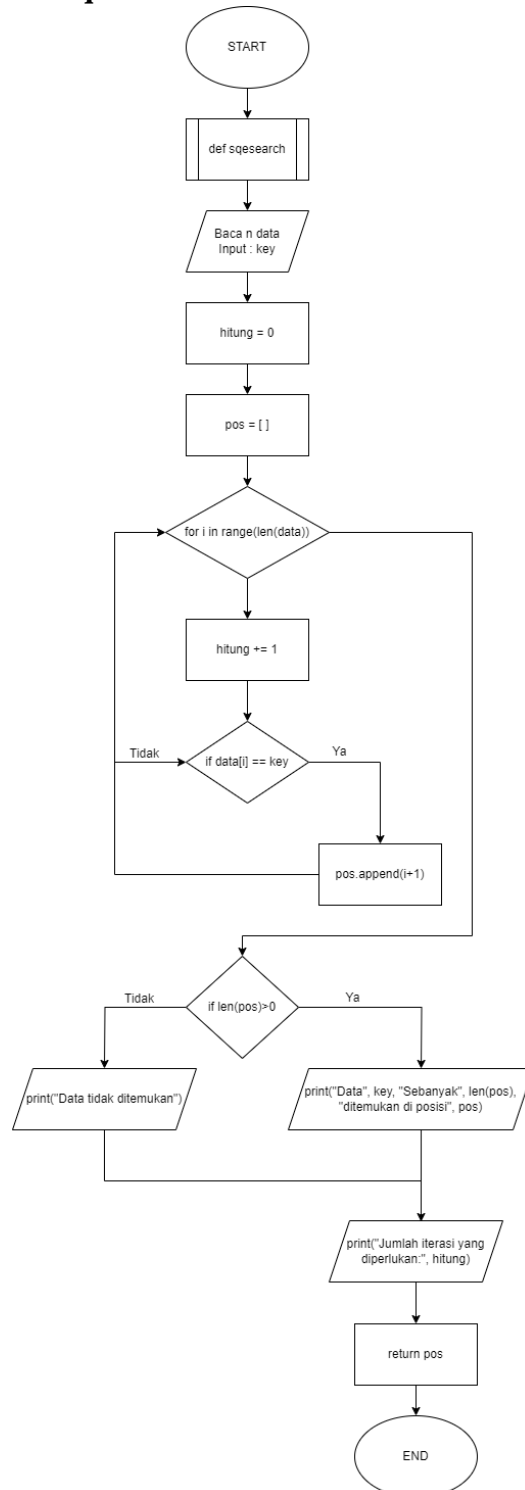
NIM : 2200016103

Kelas : A

- Jika **key** tidak ditemukan, akan mencetak bahwa data tidak ditemukan.
- Juga akan mencetak jumlah total iterasi yang diperlukan untuk melakukan pencarian.

B. Flowchart

1. Sequential Search

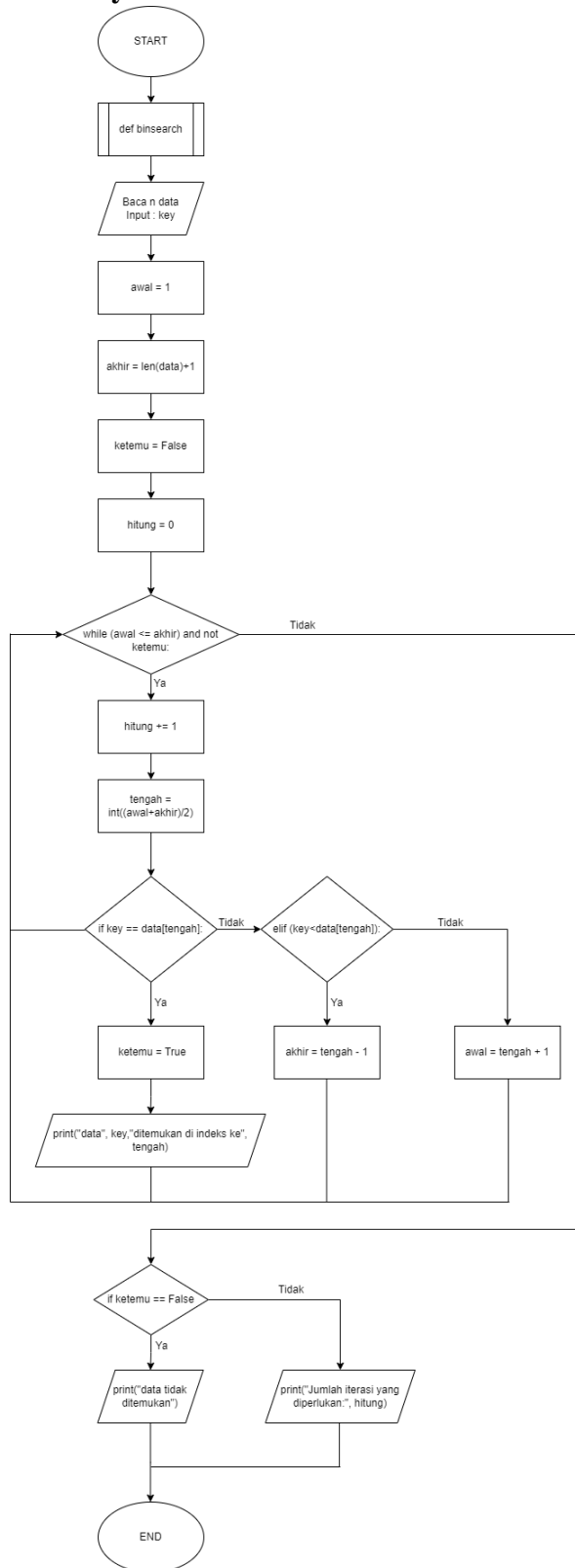


Nama : Abdi Setiawan

NIM : 2200016103

Kelas : A

2. Binary Search



Nama : Abdi Setiawan

NIM : 2200016103

Kelas : A

C. Hasil Run Pemrograman

```
def sqsearch(data,key):
    pos = []
    hitung = 0
    for i in range(len(data)):
        hitung += 1
        if data[i] == key:
            pos.append(i+1)
    if len(pos)>0:
        print("Data", key, "Sebanyak", len(pos), "ditemukan di posisi", pos)
    else:
        print("Data tidak ditemukan")
    print("Jumlah iterasi yang diperlukan:", hitung)
    return pos

data = "sistem informasi"
res = "".join(sorted(a))
sqsearch(res,"i")
```

[19]

... Data i Sebanyak 3 ditemukan di posisi [5, 6, 7]
Jumlah iterasi yang diperlukan: 16

[5, 6, 7]

```
# "SISTEM INFORMASI" NIH BOS
def binsearch(data,key):
    awal = 1
    akhir = len(data)+1
    ketemu = False
    hitung = 0
    while (awal <= akhir) and not ketemu:
        hitung += 1
        tengah = int((awal+akhir)/2)
        if key == data[tengah]:
            ketemu = True
            print("data", key,"ditemukan di indeks ke", tengah)
        elif (key<data[tengah]):
            akhir = tengah - 1
        else:
            awal = tengah + 1
    if ketemu == False:
        print("data tidak ditemukan")
    print("Jumlah iterasi yang diperlukan:", hitung)
a = "sistem informasi"
res = "".join(sorted(a))
print("data yang diurutkan:", res)
binsearch(res,"i")
```

[17]

... data yang diurutkan: aeffiiimmnorssst
data i ditemukan di indeks ke 4
Jumlah iterasi yang diperlukan: 2

```
def sqsearch(data,key):
    pos = []
    hitung = 0
    for i in range(len(data)):
        hitung += 1
        if data[i] == key:
            pos.append(i+1)
    if len(pos)>0:
        print("Data", key, "Sebanyak", len(pos), "ditemukan di posisi", pos)
    else:
        print("Data tidak ditemukan")
    print("Jumlah iterasi yang diperlukan:", hitung)
    return pos
NIM = [2,2,0,0,0,1,6,0,3]
sqsearch(NIM,0)
```

[16]

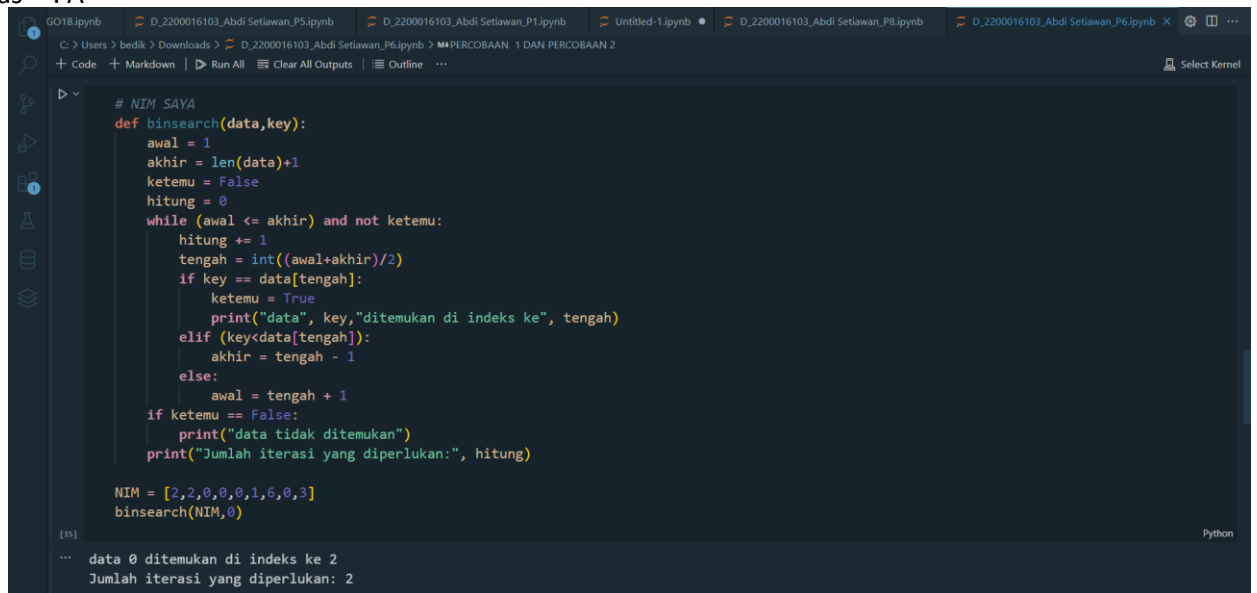
... Data 0 Sebanyak 4 ditemukan di posisi [3, 4, 5, 8]
Jumlah iterasi yang diperlukan: 9

[3, 4, 5, 8]

Nama : Abdi Setiawan

NIM : 2200016103

Kelas : A



The screenshot shows a Jupyter Notebook with several tabs at the top. The active tab is titled "D_2200016103_Abdi Setiawan_P6.ipynb". The notebook contains a Python function named `binsearch` that implements a binary search algorithm. The function takes a list `data` and a `key` as input. It initializes `awal` to 1, `akhir` to `len(data)+1`, `ketemu` to `False`, and `hitung` to 0. It then enters a `while` loop that continues as long as `awal` is less than or equal to `akhir` and `ketemu` is `False`. Inside the loop, it calculates the middle index `tengah` as `int((awal+akhir)/2)` and checks if `data[tengah]` equals the `key`. If it does, it sets `ketemu` to `True` and prints the index. If the key is less than the middle element, it updates `akhir` to `tengah - 1`; otherwise, it updates `awal` to `tengah + 1`. After the loop, it prints "data tidak ditemukan" if `ketemu` is still `False`, and finally prints the number of iterations. Below the function, it defines a list `NIM = [2,2,0,0,0,1,6,0,3]` and calls `binsearch(NIM,0)`. The output at the bottom shows that the key 0 was found at index 2 after 2 iterations.

```
# NIM SAYA
def binsearch(data,key):
    awal = 1
    akhir = len(data)+1
    ketemu = False
    hitung = 0
    while (awal <= akhir) and not ketemu:
        hitung += 1
        tengah = int((awal+akhir)/2)
        if key == data[tengah]:
            ketemu = True
            print("data", key,"ditemukan di indeks ke", tengah)
        elif (key<data[tengah]):
            akhir = tengah - 1
        else:
            awal = tengah + 1
    if ketemu == False:
        print("data tidak ditemukan")
    print("Jumlah iterasi yang diperlukan:", hitung)

NIM = [2,2,0,0,0,1,6,0,3]
binsearch(NIM,0)
```

[15]

```
... data 0 ditemukan di indeks ke 2
    Jumlah iterasi yang diperlukan: 2
```

Python