LAPORAN PROJECT PEMROGRAMAN DAN ALGORITMA

Program Cek Stok Toko Sepatu Berbasis Algoritma

Dibuat Oleh:

1. Alfina Septyanti (24030214032)
2. Lutfiatus Suuddiya (24030214034)
3. Muchamad Farid Z. (24030214056)
4. Dwi Nindy Ariyanti (24030214070)
5. Fransiska Maya S. (24030214084)
6. Ahmad Alvin H. (24030214149)

Kelas:

M 2024 B

Dosen Pengampu:

Hasanuddin Al-Habib, M.Si.

UNIVERSITAS NEGERI SURABAYA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

PROGRAM STUDI S1 MATEMATIKA

DAFTAR ISI

I. PENDAHULUAN

1.1 Latar Belakang

produk, mengurangi kesalahan inventaris, serta meningkatkan pengalaman pengguna.

Proyek ini mengembangkan sebuah program cek stok toko sepatu berbasis algoritma pencarian dengan memanfaatkan data yang disimpan dalam file Excel. Program dirancang dengan beberapa mode pencarian, yaitu pencarian berdasarkan kombinasi brand, series, dan size; pencarian berdasarkan size dengan pengurutan harga; serta pencarian berdasarkan rentang harga yang bersifat dinamis sesuai input pengguna.

Selain menampilkan hasil pencarian, program ini juga membandingkan waktu eksekusi tiga algoritma pencarian, yaitu Linear Search, Jump Search, dan Binary Search, untuk menganalisis efisiensi masing-masing algoritma pada data terurut. Dengan adanya beberapa mode pencarian dan pengukuran performa algoritma, diharapkan program ini dapat menjadi simulasi sistem pencarian stok yang efisien dan fleksibel pada toko sepatu.

1.2 Rumusan Masalah

1.2.1 Bagaimana efisiensi algoritma pencarian mempengaruhi performa database toko sepatu?

1.2.2 Bagaimana perbandingan algoritma Linear Search, Binary Search, dan Jump Search yang paling efisien untuk dataset terurut dan tidak terurut?

1.3 Tujuan

1.3.1 Mengetahui efisiensi algoritma pencarian mempengaruhi performa database toko sepatu

1.3.2 Mengetahui perbandingan algoritma Linear Search, Binary Search, dan Jump Search yang paling efisien untuk dataset terurut dan tidak terurut

II. ANALISIS DAN PERANCANGAN

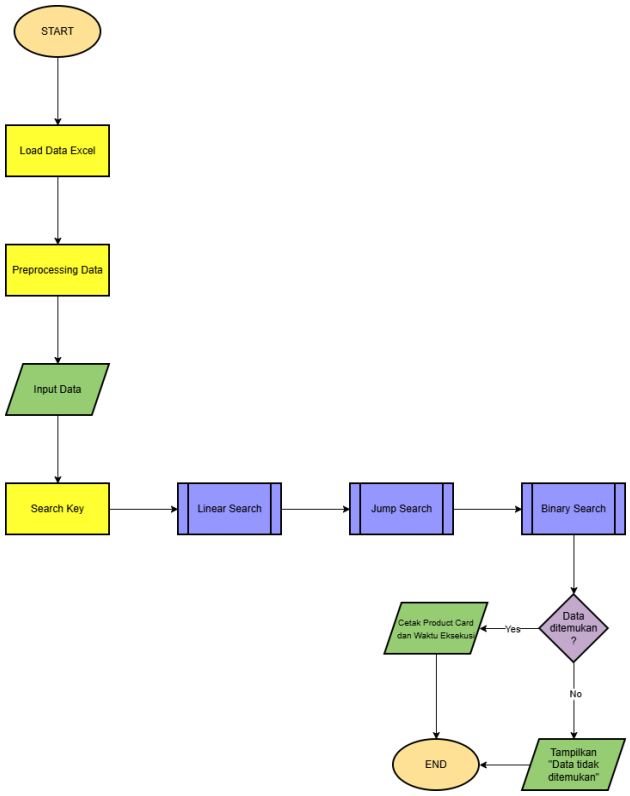
2.1 Analisis Kebutuhan Aplikasi

Aplikasi ini dirancang untuk membantu proses pencarian data produk pada database toko sepatu. Data disimpan dalam bentuk file Excel yang berisi informasi Brand, Series, Size, Price, dan Stok sepatu.

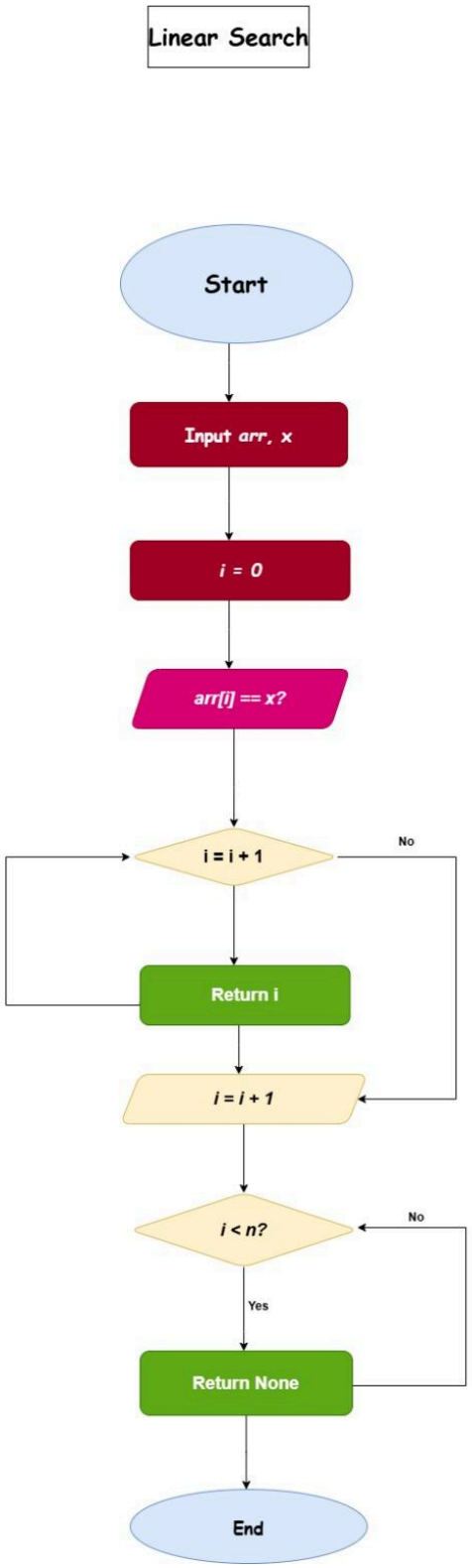
2.1.1 Kebutuhan aplikasi:

1. Data produk dari file Excel
2. Normalisasi data agar konsisten
3. Mengurutkan data
4. Melakukan pencarian data
5. Menampilkan hasil pencarian
6. Menghitung dan membandingkan waktu eksekusi

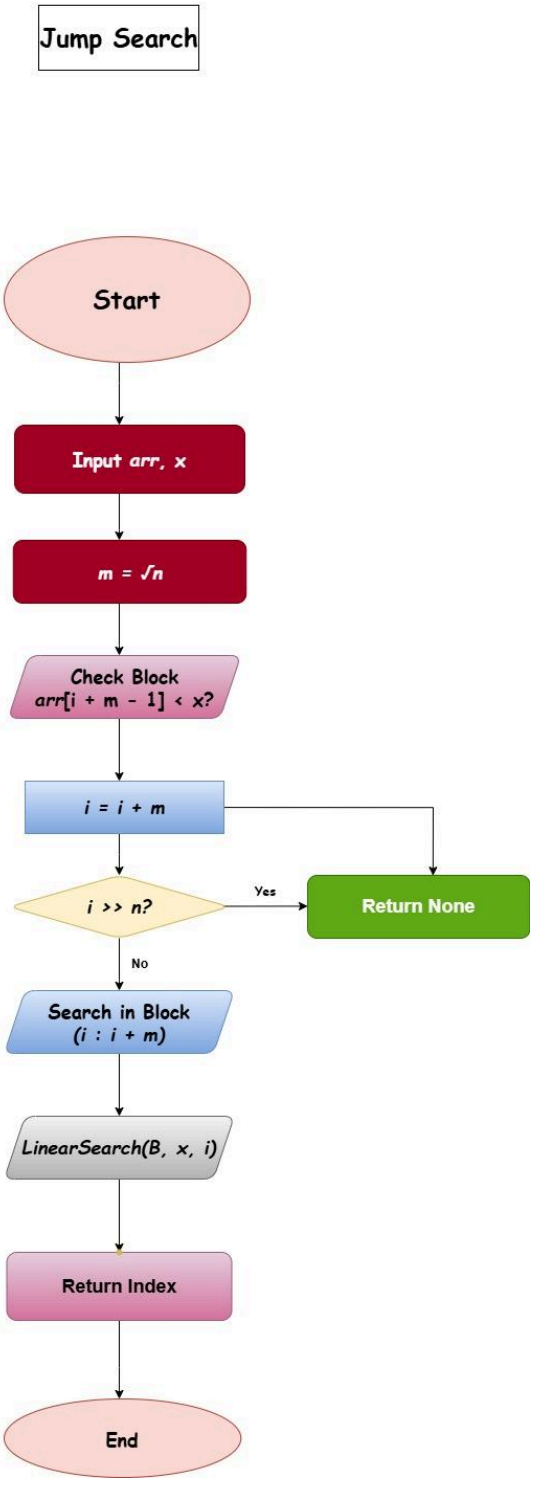
2.2 Diagram Alir



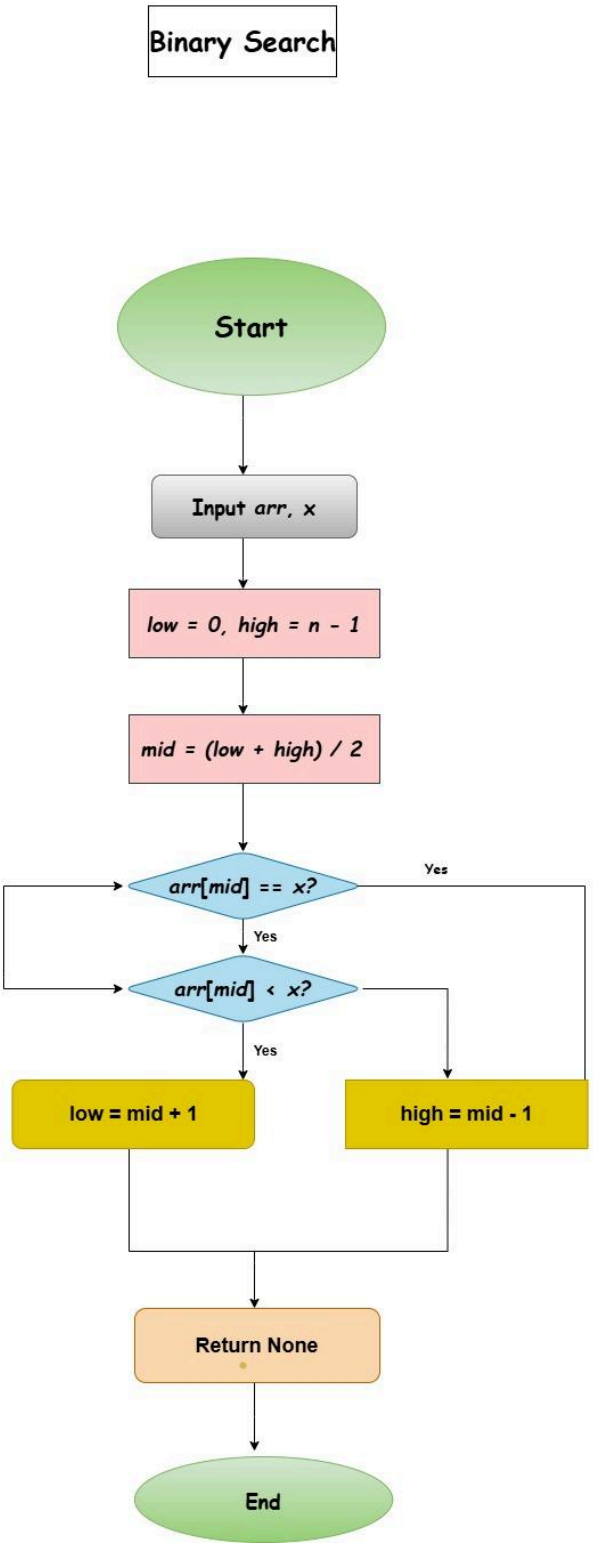
Flowchart Utama



Flowchart Linear Search



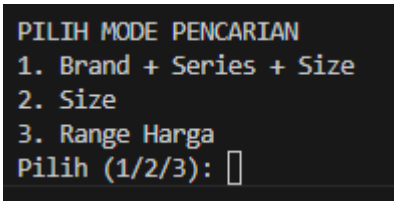
Flowchart Jump Seach



Flowchart Binary Search

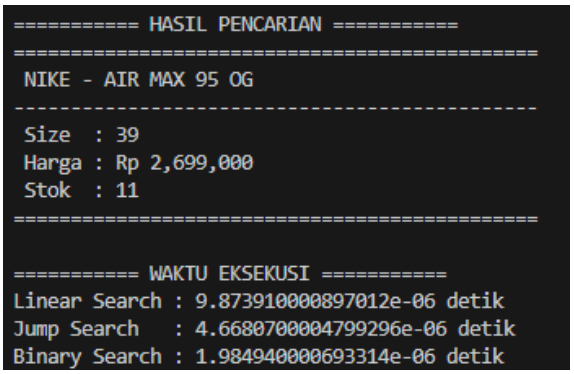
2.3 Sketsa Desain Antarmuka

2.3.1 Input Utama

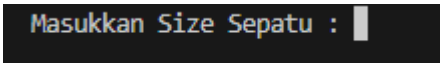


2.3.2 Input jika Pilih 1

Output setelah mengisi input Pilih 1



2.3.3 Input jika Pilih 2



Output setelah mengisi input Pilih 2 (Size 40)

```
========== HASIL PENCARIAN (SIZE) ==========
Jumlah rekomendasi: 24 sepatu
============================================
 AEROSTREET - HOOPS LOW 2.0
--------------------------------------------
 Size  : 40
 Harga : Rp 123,000
 Stok  : 21
============================================

============================================
 AEROSTREET - MASSIVE LOW
--------------------------------------------
 Size  : 40
 Harga : Rp 144,000
 Stok  : 11
============================================

============================================
 AEROSTREET - AUSTIN
--------------------------------------------
 Size  : 40
 Harga : Rp 175,000
 Stok  : 12
============================================

============================================
 AEROSTREET - OXFORD
--------------------------------------------
 Size  : 40
 Harga : Rp 179,000
 Stok  : 9
============================================
```

2.3.4 Input jika Pilih 3 (Rp.400000 - Rp.900000)

```
Masukkan harga minimum : Rp 400000
Masukkan harga maksimum: Rp (0 = tanpa batas) 900000
```

Output setelah mengisi input Pilih 3

```
========== HASIL PENCARIAN (RANGE HARGA) ==========
Range : Rp 400,000 - Rp 900,000
Jumlah sepatu yang direkomendasikan sesuai range harga: 30 sepatu

============================================
 COMPASS - GAZELLE
--------------------------------------------
 Size  : 42
 Harga : Rp 409,200
 Stok  : 11
============================================

============================================
 COMPASS - GAZELLE
--------------------------------------------
 Size  : 43
 Harga : Rp 409,200
 Stok  : 4
============================================

============================================
 COMPASS - GAZELLE
--------------------------------------------
 Size  : 38
 Harga : Rp 409,200
 Stok  : 7
============================================

============================================
 COMPASS - GAZELLE
--------------------------------------------
 Size  : 39
 Harga : Rp 409,200
 Stok  : 5
============================================
```

III. IMPLEMENTASI

3.1 Penjelasan Program

3.1.1 Linear Search

Linear Search bekerja dengan membandingkan data satu per satu dari awal hingga akhir.

```
# ------------------- LINEAR SEARCH -------------------
def LinearSearchFull(arr, x):
    n = len(arr)
    for i in range(len(arr)):
        if arr[i] == x:
            return i
    return None
```

3.1.2 Jump Search

Jump Search melompat beberapa elemen berdasarkan akar dari jumlah data, kemudian dilanjutkan dengan pencarian linear pada blok tertentu.

```
# ------------------- JUMP SEARCH -------------------
def JumpSearch(arr, x):
    n = len(arr)
    m = int(math.sqrt(n))
    if m == 0:
        m = 1

    i = 0
    while i < n and arr[min(i + m - 1, n - 1)] < x:
        #print(f"Processing Block = {arr[i:min(i+m, n)]}")
        i += m

    if i >= n:
        return None

    B = arr[i : min(i + m, n)]
    #print(f"[Processing Block = {B}]")

    return LinearSearch(B, x, i)
```

3.1.3 Binary Search

Binary Search membagi data menjadi dua bagian dan hanya dapat digunakan pada data yang sudah terurut.

```
# ------------------- BINARY SEARCH -------------------
def BinarySearch(arr, x):
    low, high = 0, len(arr) - 1

    while low <= high:
        mid = (low + high) // 2

        if arr[mid] == x:
            return mid
        elif arr[mid] < x:
            low = mid + 1
        else:
            high = mid - 1

    return None
```

3.2 Manual Penggunaan

Langkah-langkah Penggunaan:

1. Pastikan file *Data cepatu.xlsx* berada dalam satu folder dengan program
2. Jalankan program Python (*python Project Toko Sepatu Revisi.py*)
3. Masukkan pilihan yang diminta 1/2/3
   - Brand, Series, dan Size
   - Size
   - Range Harga
4. Program akan:
   - Menampilkan detail produk jika ditemukan
   - Menampilkan waktu eksekusi Linear, Jump, dan Binary Search

3.3 Screenshot Aplikasi

3.3.1 Hasil input

```
PILIH MODE PENCARIAN
1. Brand + Series + Size
2. Size
3. Range Harga
Pilih (1/2/3):
```

3.3.2 Hasil output

Jika pilih 1 (Brand : Puma + Series : Speedcat OG Unisex + Size : 40)

```
Pilih (1/2/3): 1
Masukkan Brand  : Puma
Masukkan Series : Speedcat OG Unisex
Masukkan Size   : 40

=========== HASIL PENCARIAN ===========
================================================
 PUMA - SPEEDCAT OG UNISEX
------------------------------------------------
 Size  : 40
 Harga : Rp 1,899,000
 Stok  : 15
================================================

=========== WAKTU EKSEKUSI ===========
Linear Search : 7.066470000427216e-06 detik
Jump Search   : 2.3270300000742525e-06 detik
Binary Search : 1.2053899932652712e-06 detik
PS C:\Users\User\OneDrive\Project PDA>
```

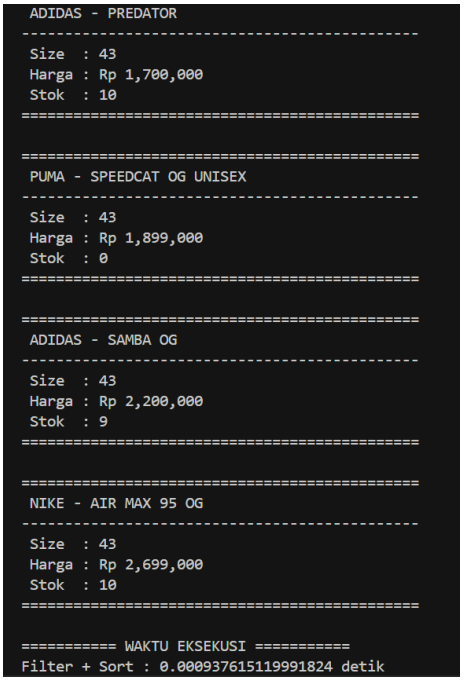
Jika pilih 2 (Size 43)

```
Pilih (1/2/3): 2
Masukkan Size Sepatu : 43

=========== HASIL PENCARIAN (SIZE) ===========
Jumlah rekomendasi: 24 sepatu
================================================
 AEROSTREET - HOOPS LOW 2.0
------------------------------------------------
 Size  : 43
 Harga : Rp 123,000
 Stok  : 0
================================================

================================================
 AEROSTREET - MASSIVE LOW
------------------------------------------------
 Size  : 43
 Harga : Rp 144,000
 Stok  : 6
================================================

================================================
 AEROSTREET - AUSTIN
------------------------------------------------
 Size  : 43
 Harga : Rp 175,000
 Stok  : 9
================================================
```

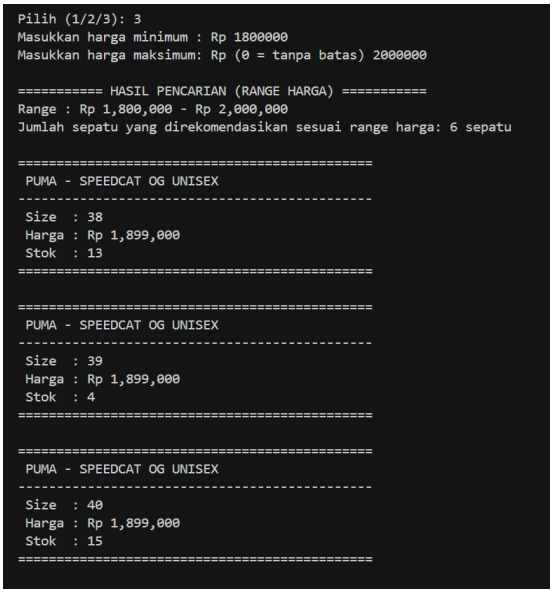
.

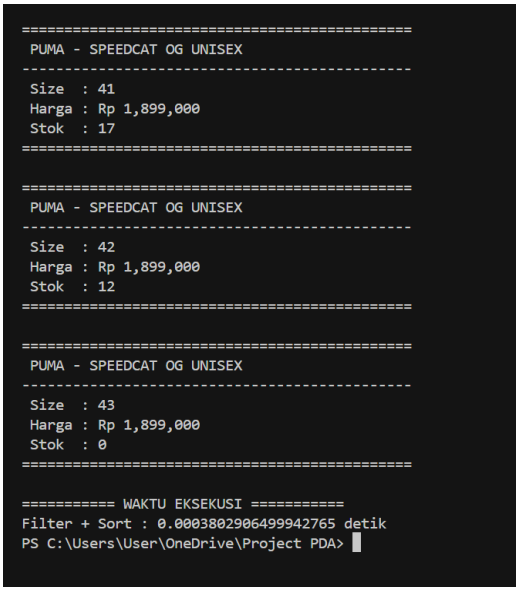
.

.



Jika pilih 3 (Range Harga min : Rp 1800.000, max : Rp 2.000.000)





IV. LAMPIRAN

```
import math
import time
import pandas as pd

# ------------------- LINEAR SEARCH -------------------
def LinearSearchFull(arr, x):
    n = len(arr)
    for i in range(len(arr)):
        if arr[i] == x:
            return i
    return None


# ------------------- LINEAR SEARCH (untuk Jump Search) -------------------
def LinearSearch(arr, x, loc):
    n = len(arr)
    for i in range(n):
        if arr[i] == x:
            return i + loc
    return None

# ------------------- JUMP SEARCH -------------------
def JumpSearch(arr, x):
    n = len(arr)
    m = int(math.sqrt(n))
    if m == 0:
        m = 1

    i = 0
    while i < n and arr[min(i + m - 1, n - 1)] < x:
        #print(f"Processing Block = {arr[i:min(i+m, n)]}")
        i += m
    if i >= n:
        return None

    B = arr[i : min(i + m, n)]
    #print(f"[Processing Block = {B}]")

    return LinearSearch(B, x, i)


# ------------------- BINARY SEARCH -------------------
def BinarySearch(arr, x):
    low, high = 0, len(arr) - 1

    while low <= high:
```

```
        mid = (low + high) // 2

        if arr[mid] == x:
            return mid
        elif arr[mid] < x:
            low = mid + 1
        else:
            high = mid - 1

    return None

# ------------------- LOAD EXCEL -------------------
def load_excel(namafile):
    df = pd.read_excel(namafile)

    df['BRAND'] = df['BRAND'].astype(str).str.lower().str.strip()
    df['SERIES'] = df['SERIES'].astype(str).str.lower().str.strip()
    df['SIZE'] = pd.to_numeric(df['SIZE'], errors='coerce')

    df = df.dropna(subset=['SIZE'])
    df['Key'] = list(zip(df['BRAND'], df['SERIES'], df['SIZE']))
    df = df.sort_values(by=['BRAND', 'SERIES', 'SIZE']).reset_index(drop=True)

    return df


# ------------------- PRODUCT CARD -------------------
def print_product_card(df, index):
    row = df.iloc[index]
    print("================================================")
    print(f" {row['BRAND'].upper()} - {row['SERIES'].upper()}")
    print("------------------------------------------------")
    print(f" Size  : {row['SIZE']}")
    print(f" Harga : Rp {row['PRICE']:,.0f}")
    print(f" Stok  : {row['STOK']}")
    print("================================================\n")


# ------------------- MODE 1 -------------------
def mode_search_key(df, keys):
    brand = input("Masukkan Brand  : ").lower().strip()
    series = input("Masukkan Series : ").lower().strip()
    size = float(input("Masukkan Size   : "))

    search_key = (brand, series, size)
    repeat = 10000

    start = time.perf_counter()
    for _ in range(repeat):
```

```
        hasil_lin = LinearSearchFull(keys, search_key)
    waktu_lin = (time.perf_counter() - start) / repeat

    start = time.perf_counter()
    for _ in range(repeat):
        hasil_jump = JumpSearch(keys, search_key)
    waktu_jump = (time.perf_counter() - start) / repeat

    start = time.perf_counter()
    for _ in range(repeat):
        hasil_bin = BinarySearch(keys, search_key)
    waktu_bin = (time.perf_counter() - start) / repeat

    hasil_akhir = hasil_bin if hasil_bin is not None else \
                hasil_jump if hasil_jump is not None else hasil_lin


print("\n========== HASIL PENCARIAN ==========")
    if hasil_akhir is not None:
        print_product_card(df, hasil_akhir)
    else:
        print("Data tidak ditemukan.")

    print("========== WAKTU EKSEKUSI ==========")
    print(f"Linear Search : {waktu_lin} detik")
    print(f"Jump Search   : {waktu_jump} detik")
    print(f"Binary Search : {waktu_bin} detik")

# ------------------- MODE 2 -------------------
def mode_search_size(df):
    size = float(input("Masukkan Size Sepatu : "))
    repeat = 10000

    start = time.perf_counter()
    for _ in range(repeat):
        hasil = df[df['SIZE'] == size].sort_values(by='PRICE')
    waktu = (time.perf_counter() - start) / repeat

    hasil = df[df['SIZE'] == size].sort_values(by='PRICE')


print("\n========== HASIL PENCARIAN (SIZE) ==========")

    print(f"Jumlah rekomendasi: {len(hasil)} sepatu")

    if hasil.empty:
        print("Tidak ada sepatu dengan size tersebut.")
    else:
        for idx in hasil.index:
```

```
            print_product_card(df, idx)

    print("========== WAKTU EKSEKUSI ==========")
    print(f"Filter + Sort : {waktu} detik")

# ------------------- MODE 3 (RANGE HARGA FINAL) -------------------
def mode_search_price_range(df):
    try:
        min_harga = int(input("Masukkan harga minimum : Rp "))
        max_harga = int(input("Masukkan harga maksimum: Rp (0 = tanpa batas) "))

        if min_harga < 0 or max_harga < 0:
            print("Harga tidak boleh bernilai negatif.")
            return

    except ValueError:
        print("Input harus berupa angka.")
        return

    repeat = 10000

    start = time.perf_counter()
    for _ in range(repeat):
        if max_harga == 0:
            hasil = df[df['PRICE'] >= min_harga].sort_values(by='PRICE')
        else:
            hasil = df[(df['PRICE'] >= min_harga) &
                       (df['PRICE'] <= max_harga)].sort_values(by='PRICE')
    waktu = (time.perf_counter() - start) / repeat

    if max_harga == 0:
        hasil = df[df['PRICE'] >= min_harga].sort_values(by='PRICE')
        range_text = f"≥ Rp {min_harga:,.0f}"
    else:
        hasil = df[(df['PRICE'] >= min_harga) &
                   (df['PRICE'] <= max_harga)].sort_values(by='PRICE')
        range_text = f"Rp {min_harga:,.0f} - Rp {max_harga:,.0f}"

    print("\n========== HASIL PENCARIAN (RANGE HARGA) ==========")
    print(f"Range : {range_text}")
    print(f"Jumlah sepatu yang direkomendasikan sesuai range harga: {len(hasil)} sepatu\n")

    if hasil.empty:
        print("Tidak ada sepatu yang direkomendasikan pada range harga tersebut.")
    else:
        for idx in hasil.index:
            print_product_card(df, idx)

    print("========== WAKTU EKSEKUSI ==========")
```

```
    print(f"Filter + Sort : {waktu} detik")

# ------------------- MAIN PROGRAM -------------------
if __name__ == "__main__":
    df = load_excel("Data cepatu.xlsx")
    keys = df['Key'].tolist()

    print("\nPILIH MODE PENCARIAN")
    print("1. Brand + Series + Size")
    print("2. Size")
    print("3. Range Harga")

    mode = input("Pilih (1/2/3): ")

    if mode == "1":
        mode_search_key(df, keys)
    elif mode == "2":
        mode_search_size(df)
    elif mode == "3":
        mode_search_price_range(df)
    else:
        print("Pilihan tidak valid.")
```

V. DAFTAR PUSTAKA

ACM Journal: "Efficiency of Search Algorithms in Databases" (2020).

IEEE Paper: "Jump Search for Unordered Data" (2018).