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

Dalam industri e-commerce, toko sepatu sering menghadapi tantangan dalam mengelola stok produk yang beragam, termasuk berdasarkan merk (misalnya Nike, Adidas, Puma, Aerostreet, Ventela, dan Compass), series (misalnya Air Max, Adizero SL 2, Oxford, dan lain-lain), ukuran (38-43), dan harga (rentang Rp 120.000 - Rp 2.700.000). Sistem cek stok yang efisien diperlukan untuk memastikan ketersediaan produk secara real-time, mengurangi kesalahan inventaris, dan meningkatkan pengalaman pelanggan. Algoritma pencarian memainkan peran krusial dalam mengoptimalkan proses ini, terutama pada database besar. Proyek ini mengembangkan program cek stok berbasis algoritma, dengan fokus pada perbandingan efisiensi tiga algoritma searching: Linear Search, Binary Search, dan Jump Search, untuk menentukan algoritma terbaik dalam konteks toko sepatu. Dataset simulasi akan mencakup brand impor seperti Adidas, Puma, dan Nike, serta brand lokal seperti Ventela, Compass, dan Aerostreet, untuk merepresentasikan variasi produk di pasar Indonesia.

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

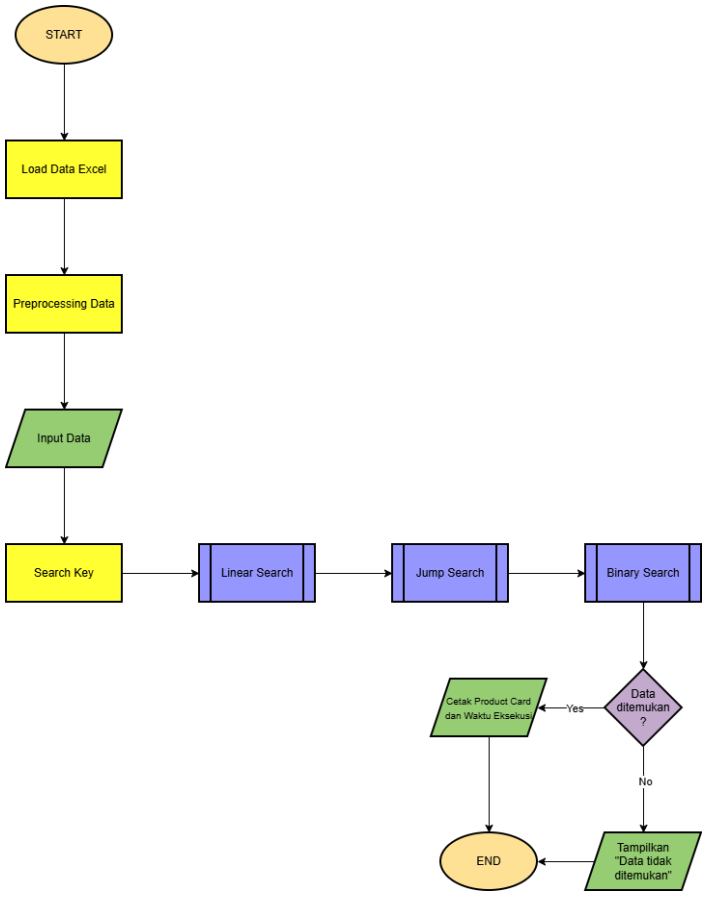
2.1 Analisis Kebutuhan Aplikasi

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:

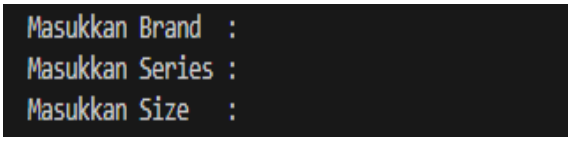
1. Data produk dari file Excel
2. Normalisasi data agar konsisten
3. Mengurutkan data
4. Melakukan pencarian data
5. Menampilkan hasil pencarian
6. Menghitung dan membandingkan waktu eksekusi

2.2 Diagram Alur

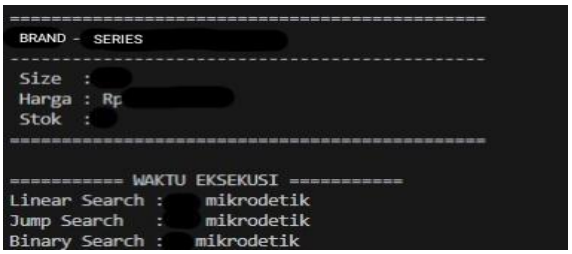


2.3 Sketsa Desain Antarmuka

2.3.1 Input oleh user



2.3.2 Output setelah mengisi input



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
```

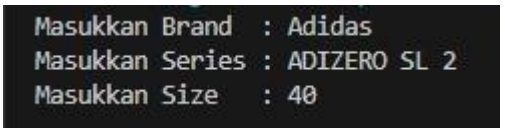
3.2 Manual Penggunaan

Langkah-langkah Penggunaan:

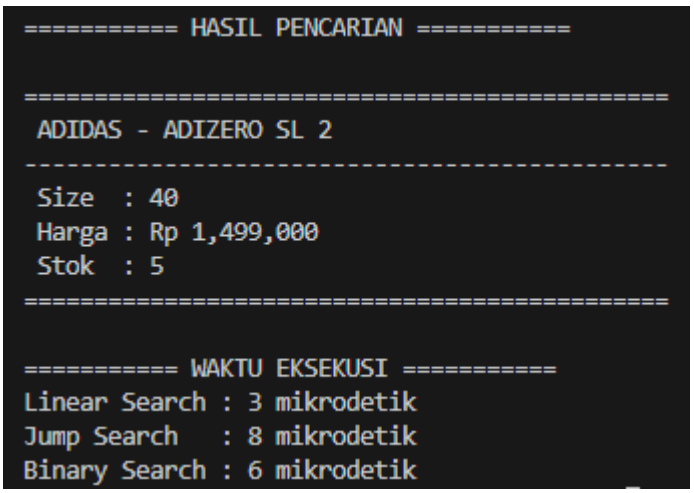
1. Pastikan file *Data cepatu.xlsx* berada dalam satu folder dengan program
2. Jalankan program Python (*python Project Toko Sepatu Kelompok 2.py*)
3. Masukkan data yang diminta:
   - Brand sepatu
   - Series sepatu
   - Size sepatu
4. Program akan:
   - Menampilkan detail produk jika ditemukan
   -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

3.3 Screenshot Aplikasi

Hasil input



Hasil output



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("\n==============================================")
    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")


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

    #print("Jumlah Data :", len(keys))

    brand = input("Masukkan Brand  : ").lower().strip()
    series = input("Masukkan Series : ").lower().strip()
    size = float(input("Masukkan Size   : "))

    search_key = (brand, series, size)

#------------------BENCHMARK WAKTU-----------------------
    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_lin or hasil_bin or hasil_jump
```

```
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 : {int(waktu_lin * 1_000_000)} mikrodetik")
print(f"Jump Search   : {int(waktu_jump * 1_000_000)} mikrodetik")
print(f"Binary Search : {int(waktu_bin * 1_000_000)} mikrodetik")
```

V. DAFTAR PUSTAKA

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

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