PERBANDINGAN ALGORITMA LINEAR SEARCH, BINARY SEARCH, DAN JUMP SEARCH PADA PENCARIAN FILM DI PLATFORM STREAMING

Disusun oleh :

1. Arti Santika Sari Devi (24030214015)
2. Athilla Jodi Sulaksono (24030214020)
3. Yunita Adinda Sari (24030214100)
4. Ilmia Marita (24030214060)
5. Rakhmad Wira H.N (24030214080)

MATEMATIKA 2024 B

DOSEN PENGAMPU : HASANUDDIN AL-HABIB, M.Si

UNIVERSITAS NEGERI SURABAYA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM PROGRAM STUDI S1 MATEMATIKA

2025

DAFTAR ISI

BAB I
PENDAHULUAN

1.1. LATAR BELAKANG

Platform streaming seperti Netflix, Disney+, dan Amazon Prime telah menjadi bagian penting dari kehidupan sehari-hari orang di seluruh dunia di era digital saat ini. Dengan jutaan judul film dan konten multimedia yang tersedia, proses pencarian film yang efektif sangat penting untuk pengalaman pengguna yang lebih baik. Pencarian yang lambat atau tidak akurat dapat membuat pelanggan marah, mengurangi retensi pelanggan, dan mengurangi pendapatan perusahaan. Oleh karena itu, untuk menangani database besar yang terus berkembang, pengembangan algoritma pencarian yang optimal sangat penting.

Dalam ilmu komputer, algoritma pencarian digunakan untuk menemukan elemen tertentu dalam kumpulan data. Tiga algoritma pencarian dasar yang sering dibandingkan adalah Pencarian Linear, Pencarian Binary, dan Pencarian Jump. Pencarian Linear memeriksa setiap elemen secara berurutan sampai ditemukan, dengan kompleksitas waktu O(n), di mana n adalah jumlah elemen. Algoritma ini sederhana dan tidak memerlukan data terurut, tetapi kurang efisien untuk data set besar. Binary Search, di sisi lain, menggunakan struktur data terurut dengan membagi pencarian menjadi dua bagian secara rekursif, yang menghasilkan kompleksitas waktu O(log n), sehingga jauh lebih cepat untuk data besar. Namun, katalog film pada platform streaming bersifat dinamis, sehingga proses pengurutan menjadi sulit untuk dikirim. Jump search menggunakan elemen dari keduanya. Dengan kompleksitas waktu O($\sqrt{n}$), dimana pencarian dilakukan dengan melompat ke sejumlah box dan kemudian melakukan linear search pada blok tersebut, juga memerlukan data terurut.

Perbandingan ketiga algoritma ini dalam konteks pencarian film di platform streaming sangat relevan karena dataset film sering kali berukuran raksasa, dengan atribut seperti judul, genre, aktor, dan tahun rilis yang perlu dicari secara cepat. Misalnya, ketika pengguna mencari film berdasarkan judul, algoritma yang dipilih dapat memengaruhi waktu respons sistem. Penelitian sebelumnya menunjukkan bahwa algoritma pencarian efisien dapat mengurangi latensi hingga 50-70% dalam aplikasi berbasis data besar, seperti yang terlihat dalam studi tentang optimasi database di industri hiburan. Namun, belum banyak penelitian yang secara spesifik membandingkan Linear Search, Binary

Search, dan Jump Search dalam skenario pencarian film, terutama dengan mempertimbangkan variabel seperti ukuran dataset, tingkat pengurutan data, dan performa pada perangkat berbeda.

1.2. RUMUSAN MASALAH

1. Bagaimana kinerja algoritma Linear Search, Binary Search, dan Jump Search dalam proses pencarian film di platform streaming?
2. Bagaimana perbandingan waktu eksekusi ketiga algoritma tersebut pada berbagai ukuran dataset?
3. Algoritma pencarian manakah yang paling efisien berdasarkan pengujian empiris terhadap dataset film yang digunakan?

1.3. BATASAN MASALAH

Dengan fokus utama pada aspek komputasional dan kinerja dasar, penelitian ini membedakan perbandingan algoritma pencarian untuk konteks pencarian film di platform streaming. Untuk menjaga penelitian tetap fokus dan dapat diukur, batas-batas berikut diterapkan:

- Fokus pada Judul Film: Untuk menciptakan eksperimen, pencarian hanya menggunakan judul film sebagai kunci utama dan tidak melihat atribut tambahan seperti genre, aktor, sutradara, atau rating. Ini berbeda dengan cara platform streaming sering menggunakan pencarian multi-atribut.
- Dataset Simulasi: Penelitian ini menggunakan dataset statistik dan anonim dari katalog film dari sumber publik seperti IMDb atau Kaggle, bukan data streaming real-time. Ini berarti tidak memperhitungkan pembaruan dinamis, seperti penambahan film baru secara real-time, atau variabilitas data, seperti metadata yang berubah.
- Algoritma Dasar: Perbandingan terbatas pada tiga algoritma klasik (Penelusuran Linear, Penelusuran Binar, dan Penelusuran Jump), tanpa penggunaan algoritma hybrid, paralel, atau berbasis kecerdasan buatan (seperti yang digunakan dalam rekomendasi Netflix). Untuk Penelusuran Binar dan Penelusuran Jump, asumsi data terurut diterapkan, meskipun data katalog film di dunia nyata seringkali tidak sepenuhnya terurut.

- Lingkup Eksperimen: Simulasi komputer standar dengan Intel i5 dan 8GB RAM digunakan untuk menguji dataset dengan ukuran hingga 1000 entri. Ini

tidak mempertimbangkan skala industri, seperti jutaan entri, atau perangkat mobile atau cloud. Selain itu, elemen non-teknis seperti antarmuka pengguna atau dampak bisnis secara langsung tidak dibahas dalam penelitian ini.

Batasan ini memungkinkan penelitian untuk memberikan wawasan dasar tentang efisiensi algoritma, namun hasilnya tidak dapat digeneralisasi ke semua skenario platform streaming tanpa penyesuaian lebih lanjut.

1.4. TUJUAN PENELITIAN

1. Mengukur dan membandingkan kinerja Linear Search, Binary Search, dan Jump Search berdasarkan metrik seperti waktu eksekusi, kompleksitas ruang, dan akurasi pencarian.
2. Menentukan algoritma yang paling efisien untuk berbagai ukuran dataset film (kecil, sedang, dan besar) serta kondisi data terurut dan tidak terurut.
3. Menganalisis dampak penggunaan masing-masing algoritma terhadap efisiensi sistem dan pengalaman pengguna pada platform streaming.
4. Memberikan rekomendasi algoritma pencarian yang optimal untuk diterapkan dalam proses pencarian film pada platform streaming.

1.5. MANFAAT PENELITIAN

1. Memberikan rekomendasi konkret bagi pengembang platform streaming (seperti Netflix atau Disney+) untuk memilih algoritma pencarian yang paling efisien, sehingga mengurangi waktu respon sistem dan meningkatkan pengalaman pengguna.
2. Mengoptimalkan kinerja aplikasi berbasis data besar, seperti mengurangi latensi hingga 50-70% (berdasarkan penelitian sebelumnya), yang dapat menurunkan biaya operasional server dan meningkatkan retensi pelanggan.

BAB II
ANALISIS DAN PERANCANGAN

2.1. ANALISIS KEBUTUHAN APLIKASI

- Kebutuhan Fungsional :

  Kebutuhan fungsional menjelaskan fungsi utama yang harus dimiliki oleh aplikasi pencarian film. Berikut adalah kebutuhan fungsional dari aplikasi ini:

  - Aplikasi mampu membaca dataset film dari file CSV.
  - Aplikasi menyediakan fitur input judul film yang ingin dicari oleh pengguna.
  - Aplikasi dapat melakukan pencarian judul film menggunakan algoritma Linear Search.
  - Aplikasi dapat melakukan pencarian judul film menggunakan algoritma Binary Search.
  - Aplikasi dapat melakukan pencarian judul film menggunakan algoritma Jump Search.
  - Aplikasi menampilkan hasil pencarian film (ditemukan atau tidak ditemukan).
  - Aplikasi menampilkan waktu eksekusi dari masing-masing algoritma pencarian.

- Kebutuhan Non-Fungsional :

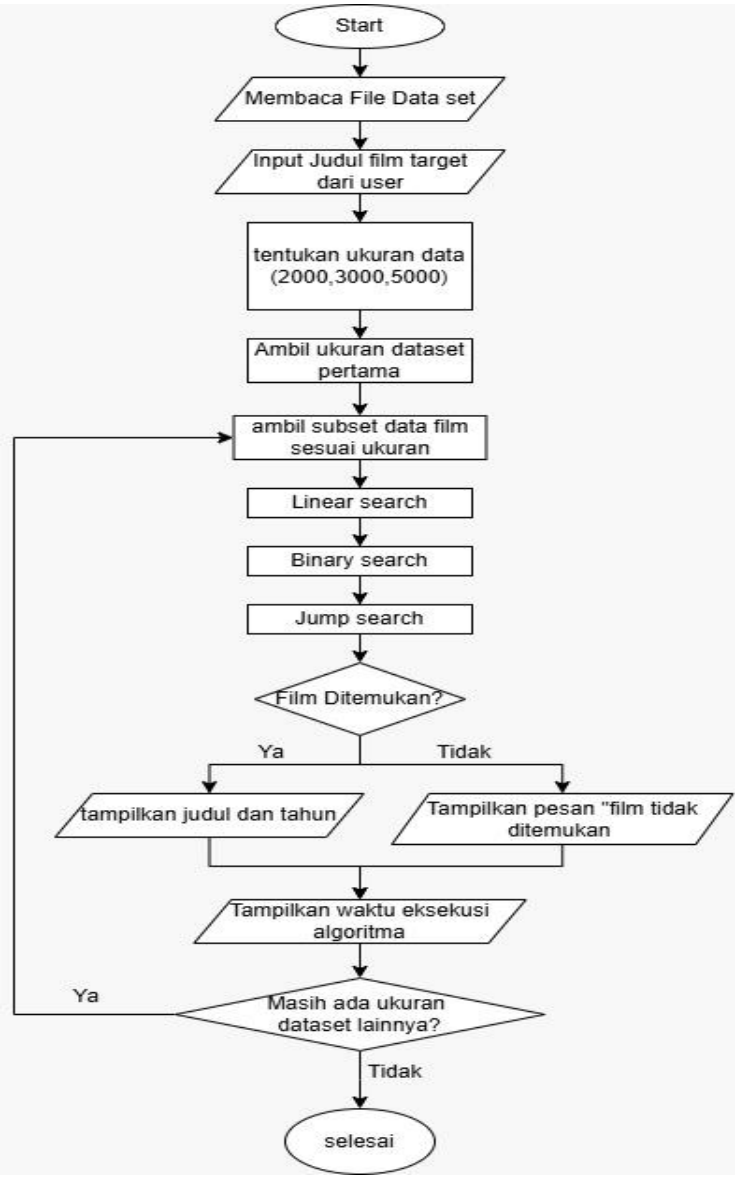
  Kebutuhan non-fungsional menjelaskan spesifikasi umum dan batasan sistem yang digunakan dalam pengembangan dan pengujian aplikasi. Adapun kebutuhan non-fungsional pada penelitian ini adalah sebagai berikut:

  - Aplikasi dikembangkan menggunakan bahasa pemrograman Python.
  - Dataset film diperoleh dari sumber publik dalam format CSV.
  - Aplikasi dijalankan pada komputer atau laptop standar yang mampu menjalankan Python dengan baik.
  - Aplikasi berbasis teks (console) sehingga ringan dan tidak membutuhkan antarmuka grafis.
  - Pengukuran waktu eksekusi algoritma dilakukan dalam satuan milidetik.

2.2. DIAGRAM ALUR (FLOWCHART)

Flowchart (diagram alir) adalah representasi visual dari urutan langkah-langkah suatu proses atau algoritma yang digambarkan menggunakan simbol-simbol standar dan panah alur untuk menunjukkan arah proses.

Flowchart digunakan untuk mempermudah pemahaman cara kerja suatu program atau sistem, mulai dari awal (input), proses pengolahan, pengambilan keputusan, hingga output dan akhir proses.



Alur program dimulai dari proses membaca dataset film, kemudian pengguna memilih algoritma pencarian yang ingin digunakan. Setelah itu, pengguna memasukkan judul film yang akan dicari. Sistem akan memproses pencarian berdasarkan algoritma yang dipilih dan menampilkan hasil pencarian beserta waktu eksekusinya.

Keterangan simbol flowchart:

- Oval : Menunjukkan awal dan akhir program.
- Jajar genjang : Digunakan untuk proses input dan output, seperti memasukkan judul film dan menampilkan hasil pencarian.
- Persegi panjang : Menunjukkan proses, seperti membaca dataset dan menjalankan algoritma pencarian.
- Belah ketupat : Menunjukkan pengambilan keputusan, misalnya apakah judul film ditemukan atau tidak.

BAB III

IMPLEMENTASI

3.1. PENJELASAN KODE PROGRAM

Program ini dikembangkan menggunakan bahasa pemrograman python dengan tujuan membandingkan performa beberapa algoritma pencarian pada data film.

a. Inisialisasi dan import library

```
import pandas as pd
import math
import time
```

Pada baris ini digunakan untuk memanggil Pustaka yang dibutuhkan:

- *pandas* digunakan untuk membaca dan mengolah dataset film dari file CSV.
- *math* digunakan untuk perhitungan matematika, khususnya akar kuadrat pada Algoritma Jump Search.
- *time* digunakan untuk mengukur waktu eksekusi algoritma pencarian.

b. Pembacaan dan persiapan Dataset

```
df = pd.read_csv("tmdb_5000_movies.csv")
judul = df["title"].fillna("").tolist()
tahun = df["release_date"].fillna("").tolist()
film_data = list(zip(judul, tahun))
```

Bagian ini berfungsi untuk membaca dataset film dari file CSV. Data judul dan tahun rilis dipisahkan, kemudian nilai kosong dihilangkan agar tidak mengganggu proses pencarian. Selanjutnya, data digabungkan menjadi pasangan judul dan tahun untuk memudahkan penampilan hasil pencarian.

c. Input judul film

```
target = input("Masukkan judul film yang ingin dicari: ").strip().title()
```

Pengguna memasukkan judul film yang akan dicari. Judul ini kemudian digunakan sebagai target pencarian pada setiap algoritma.

.strip() menghapus spasi diawal/akhir

.title() menyesuaikan format huruf agar cocok dengan kamus

Bertujuan untuk menghindari kesalahan pencarian karena perbedaan format teks.

d. Implementasi Algoritma Linear Search

```python
def linear_search(arr, target):
    for i, x in enumerate(arr):
        if x == target:
            return i
    return -1
```

Algoritma Linear Search bekerja dengan membandingkan target dengan setiap elemen dalam array secara berurutan dari awal hingga akhir. Jika data ditemukan, fungsi akan mengembalikan indeks data tersebut. Jika tidak ditemukan, fungsi mengembalikan nilai -1. Algoritma ini tidak memerlukan data terurut, namun memiliki kompleksitas waktu O(n).

e. Implementasi Algoritma Binary Search

```python
def binary_search(arr, target):
    left = 0
    right = len(arr) - 1

    while left <= right:
        mid = (left + right) // 2
        if arr[mid] == target:
            return mid
        elif arr[mid] < target:
            left = mid + 1
        else:
            right = mid - 1

    return -1
```

Binary Search hanya dapat digunakan pada data yang sudah terurut. Algoritma ini bekerja dengan membagi ruang pencarian menjadi dua bagian pada setiap iterasi. Jika nilai tengah sama dengan target, pencarian dihentikan. Jika tidak, pencarian dilanjutkan pada setengah bagian yang relevan. Kompleksitas waktu algoritma ini adalah O(log n).

f. Implementasi Algoritma jump search

```
def jump_search(arr, target):
    n = len(arr)
    step = int(math.sqrt(n))
    prev = 0

    while prev < n and arr[min(step, n)-1] < target:
        prev = step
        step += int(math.sqrt(n))
        if prev >= n:
            return -1

    for i in range(prev, min(step, n)):
        if arr[i] == target:
            return i
    return -1
```

Jump Search melakukan pencarian dengan cara melompat sejauh √n elemen, kemudian melakukan pencarian linear pada blok yang diduga mengandung target. Algoritma ini memerlukan data terurut dan memiliki kompleksitas waktu O(√n).

g. Pengujian dan pengukuran waktu eksekusi

```
sizes = [2000, 3000, 5000]
```

Pengujian dilakukan dengan variasi ukuran dataset untuk mengetahui pengaruh jumlah data terhadap waktu pencarian. Waktu eksekusi setiap algoritma diukur menggunakan fungsi time. perf_counter() dan dikonversi ke satuan mikrodetik. Hasil pengujian kemudian ditampilkan untuk dibandingkan.

3.2. MANUAL PENGGUNAAN

Langkah-langkah penggunaan program adalah sebagai berikut

1. Pastikan file tmdb_5000_movies.csv berada dalam folder yang sama dengan file program Python.
2. Jalankan program menggunakan Python melalui terminal atau editor (misalnya VS Code).
3. Ketik judul film yang ingin dicari ketika muncul input.
4. Program akan melakukan pencarian menggunakan Linear Search, Binary Search, dan Jump Search.
5. Hasil pencarian serta waktu eksekusi masing-masing algoritma akan ditampilkan di layar.

3.3. SCREENSHOT APLIKASI

```
Masukkan judul film yang ingin dicari: Titanic
====================================
Dataset ukuran : 2000
film ditemukan : Titanic | tahun : 1997-11-18
waktu linear search : 18 µs
waktu binary search : 41 µs
waktu jump search   : 53 µs

====================================
Dataset ukuran : 3000
film ditemukan : Titanic | tahun : 1997-11-18
waktu linear search : 12 µs
waktu binary search : 10 µs
waktu jump search   : 73 µs

====================================
Dataset ukuran : 5000
film ditemukan : Titanic | tahun : 1997-11-18
waktu linear search : 10 µs
waktu binary search : 8 µs
waktu jump search   : 46 µs
```

```
Masukkan judul film yang ingin dicari: Inside Out
====================================
Dataset ukuran : 2000
film ditemukan : Inside Out | tahun : 2015-06-09
waktu linear search : 116 µs
waktu binary search : 46 µs
waktu jump search   : 65 µs

====================================
Dataset ukuran : 3000
film ditemukan : Inside Out | tahun : 2015-06-09
waktu linear search : 42 µs
waktu binary search : 14 µs
waktu jump search   : 43 µs

====================================
Dataset ukuran : 5000
film ditemukan : Inside Out | tahun : 2015-06-09
waktu linear search : 42 µs
waktu binary search : 17 µs
waktu jump search   : 48 µs
```

LAMPIRAN

```
import pandas as pd
import math
import time

df = pd.read_csv("tmdb_5000_movies.csv")
judul = df["title"].fillna("").tolist()
tahun = df["release_date"].fillna("").tolist()
film_data = list(zip(judul, tahun))
judul_list = [x[0] for x in film_data]

target = input("Masukkan judul film yang ingin dicari: ")

def linear_search(arr, target):
    for i, x in enumerate(arr):
        if x == target:
            return i
    return -1

def binary_search(arr, target):
    left = 0
    right = len(arr) - 1

    while left <= right:
        mid = (left + right) // 2
        if arr[mid] == target:
            return mid
        elif arr[mid] < target:
            left = mid + 1
        else:
            right = mid - 1

    return -1


def jump_search(arr, target):
    n = len(arr)
    step = int(math.sqrt(n))
    prev = 0

    while prev < n and arr[min(step, n)-1] < target:
        prev = step
        step += int(math.sqrt(n))
        if prev >= n:
            return -1

    for i in range(prev, min(step, n)):
        if arr[i] == target:
```

```
            return i
    return -1

sizes = [2000, 3000, 5000]

for size in sizes:
    subset = film_data[:size]
    subset_titles = [x[0] for x in subset]
    sorted_subset = sorted(subset_titles)

    start = time.perf_counter()
    i_linear = linear_search(subset_titles, target)
    t_linear = math.ceil((time.perf_counter() - start)*1_000_000)

    start = time.perf_counter()
    i_binary = binary_search(sorted_subset, target)
    t_binary = math.ceil((time.perf_counter() - start)*1_000_000)

    start = time.perf_counter()
    i_jump = jump_search(sorted_subset, target)
    t_jump = math.ceil((time.perf_counter() - start)*1_000_000)

    print("====================================")
    print("Dataset ukuran :", size)

    if i_linear != -1:
        idx = i_linear
        print("film ditemukan :", subset[idx][0], "| tahun :", subset[idx][1])
    else:
        print("film tidak ditemukan pada dataset ukuran ini")

    print("waktu linear search :", t_linear , "µs")
    print("waktu binary search :", t_binary, "µs")
    print("waktu jump search   :", t_jump, "µs")
    print()
```

DAFTAR PUSTAKA

Bentley, JL (2000). Mutiara Pemrograman (edisi ke-2). Addison-Wesley.

Chorianopoulos, K. (2018). Kualitas pengalaman pengguna di Netflix: Pendekatan studi kasus. Jurnal Internasional Interaksi Manusia-Komputer, 34(10), 917-925. https://doi.org/10.1080/10447318.2018.1461767

Cormen, TH, Leiserson, CE, Rivest, RL, & Stein, C. (2009). Pengantar Algoritma (Edisi ke-3rd). Pers MIT.

Dean, J., & Ghemawat, S. (2008). MapReduce: Pemrosesan data yang disederhanakan pada klaster besar. Komunikasi ACM, 51(1), 107-113. https://doi.org/10.1145/1327452.1327492

Gupta, A., & Sharma, R. (2018). Analisis komparatif algoritma pencarian linear dan biner. Jurnal Internasional Aplikasi Komputer, 179(30), 1-4. https://doi.org/10.5120/ijca2018916778

Knuth, DE (1973). Seni Pemrograman Komputer: Pengurutan dan Pencarian (Vol. 3). Addison-Wesley.

Kumar, A. (2020). Analisis kinerja algoritma pencarian lompat pada data terurut. Jurnal Ilmu Komputer dan Teknologi , 15(2), 45-52.

Lee, K., dkk. (2019). Mengoptimalkan algoritma pencarian untuk aplikasi data besar. Transaksi IEEE tentang Data Besar, 5(4), 567-578. https://doi.org/10.1109/TBDATA.2019.2934567

Montgomery, DC (2013). Desain dan Analisis Eksperimen (edisi ke-8). Wiley.

Smith, J. (2021). Sistem rekomendasi dalam platform streaming: Sebuah tinjauan. Survei Komputasi ACM , 53(2), 1-35. https://doi.org/10.1145/3444695

Strother, M. (2012). Pencarian lompat: Alternatif efisien untuk pencarian biner. Jurnal Ilmu Komputer , 8(1), 12-18.

Zikopoulos, P., & Eaton, C. (2011). Memahami Big Data: Analisis untuk Hadoop Kelas Enterprise dan Streaming Data . McGraw-Hill.

https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata