ANALISIS EFISIENSI ALGORITMA SELECTION SORT ITERATIF DAN INSERTION SORT REKURSIF DALAM PEMILIHAN 10 MAHASISWA BERPRESTASI BERDASARKAN IPK

Disusun Oleh:

Nydia Artha Carissa (103052300101)

Keisha Hernantya Zahra (103052330063)

PRODI SAINS DATA

FAKULTAS INFORMATIKA

UNIVERSITAS TELKOM

2024

PENDAHULUAN

A. Latar Belakang

Di era digital saat ini, banyak orang berlomba lomba menemukan cara untuk pengelolaan dan analisis data yang efektif karena efektivitas menjadi semakin penting seiring dengan peningkatan volume dan kompleksitas data/informasi yang tersedia. Dalam konteks ini, algoritma sorting memiliki peran krusial karena berfungsi sebagai dasar dalam pengolahan data untuk berbagai aplikasi, seperti pencarian, pengelompokan, hingga pengambilan keputusan berbasis data. Pemahaman terhadap kinerja algoritma sorting tidak hanya membantu menentukan efisiensi dalam memproses data, tetapi juga bagaimana algoritma tersebut memanfaatkan sumber daya komputasi secara optimal.

B. Studi Kasus

Proyek ini berfokus pada analisis kinerja dua algoritma sorting, yaitu **Selection Sort secara Iteratif** dan **Insertion Sort secara Rekursif**, dalam skenario nyata dengan menggunakan dataset nyata. Pengurutan IPK (Indeks Prestasi Kumulatif) merupakan langkah penting untuk berbagai keperluan, salah satunya adalah untuk menentukan mahasiswa berprestasi. Dalam studi kasus ini, data yang digunakan mencakup *Student ID* dan *GPA (Grade Point Average)* dari sejumlah mahasiswa. Tugas utama adalah mengurutkan data berdasarkan kolom GPA secara menurun (descending), sehingga mahasiswa dengan IPK tertinggi berada di urutan teratas. Dari hasil pengurutan ini, akan diidentifikasi 10 mahasiswa dengan IPK terbaik.

C. Tujuan

1) Mengetahui Efisiensi Algoritma (seberapa cepat kedua algoritma dapat mengurutkan data dengan berbagai ukuran)
2) Mengetahui Skalabilitas Algoritma (bagaimana algoritma menangani data dalam skala yang lebih besar)
3) Mengetahui Akurasi Hasil (apakah kedua algoritma menghasilkan urutan yang sama, dan apakah data yang dihasilkan dapat diandalkan)

ANALISIS

A. Deskripsi Algoritma

1) Selection Sort

Selection Sort adalah metode penyortiran dengan mencari nilai data terkecil dari data posisi 0 ke posisi N-1. Algoritma ini bekerja dengan cara memilih elemen maksimum kemudian mempertukarkan elemen maksimum tersebut dengan elemen paling akhir untuk urutan ascending dan elemen pertama untuk descending. Proses ini dilakukan secara iteratif untuk semua elemen.

Iteratif:

- Pada iterasi pertama, algoritma membandingkan elemen pertama dengan semua elemen lainnya (n - 1 perbandingan).
- Pada iterasi kedua, algoritma membandingkan elemen kedua dengan sisa elemen yang belum terurut (n - 2 perbandingan).
- Proses ini terus berlanjut hingga hanya satu elemen yang tersisa untuk diproses.

2) Insertion Sort

Insertion Sort adalah algoritma pengurutan yang bekerja dengan cara menyisipkan elemen ke posisi yang sesuai dalam bagian array yang sudah terurut secara bertahap. Pada implementasi rekursif, setiap elemen disisipkan dengan cara memanggil fungsi secara berulang hingga elemen tersebut berada di posisi yang tepat.

Rekursif:

- Basis kasus: Jika hanya satu elemen, maka sudah terurut.
- Rekursif: Sisipkan elemen terakhir ke dalam subarray yang sudah terurut.
- Pisahkan elemen pertama sebagai array yang sudah terurut.
- Sisipkan elemen berikutnya ke posisi yang sesuai dalam array yang sudah terurut.
- Lakukan langkah ini secara rekursif untuk semua elemen.

B. Implementasi Algoritma

1) Algoritma Selection Sort secara Iteratif

```python
def selection_sort_iterative(df, column_name):
    data = df.copy()
    n = len(data)
    for i in range(n):
        max_index = i
        for j in range(i + 1, n):
            if data[column_name].iloc[j] > data[column_name].iloc[max_index]:
                max_index = j
        # Swap entire rows
        if max_index != i:
            data.iloc[[i, max_index]] = data.iloc[[max_index, i]].values
    return data
```

Langkah langkah algoritma:

1. DataFrame asli disalin untuk memastikan proses sorting tidak memodifikasi data asli.
2. Algoritma melakukan iterasi melalui setiap baris data menggunakan indeks *i*, pada setiap iterasi, indeks *i* menunjukkan posisi elemen yang sedang dicari untuk ditukar.
3. Dari posisi *i* ke akhir dataset, algoritma mencari indeks elemen dengan nilai maksimum pada kolom yang ditentukan (dalam hal ini kolom 'GPA'). Hal ini dilakukan dengan membandingkan setiap elemen dari posisi saat ini dengan elemen maksimum yang ditemukan sejauh ini.
4. Pertukaran Baris. Jika elemen maksimum yang ditemukan tidak berada pada posisi *i*, algoritma menukar baris tersebut dengan baris pada posisi *i*. Pertukaran dilakukan dengan menggunakan slicing pada pandas DataFrame, sehingga seluruh baris dipindahkan.
5. Setelah elemen maksimum untuk posisi *i* ditempatkan dengan benar, algoritma bergerak ke indeks berikutnya ($i + 1$) dan mengulangi proses hingga seluruh data diurutkan.

2) Algoritma Insertion Sort secara Rekursif

```python
def insertion_sort_recursive(df, column_name, n=None):
    if n is None:
        n = len(df)
        df = df.copy()
    # Base case
    if n <= 1:
        return df
    # Sort the first n-1 elements
    df = insertion_sort_recursive(df, column_name, n - 1)
    # Insert the nth element in the correct position
    last_row = df.iloc[n - 1].copy()
    j = n - 2
    while j >= 0 and df[column_name].iloc[j] < last_row[column_name]:
        df.iloc[j + 1] = df.iloc[j]
        j -= 1
    df.iloc[j + 1] = last_row
    return df
```

Langkah langkah algoritma:

Base Case : Jika jumlah elemen yang akan diurutkan kurang dari atau sama dengan 1 ($n \leq 1$), DataFrame dianggap sudah terurut lalu algoritma akan menghentikan rekursi.

1. Algoritma memanggil dirinya sendiri secara rekursif untuk mengurutkan elemen pertama hingga elemen ke n-1 (yaitu semua elemen kecuali elemen terakhir).
2. Setelah subset awal diurutkan, elemen terakhir (n) dimasukkan ke dalam posisi yang sesuai dengan membandingkannya secara berulang dengan elemen sebelumnya.
3. Selama elemen sebelumnya lebih kecil dari elemen terakhir (karena pengurutan menurun), algoritma akan memindahkan elemen sebelumnya ke posisi berikutnya.
4. Setelah menemukan posisi yang tepat, elemen terakhir ditempatkan di sana, dan algoritma selesai untuk subset tersebut.

C. Hasil Implementasi Algoritma

1) Algoritma Selection Sort Iteratif

```
Data Setelah Selection Sort Iteratif:
   StudentID       GPA
0       1045  4.000000
1       1443  4.000000
2       2279  4.000000
3       2706  4.000000
4       2920  4.000000
5       3029  4.000000
6       3320  4.000000
7       2261  3.984674
8       3355  3.979421
9       2901  3.951460
```

2) Algoritma Insertion Sort Rekursif

```
Data Setelah Insertion Sort Rekursif:
   StudentID       GPA
0       1045  4.000000
1       1443  4.000000
2       2279  4.000000
3       2706  4.000000
4       2920  4.000000
5       3029  4.000000
6       3320  4.000000
7       2261  3.984674
8       3355  3.979421
9       2901  3.951460
```

Baik algoritma Selection Sort Iteratif maupun Insertion Sort Rekursif menghasilkan urutan data yang identik menunjukkan bahwa kedua algoritma memiliki akurasi yang dapat diandalkan dan secara konsisten menempatkan data dalam urutan menurun berdasarkan kolom GPA.

D. Kompleksitas Waktu

1) Algoritma Selection Sort Iteratif

```
for i in range(n):
    max_index = i
    for j in range(i + 1, n):
        if data[column_name].iloc[j] > data[column_name].iloc[max_index]:
            max_index = j
    # Swap entire rows
    if max_index != i:
        data.iloc[[i, max_index]] = data.iloc[[max_index, i]].values
return data
```

$$\sum_{i=1}^{n}\sum_{j=i+1}^{n} 1 = \sum_{i=1}^{n}(n-i)$$

$$= \sum_{i=1}^{n} n - \sum_{i=1}^{n} i$$

$$= n^2 - \frac{n(n+1)}{2}$$

$$= \frac{2n^2-n^2-n}{2} = \frac{n^2-n}{2} \in \Theta(n^2)$$

2) Algoritma Insertion Sort Rekursif

```
if n <= 1:
    return df
# Sort the first n-1 elements
df = insertion_sort_recursive(df, column_name, n - 1)
# Insert the nth element in the correct position
last_row = df.iloc[n - 1].copy()
j = n - 2
while j >= 0 and df[column_name].iloc[j] < last_row[column_name]:
    df.iloc[j + 1] = df.iloc[j]
    j -= 1
df.iloc[j + 1] = last_row
return df
```

$T(n) = T(n-1) + n$ untuk $n \geq 1$

$T(n) = 1$ untuk $n \leq 1$

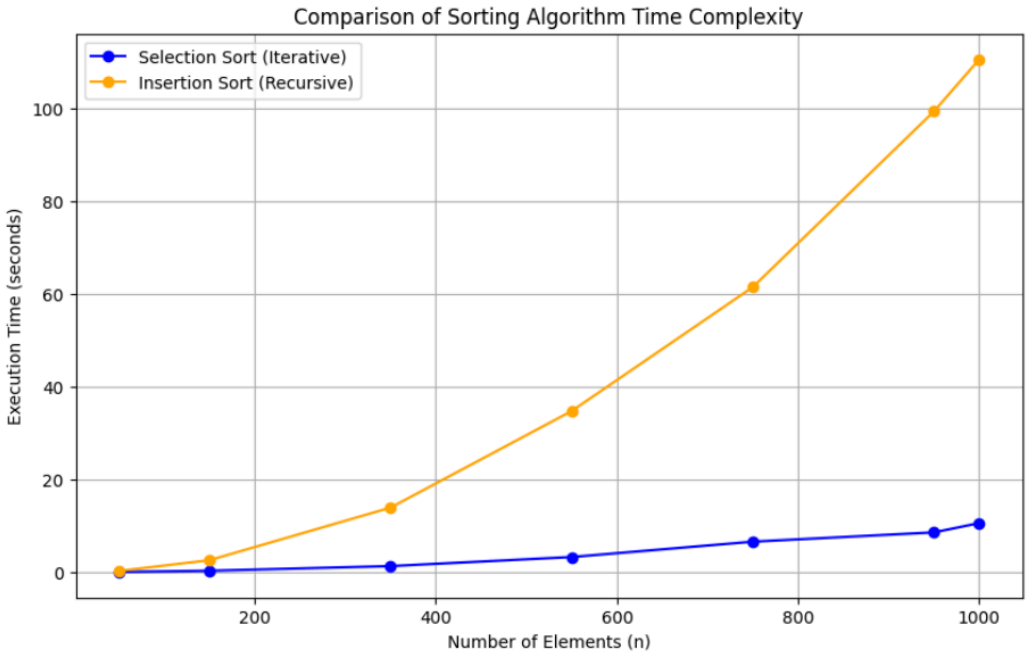
$$T(n) = T(n-1) + n$$
$$= (T(n-2) + n - 1) + n$$
$$= ((T(n-3) + (n-2)) + (n-1)) + n$$
$$\vdots$$
$$T(n) = T(1) + 2 + 3 + \ldots + (n-1) + n$$

$$T(n) = 1 + 2 + 3 + \ldots + n$$

$$T(n) = \frac{n(n+1)}{2} \in \Theta(n^2)$$

E. Grafik Perbandingan Running Time

Berikut adalah grafik perbandingan waktu eksekusi antara algoritma Selection Sort Iteratif dan Insertion Sort Rekursif berdasarkan dataset dengan ukuran berbeda:



Grafik perbandingan running time dihasilkan melalui implementasi yang memplot waktu eksekusi dari kedua algoritma terhadap ukuran dataset (n). Grafik menunjukkan bagaimana performa kedua algoritma berubah seiring bertambahnya ukuran data.

F. Analisis Perbandingan Kedua Algoritma

1) Efisiensi Waktu

- Selection Sort Iteratif menunjukkan waktu eksekusi yang lebih stabil karena sifat iteratifnya yang tidak tergantung pada stack rekursi.

  Namun, kompleksitasnya tetap $\Theta(n^2)$, sehingga pada dataset yang sangat besar, kinerjanya dapat menurun secara signifikan.
- Insertion Sort Rekursif memiliki performa yang baik pada dataset kecil karena efisiensi dalam mengatur elemen yang hampir terurut. Namun, pada dataset yang lebih besar, algoritma ini terhambat oleh batasan sistem (stack overflow) yang menyebabkan waktu eksekusi meningkat drastis atau bahkan gagal.

2) Skalabilitas

- Selection Sort lebih stabil untuk dataset dengan ukuran besar karena tidak menggunakan rekursi. Sebaliknya, Insertion Sort menghadapi

keterbatasan rekursi yang membuatnya kurang efisien untuk data berukuran besar.

3) Akurasi

- Kedua algoritma menghasilkan urutan data yang sama, sehingga sama-sama dapat diandalkan untuk tugas pengurutan data.

KESIMPULAN

Berdasarkan hasil analisis, algoritma Selection Sort Iteratif dan Insertion Sort Rekursif memiliki keunggulan dan kelemahan masing-masing, terutama dalam hal efisiensi, kompleksitas, dan skalabilitas. Selection Sort Iteratif terbukti lebih stabil saat digunakan untuk dataset besar, karena tidak bergantung pada stack rekursi. Dengan kompleksitas waktu, algoritma ini menunjukkan pola peningkatan waktu eksekusi yang bersifat kuadratik seiring bertambahnya ukuran dataset, seperti yang terlihat pada grafik. Namun, algoritma ini tidak efisien untuk data yang hampir terurut karena akan tetap dilakukan perbandingan di setiap iterasi. Sebaliknya, Insertion Sort Rekursif lebih unggul pada dataset kecil atau yang hampir terurut. Namun, algoritma ini memiliki keterbatasan signifikan pada dataset besar, karena penggunaan rekursi dapat menyebabkan waktu eksekusi meningkat secara drastis atau bahkan gagal akibat adanya batasan sistem. Kedua algoritma menghasilkan urutan data yang sama, membuktikan akurasi yang dapat diandalkan. Grafik perbandingan menunjukkan bahwa Selection Sort Iteratif lebih cocok untuk dataset besar, sementara Insertion Sort Rekursif lebih optimal untuk dataset kecil atau data yang sudah hampir terurut.

REFERENSI

1. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (1990). 33.3: Finding the convex hull. *Introduction to Algorithms*, 955-956.
2. Bisong, E., & Bisong, E. (2019). Matplotlib and seaborn. *Building machine learning and deep learning models on google cloud platform: A comprehensive guide for beginners*, 151-165.
3. McKinney, W. (2012). *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython*. " O'Reilly Media, Inc.".
4. Chivers, I., Sleightholme, J., Chivers, I., & Sleightholme, J. (2015). An introduction to Algorithms and the Big O Notation. *Introduction to Programming with Fortran: With Coverage of Fortran 90, 95, 2003, 2008 and 77*, 359-364.