Analisis Algoritma *Bubble Sort Ascending/Descending* dan Implementasinya Menggunakan Bahasa Pemrograman Python

Ahmad Yusuf[1], Yerix Ramadhani[2]
Sistem Informasi, Sains dan Teknologi, UIN Sulthan Thaha Saifuddin Jambi
*e-mail: ahmatyusufaja@gmail.com ,Yerixramadhani@uinjambi.ac.id

Abstrak

Penelitian ini bertujuan untuk menganalisis algoritma Bubble Sort dalam pengurutan data, baik secara *ascending* (menaik) maupun *descending* (menurun), serta mengimplementasikannya menggunakan bahasa pemrograman Python. Algoritma *Bubble Sort* adalah algoritma pengurutan yang sederhana namun efektif dalam mengurutkan elemen-elemen dalam suatu daftar dengan cara membandingkan pasangan elemen yang berdekatan dan menukarnya jika urutannya salah. Penelitian ini akan membahas cara kerja algoritma *Bubble Sort*, langkah-langkah yang dilakukan pada proses pengurutan secara ascending dan descending, serta membandingkan efisiensi keduanya berdasarkan kompleksitas waktu. Implementasi algoritma ini dilakukan dengan menggunakan Python untuk mempermudah visualisasi dan pengujian berbagai skenario pengurutan. Hasil analisis menunjukkan bahwa meskipun *Bubble Sort* memiliki kelebihan dalam kesederhanaannya, algoritma ini tidak efisien untuk data dalam jumlah besar, dengan kompleksitas waktu $O(n^2)$. Namun, algoritma ini tetap berguna untuk pengurutan data kecil atau sebagai pembelajaran dasar tentang pengurutan. Penelitian ini diharapkan dapat memberikan pemahaman yang lebih baik tentang implementasi dan efisiensi algoritma *Bubble Sort* dalam pengurutan data.

Kata kunci: *Algoritma Bubble Sort, Ascending, Descending, Pengurutan, Python*.

Abstract

This study aims to analyze the Bubble Sort algorithm in sorting data, both in ascending and descending order, and to implement it using the Python programming language. The Bubble Sort algorithm is a simple yet effective sorting method that arranges elements in a list by comparing adjacent pairs and swapping them if they are in the wrong order. This research discusses the working mechanism of the Bubble Sort algorithm, the steps involved in ascending and descending sorting processes, and compares the efficiency of both approaches based on time complexity. The algorithm is implemented using Python to facilitate visualization and testing across various sorting scenarios. The analysis shows that although Bubble Sort has the advantage of simplicity, it is inefficient for large datasets, with a time complexity of $O(n^2)$. However, it remains useful for sorting small datasets or as an introductory tool for learning sorting algorithms. This study is expected to provide a better understanding of the implementation and efficiency of the Bubble Sort algorithm in data sorting..

Keywords: *Bubble Sort Algorithm, Ascending, Descending, Sorting, Python*.

1 Pendahuluan (or Introduction)

Pengurutan data merupakan salah satu operasi dasar yang sering digunakan dalam berbagai aplikasi komputer, seperti pencarian, penyimpanan, dan pemrosesan data. Dalam dunia pemrograman, algoritma pengurutan memiliki peranan yang sangat penting untuk memastikan data yang ada dapat diakses atau diproses dengan efisien[1]. Salah satu algoritma pengurutan yang paling sederhana dan paling dikenal adalah *Bubble Sort*[2]. Algoritma ini bekerja dengan cara membandingkan dua elemen bersebelahan dalam sebuah array atau daftar, kemudian menukarnya jika urutannya salah. Proses ini diulang terus-menerus hingga seluruh elemen terurut dengan benar[3].

Bubble Sort memiliki dua varian utama dalam pengurutan, yaitu ascending (menaik) dan descending (menurun). Dalam urutan menaik (ascending), elemen-elemen diurutkan dari yang terkecil hingga yang terbesar, sedangkan pada urutan menurun (descending), elemen-elemen diurutkan dari

yang terbesar hingga yang terkecil[2]. Meskipun algoritma ini terkenal dengan kesederhanaannya dan mudah dipahami, ia tidak seefisien algoritma pengurutan lainnya, seperti *Merge Sort* atau *Quick Sort,* terutama pada data yang berukuran besar, dengan kompleksitas waktu $O(n^2)$. Oleh karena itu, algoritma *Bubble Sort* lebih cocok digunakan untuk pengurutan data dengan jumlah elemen yang relatif kecil atau dalam konteks pembelajaran algoritma dasar[4].

Penelitian ini bertujuan untuk menganalisis algoritma *Bubble Sort* dalam dua bentuk pengurutan, yaitu *ascending* dan *descending*, serta mengimplementasikannya menggunakan bahasa pemrograman Python. Python dipilih karena kemudahannya dalam menulis dan membaca kode serta banyaknya pustaka dan dokumentasi yang mendukung proses implementasi[5]. Dengan memahami cara kerja dan efisiensi algoritma *Bubble Sort* melalui analisis dan implementasi praktis, diharapkan pembaca dapat memperoleh pemahaman yang lebih dalam tentang mekanisme pengurutan serta keterbatasan yang dimiliki oleh algoritma ini.

Selain itu, penelitian ini juga akan membahas pengaruh pengurutan secara *ascending* dan *descending* terhadap performa *Bubble Sort*. Dengan menggunakan Python untuk implementasi algoritma, berbagai contoh dan skenario pengurutan akan diuji untuk memperoleh gambaran lebih jelas mengenai kecepatan dan efisiensi algoritma pada dataset yang berbeda[6].

Dengan demikian, penelitian ini tidak hanya bertujuan untuk mengimplementasikan algoritma *Bubble Sort* secara praktis, tetapi juga untuk memberikan analisis menyeluruh mengenai efisiensi dan aplikasinya dalam pengurutan data menggunakan Python.

2 Tinjauan Literatur

Bubble Sort adalah salah satu algoritma pengurutan yang paling sederhana. Algoritma ini bekerja dengan membandingkan elemen yang berdekatan dalam daftar dan menukarnya jika urutannya salah. Proses ini diulangi sampai tidak ada lagi pertukaran yang diperlukan, yang menunjukkan bahwa daftar telah diurutkan. Meski tidak efisien untuk data berukuran besar, *Bubble Sort* memiliki peran penting dalam pengenalan konsep dasar algoritma pengurutan.

Bubble Sort memiliki beberapa karakteristik penting yang mencerminkan kelebihan dan kelemahan algoritma ini[2]. Kompleksitas waktu pada kasus terburuk dan rata-rata adalah O(n2), karena setiap elemen dibandingkan dengan elemen lainnya dalam dua *loop* bersarang (nested loop). Ini membuat *Bubble Sort* tidak efisien untuk data berukuran besar. Misalnya, untuk n=10, dibutuhkan hingga $10^2=100$ operasi. Sedangkan jika data sudah hampir terurut, kompleksitas waktunya menjadi O(n). Algoritma hanya memerlukan satu iterasi penuh untuk memastikan tidak ada pertukaran elemen.

Bubble Sort bekerja dengan membandingkan elemen yang berdekatan. Jika elemen tidak dalam urutan yang benar, mereka ditukar. Proses ini berulang dari awal hingga akhir daftar. Setiap iterasi menempatkan elemen terbesar (atau terkecil, tergantung urutan) ke posisi akhir, sehingga disebut *bubbling up*[3].

Ascending adalah merngurutkan elemen dari nilai terkecil ke terbesar. Dan d*escending* mengurutkan elemen dari nilai terbesar ke terkecil. Perubahan logika sederhana, seperti mengganti operator perbandingan, dapat mengubah urutan dari ascending menjadi descending atau sebaliknya.[2]

Python adalah salah satu bahasa pemrograman yang populer dan sering digunakan dalam pendidikan, penelitian, dan pengembangan aplikasi, termasuk dalam implementasi algoritma seperti *Bubble Sort*. Python dirancang dengan filosofi *readability counts*, yang berarti sintaksnya mudah dibaca dan dipahami. Python adalah bahasa interpretatif, artinya kode dieksekusi secara langsung tanpa perlu dikompilasi. Python mendukung tipe data dinamis, sehingga programmer tidak perlu secara eksplisit menentukan tipe elemen dalam daftar (list)[5]. Fitur ini membuat implementasi algoritma menjadi lebih fleksibel dan mengurangi overhead pemrograman. Sehingga Python mendukung eksperimen algoritmik karena sintaks yang sederhana, interpretatif, dan kaya fitur untuk debugging, pengujian, serta visualisasi. Dengan sifatnya yang fleksibel, Python menjadi pilihan utama untuk memahami, mengimplementasikan, dan mengeksplorasi algoritma seperti *Bubble Sort*.

3 Metode Penelitian

Metode pada penelitian ini yaitu menggunakan Algoritma *Bubble Sort* terhadap dua buah varian model pengurutan data. Pertama secara *ascending* dari data terkecil ke data terbesar dan d*escending* dari data terbesar ke data terkecil, dengan menggunakan deretan data Array.

4 Hasil dan Pembahasan

a. Pengurutan data secara *Ascending*

Berikut adalah implementasi pengurutan data secara *ascending* menggunakan algoritma *Bubble Sort* dalam Python :

```
def bubble_sort_ascending(arr):
    n = len(arr)
    for i in range(n):  # Loop untuk setiap iterasi
        for j in range(0, n - i - 1):  # Loop untuk membandingkan
elemen
            if arr[j] > arr[j + 1]:  # Perbandingan untuk ascending
                # Tukar elemen jika elemen sekarang lebih besar dari
elemen berikutnya
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
    return arr

# Contoh penggunaan
data = [64, 34, 25, 12, 22, 11, 90]
sorted_data = bubble_sort_ascending(data)
print("Data sebelum diurutkan:", data)
print("Data setelah diurutkan (Ascending):", sorted_data)
```

a) Parameter Input
   - Fungsi bubble_sort_ascending menerima sebuah daftar arr sebagai input.

b) Iterasi Luar:
   - Loop for i in range(n) digunakan untuk memastikan semua elemen telah diperiksa.

c) Iterasi Dalam:
   - Loop for j in range(0, n - i - 1) membandingkan elemen berurutan dalam daftar.
   - Pengurangan **i** memastikan elemen terbesar yang sudah ditempatkan di akhir tidak dibandingkan lagi.

d) Logika Perbandingan:
   - if arr[j] > arr[j + 1]: Membandingkan elemen saat ini dengan elemen berikutnya.
   - Jika elemen saat ini lebih besar, elemen ditukar menggunakan sintaks Python: arr[j], arr[j + 1] = arr[j + 1], arr[j].

e) Return:
   - Fungsi mengembalikan daftar arr yang sudah diurutkan.

Berikut output dari kode program diatas :

```
Data sebelum diurutkan: [11, 12, 22, 25, 34, 64, 90]
Data setelah diurutkan (Ascending): [11, 12, 22, 25, 34, 64, 90]
```

b. Pengurutan data secara *Descending*

Berikut adalah implementasi pengurutan data secara d*escending* menggunakan algoritma *Bubble Sort* dalam Python :

```
def bubble_sort_descending(arr):
    n = len(arr)
```

```
    for i in range(n):  # Loop untuk setiap iterasi
        for j in range(0, n - i - 1):  # Loop untuk membandingkan elemen
            if arr[j] < arr[j + 1]:  # Perbandingan untuk descending
                # Tukar elemen jika elemen sekarang lebih kecil dari
elemen berikutnya
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
    return arr

# Contoh penggunaan
data = [64, 34, 25, 12, 22, 11, 90]
sorted_data = bubble_sort_descending(data)
print("Data sebelum diurutkan:", data)
print("Data setelah diurutkan (Descending):", sorted_data)
```

a) Parameter Input
   - Fungsi bubble_sort_descending menerima sebuah daftar arr sebagai input.

b) Iterasi Luar:
   - Loop for i in range(n) digunakan untuk memastikan semua elemen telah diperiksa.

c) Iterasi Dalam:
   - Loop for j in range(0, n - i - 1) membandingkan elemen berurutan dalam daftar.
   - Pengurangan i memastikan elemen terkecil yang sudah ditempatkan di akhir tidak dibandingkan lagi.

d) Logika Perbandingan:
   - if arr[j] < arr[j + 1]: Membandingkan elemen saat ini dengan elemen berikutnya.
   - Jika elemen saat ini lebih besar, elemen ditukar menggunakan sintaks Python: arr[j], arr[j + 1] = arr[j + 1], arr[j].

e) Return:
   - Fungsi mengembalikan daftar arr yang sudah diurutkan dalam urutan menurun.

Berikut output dari kode program diatas :

```
Data sebelum diurutkan: [90, 64, 34, 25, 22, 12, 11]
Data setelah diurutkan (Descending): [90, 64, 34, 25, 22, 12, 11]
```

Berikut hasil analisis alogoritma *Bubble Sort ascending* dan *descending* dengan implemetasi pemograman python, bahwa :

a) Pengurutan *ascending* menyusun data dari nilai terkecil ke terbesar.
b) Implementasi Python untuk pengurutan *ascending* menggunakan algoritma *Bubble Sort* membandingkan dua elemen yang berurutan dan menukar elemen jika elemen pertama lebih besar dari elemen kedua (arr[j] > arr[j + 1]).
c) Pengurutan *descending* menyusun data dari nilai terbesar ke terkecil.
d) Implementasi Python untuk pengurutan *descending* membandingkan elemen yang berurutan dan menukar elemen jika elemen pertama lebih kecil dari elemen kedua (arr[j] < arr[j + 1])
e) Kesamaannya bahwa keduanya menggunakan *loop* bersarang untuk membandingkan elemen. Tukar elemen dilakukan jika kondisi perbandingan terpenuhi. Dan algoritma *Bubble Sort* tetap stabil, menjaga urutan relatif elemen yang nilainya sama.
f) Dari efisiensi kompleksitas waktu:, kasus terbaik (data sudah terurut): O(n)O(n)O(n), dan kasus terburuk dan rata-rata: O(n2)O(n^2)O(n2).
g) Kompleksitas ruang, *Bubble sort* memiliki kompleksitas ruang O(1)O(1)O(1) karena tidak membutuhkan struktur data tambahan.
h) *Bubble Sort* lebih cocok untuk mempelajari dasar algoritma pengurutan, tetapi kurang efisien untuk pengurutan data dalam jumlah besar.

i) Python adalah pilihan tepat untuk implementasi algoritma *Bubble Sort* karena sintaks yang sederhana dan mendukung pengembangan kode fleksibel.
j) Implementasi *Bubble Sort* dengan Python memberikan dasar pemahaman tentang logika pengurutan baik secara *ascending* maupun *descending*.

5 Kesimpulan

Penelitian ini menyimpulkan bahwa algoritma *Bubble Sort* adalah salah satu metode pengurutan yang sederhana dan mudah dipahami, sehingga sangat cocok digunakan sebagai pengantar dalam mempelajari konsep dasar algoritma pengurutan. Algoritma ini bekerja dengan cara membandingkan elemen-elemen yang berdekatan dan menukarnya jika urutannya tidak sesuai, baik untuk pengurutan secara *ascending* (menaik) maupun *descending* (menurun). Namun, berdasarkan analisis kompleksitas waktu, algoritma *Bubble Sort* memiliki efisiensi yang rendah untuk data berukuran besar, dengan kompleksitas waktu $O(n^2)$. Implementasi algoritma ini menggunakan Python menunjukkan bahwa bahasa pemrograman ini sangat membantu dalam visualisasi proses pengurutan, sekaligus mempermudah pengujian berbagai skenario.

Beberapa saran sebagai dasar algoritma *Buble Sort*, bahwa penggunaan dalam pembelajaran *Bubble Sort* dapat terus digunakan sebagai alat pembelajaran untuk memahami dasar-dasar algoritma pengurutan, terutama bagi pemula di bidang algoritma dan pemrograman. Dan untuk aplikasi nyata yang melibatkan dataset besar, disarankan untuk menggunakan algoritma pengurutan yang lebih efisien seperti *Quick Sort, Merge Sort*, atau *Heap Sort*. Pengujian Algoritma Lain dapat dijadikan sebagai langkah lanjutan, membandingkan algoritma *Bubble Sort* dengan algoritma pengurutan lainnya dalam berbagai skenario dapat memberikan wawasan yang lebih luas tentang efisiensi algoritma dalam berbagai kondisi data.

Referensi

[1] F. Yanti and Mk. Emi Sita Eriana, “Algoritma Sorting Dan Searching,” vol. 1, pp. 1–122, 2024.
[2] N. Sari, W. A. Gunawan, P. K. Sari, I. Zikri, and A. Syahputra, “Analisis Algoritma Bubble Sort Secara Ascending Dan Descending Serta Implementasinya Dengan Menggunakan Bahasa Pemrograman Java,” *ADI Bisnis Digit. Interdisiplin J.*, vol. 3, no. 1, pp. 16–23, 2022, doi: 10.34306/abdi.v3i1.625.
[3] D. S. Rita Wahyuni Arifin, “Algoritma Metode Pengurutan Bubble Sort dan Quick Dalam Bahasa Pemrograman C++,” *Inf. Syst. Educ. Prof.*, vol. 4, no. 2, pp. 178–187, 2020.
[4] Y. A. Sandria, M. R. A. Nurhayoto, L. Ramadhani, R. S. Harefa, and A. Syahputra, “Penerapan Algoritma Selection Sort untuk Melakukan Pengurutan Data dalam Bahasa Pemrograman PHP,” *Hello World J. Ilmu Komput.*, vol. 1, no. 4, pp. 190–194, 2022, doi: 10.56211/helloworld.v1i4.187.
[5] V. Cutting and N. Stephen, “A Review on Using Python as A Preferred Programming Language for Beginners,” *Int. Res. J. Eng. Technol.*, vol. 8, no. 8, pp. 4258–4263, 2021, [Online]. Available: https://www.irjet.net/archives/V8/i8/IRJET-V8I8505.pdf
[6] Febby Wilyani, Qonaah Nuryan Arif, and Fitri Aslimar, “Pengenalan Dasar Pemrograman Python Dengan Google Colaboratory,” *J. Pelayanan dan Pengabdi. Masy. Indones.*, vol. 3, no. 1, pp. 08–14, 2024, doi: 10.55606/jppmi.v3i1.1087.