

LAPORAN PRAKTIKUM ▼

ALGORITMA DAN STRUKTUR DATA



IKA SULASTRI NINGSIH▼

240306030 ▼

[Github.com/IKASULASTRININGSIH](https://github.com/IKASULASTRININGSIH)▼

Program Studi Teknologi Informasi
Fakultas Dakwah dan Ilmu Komunikasi
Universitas Islam Negeri Mataram
2024 ▼

Pertemuan	11
Topik	Sorting
Repository	
Tanggal	9 Desember 2024

A. Tujuan

Tujuan praktikum pengenalan algoritma dengan flowchart

Tujuan sorting (pengurutan) adalah untuk mengatur elemen-elemen dalam suatu kumpulan data (array, daftar, atau struktur data lainnya) agar disusun dalam urutan tertentu, baik menaik (ascending) maupun menurun (descending). Sorting memiliki beberapa tujuan utama:

1. Mempermudah Pencarian Data

Data yang terurut memungkinkan algoritma pencarian, seperti binary search, bekerja lebih efisien dibandingkan dengan data yang tidak terurut.

2. Mengoptimalkan Proses Analisis Data

Data yang terurut mempermudah identifikasi pola, seperti tren, nilai maksimum/minimum, atau elemen duplikat.

3. Peningkatan Efisiensi Pengolahan Data

Sorting mempermudah pengolahan data pada langkah berikutnya, seperti penggabungan (merge), penghapusan duplikat, atau pemrosesan statistik.

4. Penyajian Data yang Lebih Terstruktur

Data yang terurut lebih mudah dipahami, sehingga memudahkan dalam pelaporan atau visualisasi.

5. Mengelompokkan Data Berdasarkan Kriteria Tertentu

Sorting membantu dalam mengelompokkan data dengan kriteria tertentu untuk analisis lebih lanjut. Misalnya, mengurutkan siswa berdasarkan nilai ujian atau pendapatan.

B. Requirement

1. Sistem Operasi yang digunakan : Windows 10 vers
2. Browser : Google Chrome Version
129.0.6668.100 (Official Build)
(x86_64)
3. Tools yang digunakan : Visual Studio Code

C. Dasar Teori

Dasar Teori Sorting

Sorting adalah salah satu proses fundamental dalam ilmu komputer yang bertujuan untuk mengatur elemen dalam kumpulan data sesuai dengan urutan tertentu, seperti menaik (ascending) atau menurun (descending). Sorting digunakan untuk meningkatkan efisiensi dalam pengolahan data, pencarian, dan analisis.

Klasifikasi Sorting

Sorting Internal dan Eksternal:

Internal Sorting: Data yang akan diurutkan seluruhnya dimuat ke dalam memori utama. Contoh: Bubble Sort, Insertion Sort.

External Sorting: Digunakan untuk data yang lebih besar dari kapasitas memori utama. Contoh: Merge Sort eksternal.

Metode Sorting Berdasarkan Pendekatan:

Comparison-based Sorting: Menggunakan perbandingan antar elemen. Contoh: Quick Sort, Merge Sort, Heap Sort.

Non-comparison-based Sorting: Menggunakan metode lain seperti hashing atau distribusi. Contoh: Counting Sort, Radix Sort, Bucket Sort.

Kompleksitas Sorting

Kompleksitas dihitung berdasarkan waktu (time complexity) dan ruang (space complexity).

Sorting berbasis perbandingan memiliki batas bawah $O(n \log n)$

$O(n \log n)$ untuk waktu rata-rata.

Sorting berbasis distribusi dapat mencapai $O(n)$

$O(n)$ dalam kasus tertentu.

Algoritma Sorting Populer

Bubble Sort: Mengurutkan dengan membandingkan elemen berurutan dan menukar jika tidak sesuai.

Selection Sort: Memilih elemen terkecil dan menempatkannya di posisi yang benar.

Insertion Sort: Memasukkan elemen ke dalam posisi yang tepat di bagian yang telah diurutkan.

Merge Sort: Membagi data menjadi dua bagian, mengurutkan masing-masing, dan menggabungkannya.

Quick Sort: Memilih elemen pivot untuk membagi data, kemudian mengurutkan setiap bagian secara rekursif.

Penerapan Sorting

Sorting sangat penting dalam berbagai bidang, seperti:

Database management.

Pengurutan daftar dokumen.

Optimalisasi algoritma lain, seperti pencarian dan penggabungan data

D. Implementasi

1. Bubble Sort adalah salah satu algoritma pengurutan yang paling sederhana.

Algoritma ini bekerja dengan cara membandingkan elemen bersebelahan dalam array dan menukar posisi mereka jika tidak dalam urutan yang benar. Proses ini diulangi hingga seluruh array terurut.

Cara Kerja Bubble Sort

Bandingkan elemen pertama dengan elemen kedua.

Jika elemen pertama lebih besar dari elemen kedua, tukar posisi mereka.

Lanjutkan proses ini untuk setiap pasangan elemen bersebelahan dalam array hingga akhir array.

Setelah satu iterasi, elemen terbesar akan berada di posisi terakhir.

Ulangi proses ini untuk elemen yang tersisa (tidak termasuk elemen terakhir yang sudah terurut).

Teruskan hingga tidak ada lagi elemen yang perlu ditukar.

Contoh

Misalkan kita memiliki array berikut: [5, 3, 8, 6, 2]

Prosesnya:

Iterasi 1:

Bandingkan 5 dan 3, tukar: [3, 5, 8, 6, 2]

Bandingkan 5 dan 8, tidak tukar: [3, 5, 8, 6, 2]

Bandingkan 8 dan 6, tukar: [3, 5, 6, 8, 2]

Bandingkan 8 dan 2, tukar: [3, 5, 6, 2, 8]

Elemen terakhir 8 sudah terurut.

Iterasi 2:

Bandingkan 3 dan 5, tidak tukar: [3, 5, 6, 2, 8]

Bandingkan 5 dan 6, tidak tukar: [3, 5, 6, 2, 8]

Bandingkan 6 dan 2, tukar: [3, 5, 2, 6, 8]

Elemen kedua terakhir 6 sudah terurut.

Iterasi 3:

Bandingkan 3 dan 5, tidak tukar: [3, 5, 2, 6, 8]

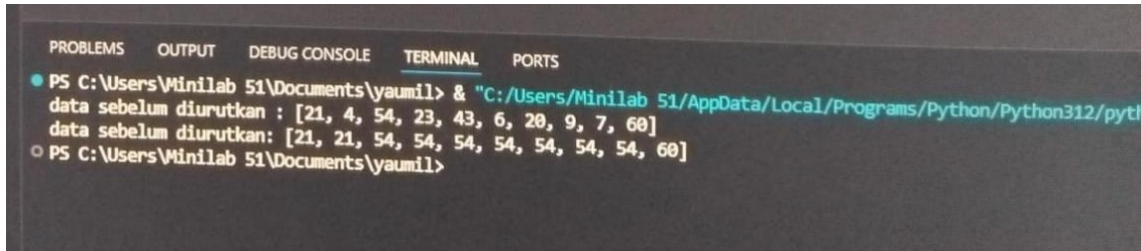
Bandingkan 5 dan 2, tukar: [3, 2, 5, 6, 8]

Elemen ketiga terakhir 5 sudah terurut.

Iterasi 4:

Bandingkan 3 dan 2, tukar: [2, 3, 5, 6, 8]

Array sudah terurut.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Minilab 51\Documents\yaumil> & "C:/Users/Minilab 51/AppData/Local/Programs/Python/Python312/pyt
data sebelum diurutkan : [21, 4, 54, 23, 43, 6, 20, 9, 7, 60]
data sebelum diurutkan: [21, 21, 54, 54, 54, 54, 54, 54, 54, 60]
PS C:\Users\Minilab 51\Documents\yaumil>
```

Gambar 1. Ouput file halo.py

2. Merge Sort adalah algoritma pengurutan berbasis divide and conquer yang membagi array menjadi dua bagian yang lebih kecil, mengurutkan kedua bagian tersebut secara rekursif, lalu menggabungkannya kembali menjadi array yang terurut.

Cara Kerja Merge Sort

Divide

Bagi array menjadi dua bagian hingga setiap bagian hanya terdiri dari satu elemen (karena array dengan satu elemen sudah pasti terurut).

Conquer

Urutkan kedua bagian secara rekursif.

Combine

Gabungkan dua bagian yang terurut menjadi satu array terurut.

Contoh

Misalkan kita ingin mengurutkan array berikut: [38, 27, 43, 3, 9, 82, 10]

Prosesnya adalah sebagai berikut:

Divide

Bagi array menjadi dua: [38, 27, 43] dan [3, 9, 82, 10].

Divide Lebih Lanjut

$[38, 27, 43] \rightarrow [38]$ dan $[27, 43]$

$[27, 43] \rightarrow [27]$ dan $[43]$

$[3, 9, 82, 10] \rightarrow [3, 9]$ dan $[82, 10]$

$[3, 9] \rightarrow [3]$ dan $[9]$

$[82, 10] \rightarrow [82]$ dan $[10]$

Conquer (Gabungkan)

$[27]$ dan $[43] \rightarrow [27, 43]$

$[38]$ dan $[27, 43] \rightarrow [27, 38, 43]$

$[3]$ dan $[9] \rightarrow [3, 9]$

$[82]$ dan $[10] \rightarrow [10, 82]$

$[3, 9]$ dan $[10, 82] \rightarrow [3, 9, 10, 82]$

Combine

$[27, 38, 43]$ dan $[3, 9, 10, 82] \rightarrow [3, 9, 10, 27, 38, 43, 82]$

Array terurut: $[3, 9, 10, 27, 38, 43, 82]$

3. Insertion Sort adalah algoritma pengurutan sederhana yang bekerja seperti cara kita menyortir kartu dalam permainan kartu. Elemen-elemen dari array diproses satu per satu, dengan masing-masing elemen dimasukkan ke posisi yang sesuai dalam bagian array yang sudah terurut.

Cara Kerja Insertion Sort

Anggap elemen pertama dari array sudah terurut.

Ambil elemen berikutnya dan bandingkan dengan elemen-elemen dalam bagian array yang sudah terurut.

Pindahkan elemen yang lebih besar ke kanan untuk memberikan ruang bagi elemen yang sedang diproses.

Masukkan elemen tersebut ke posisi yang sesuai.

Ulangi langkah ini untuk setiap elemen hingga seluruh array terurut.

Contoh

Misalkan kita memiliki array berikut:

[5, 3, 8, 6, 2]

Prosesnya:

Mulai dari elemen kedua:

Elemen 3 dibandingkan dengan 5, pindahkan 5 ke kanan: [5, 5, 8, 6, 2]

Masukkan 3 ke posisi yang benar: [3, 5, 8, 6, 2]

Proses elemen ketiga:

Elemen 8 sudah pada posisi yang benar karena lebih besar dari elemen sebelumnya: [3, 5, 8, 6, 2]

Proses elemen keempat:

Elemen 6 dibandingkan dengan 8, pindahkan 8 ke kanan: [3, 5, 8, 8, 2]

Masukkan 6 di antara 5 dan 8: [3, 5, 6, 8, 2]

Proses elemen kelima:

Elemen 2 dibandingkan dengan semua elemen sebelumnya, pindahkan 3, 5, 6, dan 8 ke kanan: [3, 3, 5, 6, 8]

Masukkan 2 ke posisi awal: [2, 3, 5, 6, 8]

4. Quick Sort adalah salah satu algoritma pengurutan berbasis divide and conquer yang sangat efisien. Algoritma ini bekerja dengan memilih satu elemen sebagai pivot, kemudian mempartisi array menjadi dua bagian: elemen yang lebih kecil dari pivot di sebelah kiri, dan elemen yang lebih besar dari pivot di sebelah kanan. Proses ini dilakukan secara rekursif hingga seluruh array terurut.

Cara Kerja Quick Sort

Pilih Pivot

Pilih satu elemen dari array sebagai pivot (biasanya elemen pertama, terakhir, atau elemen tengah).

Partisi Array

Tempatkan elemen yang lebih kecil dari pivot di sebelah kiri.

Tempatkan elemen yang lebih besar dari pivot di sebelah kanan.

Rekursif

Terapkan Quick Sort pada subarray kiri dan kanan secara rekursif.

Gabungkan

Setelah kedua subarray terurut, gabungkan keduanya dengan pivot untuk membentuk array yang terurut.

Contoh

Misalkan kita memiliki array berikut:

[8, 3, 1, 7, 0, 10, 2]

Prosesnya adalah sebagai berikut:

Pilih Pivot:

Pilih elemen terakhir sebagai pivot: 2.

Partisi Array:

Elemen lebih kecil dari 2: [1, 0]

Elemen lebih besar dari 2: [8, 3, 7, 10]

Array setelah partisi: [1, 0, 2, 8, 3, 7, 10]

Rekursif pada Subarray Kiri [1, 0]:

Pilih 0 sebagai pivot.

Partisi menjadi [0, 1].

Rekursif pada Subarray Kanan [8, 3, 7, 10]:

Pilih 10 sebagai pivot.

Partisi menjadi [8, 3, 7, 10].

Rekursif pada [8, 3, 7] dengan pivot 7.

Array terurut: [0, 1, 2, 3, 7, 8, 10].

E. Daftar Pustaka

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms (3rd ed.). MIT Press.

Knuth, D. E. (1997). The Art of Computer Programming, Volume 3: Sorting and Searching (2nd ed.). Addison-Wesley.

Sedgewick, R. (2011). Algorithms in Java. Addison-Wesley.

Hoare, C. A. R. (1962). "Quicksort". The Computer Journal, 5(1), 10–16.

Weiss, M. A. (2011). Data Structures and Algorithm Analysis in C++ (4th ed.). Pearson.

Goodrich, M. T., & Tamassia, R. (2010). Data Structures and Algorithms in Java (6th ed.). Wiley.