VISA: Journal of Visions and Ideas
Vol 4 No 3 (2024) 1197–1204 E-ISSN 2809-2058 P-ISSN 2809-2643
DOI: 47467/visa.v4i3.5409

Penerapan Algoritma Quick Sort untuk Menyortir Array Berdasarkan Kriteria Tertentu untuk Meningkatkan Efisiensi Komputasi

Afifah Naila Nasution[1], Pritiy Singgam[2], Ahmad Yusuf Al-Hafiz[3], Oka Alvansyah[4], Josua Deo Tampubolon[5]
[1,2,3,4,5]Program Studi Ilmu Komputer, Universitas Negeri Medan, Indonesia
afifahnaila90@gmail.com[1], pritymirota@gmail.com[2], nur23aisyah11@gmail.com[3], okaalv3@gmail.com[4], josuatampubolon30@gmail.com[5]

ABSTRAK

This research focuses on implementing the Quick Sort algorithm for sorting arrays based on specific criteria, with the aim of increasing computational efficiency in data management. Quick Sort was chosen because of its superiority in sorting data with an average time complexity of O(n log n), making it efficient for use on large datasets. This research applies this algorithm to student data sorted by NIM and name, and considers the grouping of student entry routes. The research results show that the implementation of Quick Sort in sorting student data not only speeds up the data processing process but also allows for more efficient processing in various data conditions.
Keywords: *quick sort, arrays, data sorting, algorithms, descriptive research, programming*

ABSTRAK

Penelitian ini berfokus pada implementasi algoritma Quick Sort untuk pengurutan array berdasarkan kriteria khusus, dengan tujuan meningkatkan efisiensi komputasi dalam pengelolaan data. Quick Sort dipilih karena keunggulannya dalam mengurutkan data dengan kompleksitas waktu rata-rata O(n log n), membuatnya efisien untuk digunakan pada dataset besar. Penelitian ini mengaplikasikan algoritma tersebut pada data mahasiswa yang diurutkan berdasarkan NIM dan nama, serta mempertimbangkan pengelompokan jalur masuk mahasiswa. Hasil penelitian menunjukkan bahwa implementasi Quick Sort pada pengurutan data mahasiswa tidak hanya mempercepat proses pengolahan data tetapi juga memungkinkan pemrosesan yang lebih efisien dalam berbagai kondisi data.
Kata kunci: quick sort, array, pengurutan data, algoritma, penelitian deskriptif, pemrograman

PENDAHULUAN

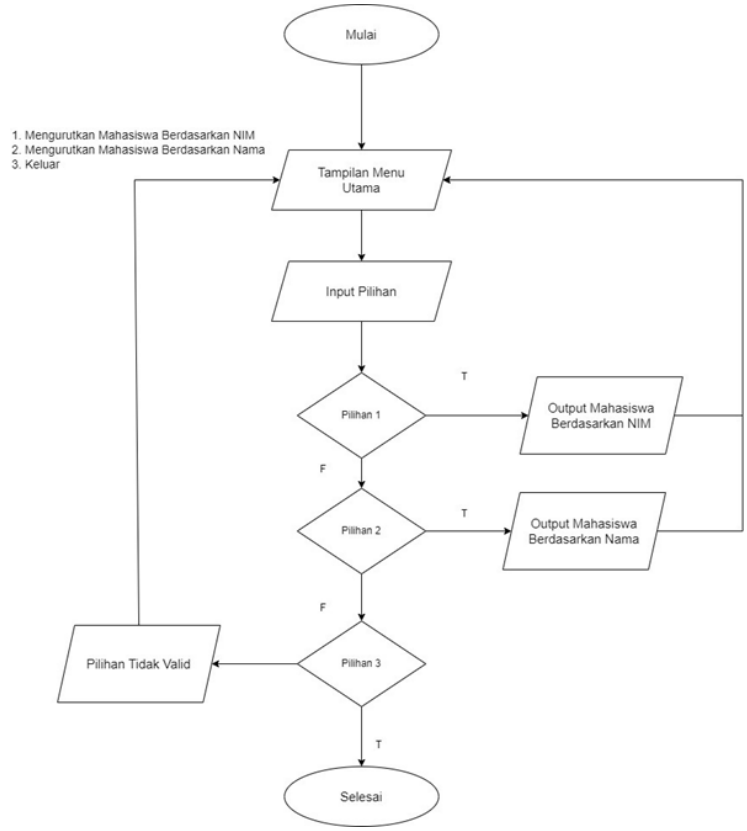
Pengurutan data memainkan peran penting dalam dunia komputasi, terutama dengan semakin meningkatnya jumlah data yang disimpan secara masif di komputer. Data yang tidak terorganisir akan menjadi sulit diakses dan dikelola, sehingga proses pengurutan menjadi esensial. Data yang terurut tidak hanya memudahkan pencarian, tetapi juga mempermudah pemeriksaan serta perbaikan kesalahan yang mungkin terjadi.

Penelitian ini memfokuskan pada algoritma pengurutan data seperti Quick Sort, yang digunakan untuk mengatur deretan angka dan huruf secara berurutan. Setiap algoritma memiliki keunggulan dan kelemahannya masing-masing, terutama dalam hal kecepatan pemrosesan, jumlah langkah yang dibutuhkan, serta penggunaan memori. Quick Sort merupakan algoritma yang paling efisien, di mana

VISA: Journal of Visions and Ideas
Vol 4 No 3 (2024) 1197–1204 E-ISSN 2809-2058 P-ISSN 2809-2643
DOI: 47467/visa.v4i3.5409

algoritma ini membutuhkan langkah lebih sedikit, mengonsumsi lebih sedikit memori, dan memproses data dengan lebih cepat.

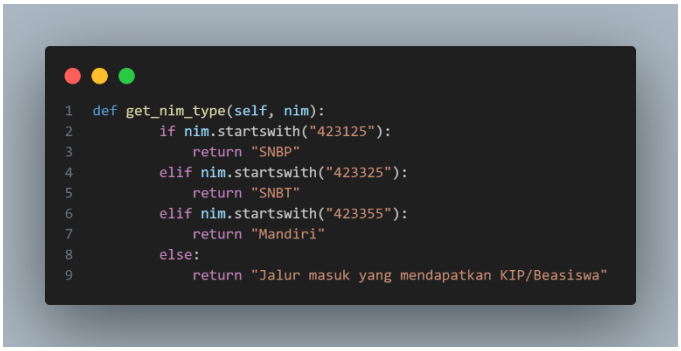
METODE PENELITIAN



1. User memulai program
2. Program menampilkan Menu Utama yang berisikan menu inputan yang berisi pilihan yaitu Pilihan 1) mengurutkan mahasiswa berdasarkan NIM, 2) Mengurutkan Mahasiswa berdasarkan Nama, 3) Menu Keluar Program.
3. Apabila User memilih menu 1 maka program akan menampilkan pengurutan Mahasiswa berdasarkan NIM dari yang terkecil sampai yang terbesar, jika User Memilih menu 2 maka program akan menampilkan pengurutan Mahasiswa berdasarkan Nama sesuai abjad, jika User memilih menu 3 maka user akan keluar dari Program.
4. Apabila inputan yang diberikan user tidak sesuai dengan menu yang tertera maka program akan menampilkan "Pilihan Tidak Valid" dan program Kembali pada menu utama.
5. Setelah program menampilkan menu 1 dan 2 maka program akan Kembali ke bagian menu Utama.

VISA: Journal of Visions and Ideas
Vol 4 No 3 (2024) 1197–1204 E-ISSN 2809-2058 P-ISSN 2809-2643
DOI: 47467/visa.v4i3.5409

HASIL DAN PEMBAHASAN



```
def get_nim_type(self, nim):
        if nim.startswith("423125"):
            return "SNBP"
        elif nim.startswith("423325"):
            return "SNBT"
        elif nim.startswith("423355"):
            return "Mandiri"
        else:
            return "Jalur masuk yang mendapatkan KIP/Beasiswa"
```

Fungsi ini bernama get_nim_type, dan menerima dua parameter self (yang biasanya merujuk pada objek kelas yang sedang diproses) dan nim (yang merupakan nomor induk mahasiswa yang ingin diperiksa).

if nim.startswith("423125"):
return "SNBP"

Baris ini memeriksa apakah nim diawali dengan string "423125". Jika benar, fungsi akan mengembalikan string "SNBP", yang mungkin merujuk pada suatu jenis program pendidikan.

elif nim.startswith("423325"):
return "SNBT"

Jika kondisi sebelumnya tidak terpenuhi, baris ini memeriksa apakah nim diawali dengan "423325". Jika ya, fungsi akan mengembalikan "SNBT".

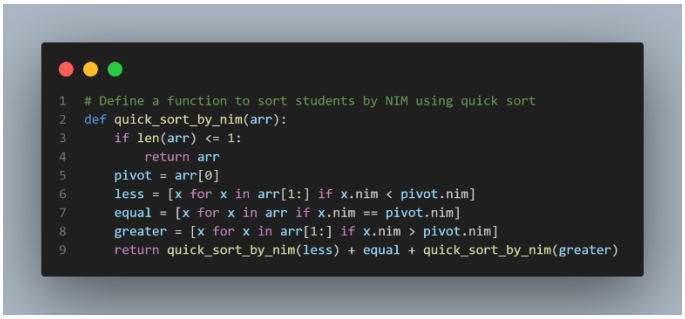
elif nim.startswith("423355"):
return "Mandiri"

Sekali lagi, jika kedua kondisi sebelumnya tidak terpenuhi, kode ini akan memeriksa apakah nim diawali dengan "423355". Jika ya, maka fungsi akan mengembalikan "Mandiri".

else:
return "Jalur masuk yang mendapatkan KIP/Beasiswa"

Jika nim tidak memenuhi salah satu dari tiga kondisi sebelumnya, maka fungsi akan mengembalikan string "Jalur masuk yang mendapatkan KIP/Beasiswa". Ini menunjukkan bahwa NIM tersebut mungkin berasal dari jalur penerimaan khusus.

VISA: Journal of Visions and Ideas
Vol 4 No 3 (2024) 1197–1204 E-ISSN 2809-2058 P-ISSN 2809-2643
DOI: 47467/visa.v4i3.5409



```python
# Define a function to sort students by NIM using quick sort
def quick_sort_by_nim(arr):
    if len(arr) <= 1:
        return arr
    pivot = arr[0]
    less = [x for x in arr[1:] if x.nim < pivot.nim]
    equal = [x for x in arr if x.nim == pivot.nim]
    greater = [x for x in arr[1:] if x.nim > pivot.nim]
    return quick_sort_by_nim(less) + equal + quick_sort_by_nim(greater)
```

Definisi Fungsi Kode ini mendefinisikan sebuah fungsi bernama quick_sort_by_nim yang mengambil array arr sebagai input. Fungsi ini merupakan implementasi dari algoritma quicksort.

def quick_sort_by_nim(arr):

Fungsi ini dinamakan *quick_sort_by_nim*, yang menerima satu parameter arr, yaitu daftar objek yang akan diurutkan.

if len(arr) <= 1:
return arr

Jika panjang arr kurang dari atau sama dengan 1, maka sudah terurut (karena tidak ada atau hanya ada satu elemen). Fungsi akan mengembalikan arr itu sendiri.

pivot = arr[0]

Elemen pertama dari arr dipilih sebagai pivot. Ini adalah elemen yang akan digunakan sebagai referensi untuk membandingkan elemen lainnya.

less = [x for x in arr[1:] if x.nim < pivot.nim]

Daftar ini berisi semua objek dalam arr (kecuali pivot) yang memiliki atribut nim lebih kecil dari pivot.nim.

equal = [x for x in arr if x.nim == pivot.nim]

Daftar ini berisi semua objek dalam arr yang memiliki atribut nim sama dengan pivot.nim.

greater = [x for x in arr[1:] if x.nim > pivot.nim]

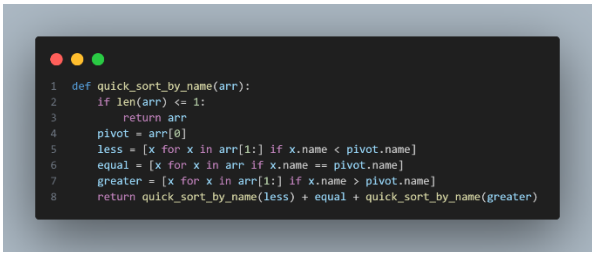
Daftar ini berisi semua objek dalam arr (kecuali pivot) yang memiliki atribut nim lebih besar dari pivot.nim.

return quick_sort_by_nim(less) + equal + quick_sort_by_nim(greater)

Fungsi ini kemudian memanggil dirinya sendiri secara rekursif untuk mengurutkan daftar less dan greater.

Hasil pengurutan dari less, ditambah dengan elemen equal, dan hasil pengurutan dari greater dikembalikan sebagai output.

VISA: Journal of Visions and Ideas
Vol 4 No 3 (2024) 1197–1204 E-ISSN 2809-2058 P-ISSN 2809-2643
DOI: 47467/visa.v4i3.5409



def quick_sort_by_name(arr):

Fungsi ini dinamakan quick_sort_by_name, dan menerima satu parameter arr, yaitu daftar objek yang akan diurutkan.

if len(arr) <= 1:
return arr

Jika panjang arr kurang dari atau sama dengan 1, maka daftar tersebut sudah terurut (karena tidak ada elemen atau hanya satu elemen). Fungsi akan mengembalikan arr itu sendiri.

pivot = arr[0]

Elemen pertama dari arr dipilih sebagai pivot. Ini adalah elemen yang akan digunakan sebagai referensi untuk membandingkan elemen lainnya.

less = [x for x in arr[1:] if x.name < pivot.name]

Daftar ini berisi semua objek dalam arr (kecuali pivot) yang memiliki atribut name lebih kecil dari pivot.name.

equal = [x for x in arr if x.name == pivot.name]

Daftar ini berisi semua objek dalam arr yang memiliki atribut name sama dengan pivot.name.

greater = [x for x in arr[1:] if x.name > pivot.name]

Daftar ini berisi semua objek dalam arr (kecuali pivot) yang memiliki atribut name lebih besar dari pivot.name.

return quick_sort_by_name(less) + equal + quick_sort_by_name(greater)

Fungsi ini kemudian memanggil dirinya sendiri secara rekursif untuk mengurutkan daftar less dan greater.

Hasil pengurutan dari less, ditambah dengan elemen equal, dan hasil pengurutan dari greater dikembalikan sebagai output.

Adapun hasil dari penelitian ini sesuai dengan tujuan awal peneliti, yaitu untuk membangun sistem yang mampu menyimpan dan menampilkan data mahasiswa, termasuk nama dan NIM (Nomor Induk Mahasiswa) secara terstruktur dan mengimplementasikan algoritma Array dan Quick Sort untuk mengurutkan data

Vol 4 No 3 (2024) 1197–1204 E-ISSN 2809-2058 P-ISSN 2809-2643
DOI: 47467/visa.v4i3.5409

mahasiswa berdasarkan NIM atau nama. Berdasarkan rumusan masalah dalam penelitian ini, dapat diambil jawaban yaitu:

1. Pengimplementasian algoritma quick sort secara efisien untuk menyortir data mahasiswa berdasarkan NIM dan nama
   Algoritma quick sort diimplementasikan secara efisien dengan cara memisahkan data mahasiswa berdasarkan nilai NIM atau nama sebagai pivot, kemudian mengelompokkan data yang lebih kecil, sama, dan lebih besar dari pivot. Proses ini dilakukan secara rekursif hingga seluruh data terurut. Pada kode di atas, dua fungsi quick_sort_by_nim dan quick_sort_by_name digunakan untuk menyortir data berdasarkan NIM dan nama, dengan masing-masing mahasiswa direpresentasikan oleh objek dari kelas Student.
2. Pengelompokan jalur masuk mahasiswa berdasarkan prefiks NIM memengaruhi struktur dan proses pengolahan data
   Jalur masuk mahasiswa diidentifikasi berdasarkan prefiks NIM, yang kemudian dikategorikan dalam berbagai jalur seperti SNBP, SNBT, Mandiri, dan KIP/Beasiswa. Pengelompokkan ini memudahkan analisis dan pengelolaan data mahasiswa, karena setiap mahasiswa secara otomatis dikelompokkan berdasarkan prefiks NIM yang berbeda, seperti terlihat pada metode get_nim_type. Hal ini memengaruhi struktur data dengan menambah atribut jalur masuk, yang dapat digunakan untuk analisis lebih lanjut terkait karakteristik atau pengelompokan mahasiswa.
3. Desain sistem informasi dapat mendukung tampilan data mahasiswa yang disortir berdasarkan NIM dan nama, serta jalur masuk mereka
   Desain sistem informasi yang baik dapat memfasilitasi tampilan data yang disortir berdasarkan NIM dan nama secara interaktif. Sistem tersebut dapat menampilkan hasil pengurutan dengan menggunakan algoritma quick sort pada antarmuka pengguna. Informasi seperti nama, NIM, dan jalur masuk dapat ditampilkan dalam format yang jelas, seperti yang ditunjukkan pada fungsi display_students. Setiap entri mahasiswa menampilkan nama, NIM, serta jalur masuk, yang memudahkan pengguna untuk melihat hasil pengelompokan dan pengurutan secara efisien.

Adapun kekuatan dan kelemahan yang dijumpai dari program yang penleiti buat yaitu:

Kelebihan dari Program:

1. Penggunaan Algoritma Quick Sort: Program ini menggunakan algoritma Quick Sort, yang dikenal sebagai salah satu algoritma pengurutan paling efisien. Ini sangat cocok untuk mengurutkan data mahasiswa dalam jumlah besar, baik berdasarkan NIM maupun nama.
2. Struktur Data yang Jelas: Program menggunakan kelas Student untuk merepresentasikan mahasiswa, yang memudahkan pengelolaan dan pemrosesan data. Setiap objek mahasiswa memiliki atribut nama, NIM, dan jalur masuk, membuat data lebih terstruktur.
3. Fungsi Pengurutan Terpisah: Program memisahkan fungsi untuk mengurutkan berdasarkan NIM (quick_sort_by_nim) dan nama

Vol 4 No 3 (2024) 1197–1204 E-ISSN 2809-2058 P-ISSN 2809-2643
DOI: 47467/visa.v4i3.5409

(quick_sort_by_name), sehingga fleksibel dan mudah dimodifikasi jika diperlukan pengurutan dengan kriteria lain di masa depan.
4. Pengelompokan Berdasarkan Jalur Masuk: Dengan adanya metode get_nim_type, program bisa mengidentifikasi jalur masuk mahasiswa berdasarkan prefiks NIM, memberikan informasi tambahan yang berguna saat menampilkan data.
5. Readable Output: Data mahasiswa ditampilkan dengan format yang mudah dibaca, termasuk nama, NIM, dan jalur masuk, sehingga program ini cocok untuk laporan atau tampilan langsung ke pengguna.

Kelemahan dari Program:
1. Efisiensi Ruang *(Space Efficiency)*: Quick Sort memiliki efisiensi waktu yang baik, tetapi karena implementasi rekursifnya, algoritma ini bisa menggunakan banyak memori untuk daftar yang besar, terutama jika daftar sangat tidak teratur, karena dapat menyebabkan masalah dengan kedalaman rekursi.
2. Penanganan Kasus Jalur Masuk Terbatas: Pengelompokan jalur masuk hanya didasarkan pada prefiks tertentu di NIM. Jika ada jalur masuk baru atau perubahan format NIM di masa mendatang, program ini perlu diperbarui secara manual untuk mendukungnya.
3. Kurangnya Validasi Input: Program ini tidak memiliki validasi untuk memastikan NIM yang dimasukkan benar atau sesuai dengan format yang diharapkan. Jika ada NIM yang tidak sesuai prefiks, program akan menetapkan jalur yang tidak akurat atau tidak memberikan informasi jalur yang tepat.
4. Tidak Dilengkapi dengan Antarmuka Pengguna: Saat ini, program hanya menampilkan data di terminal/*console*. Untuk aplikasi nyata, mungkin akan lebih baik jika data ditampilkan melalui antarmuka pengguna (misalnya web atau aplikasi desktop) untuk memperbaiki pengalaman pengguna.
5. Skalabilitas Pengelompokan Jalur Masuk: Meskipun jalur masuk dikelompokkan berdasarkan prefiks NIM, program tidak mendukung pengelompokan yang lebih fleksibel atau kompleks. Jika pengelompokan lebih rumit diperlukan, kode perlu dimodifikasi secara manual.
6. Kurang Fleksibel dalam Metode Pengurutan: Program hanya mendukung pengurutan berdasarkan satu atribut pada satu waktu (entah NIM atau nama). Jika ingin mengurutkan berdasarkan kombinasi kriteria (misalnya NIM dulu, lalu nama), diperlukan modifikasi tambahan.

KESIMPULAN DAN SARAN

Kesimpulan

Implementasi algoritma Quick Sort untuk pengurutan data mahasiswa berdasarkan NIM dan nama terbukti efisien dalam memproses data berukuran besar. Algoritma ini mampu mengurangi waktu komputasi dengan cara membagi data menjadi subkelompok menggunakan metode *divide and conquer*, sehingga

Vol 4 No 3 (2024) 1197–1204 E-ISSN 2809-2058 P-ISSN 2809-2643
DOI: 47467/visa.v4i3.5409

pengurutan dapat dilakukan lebih cepat dibandingkan algoritma lainnya seperti *bubble sort*. Penggunaan Quick Sort dalam pengelompokan jalur masuk mahasiswa berdasarkan prefiks NIM juga membantu mempermudah klasifikasi data. Meskipun begitu, pemilihan elemen pivot yang tepat sangat penting untuk menghindari terjadinya inefisiensi pada kasus tertentu.

Saran

Adapun saran yang dapat peneliti berikan untuk penelitian selanjutnya yang berdasarkan hasil yang sudah didapati di dalam implementasi tersebut yaitu diharapkan untuk menggunakan metode pemilihan pivot yang lebih canggih, seperti median-of-three, untuk mengotimalkan performa Quick Sort pada data yang lebih besar dan tidak teratur. Penelitian selanjutnya diharapkan dapat membandingkan algoritma Quick Sort dengan algoritma pengurutan lainnya seperti Marge Sort atau Heap Sort, untuk mengevaluasi efisiensi dalam berbagai jenis data. Pengembangan lebih lanjut dapat dilakukan dengan integrasi antara Quick Sort dengan sistem antarmuka pengguna (Interface) yang lebih interaktif, seperti aplikasi berbasis web atau desktop, untuk meningkatkan pengalaman pengguna.

DAFTAR PUSTAKA

Poetra, D. R. (2022). Performa Algoritma Bubble Sort dan Quick Sort pada Framework Flutter dan Dart SDK (Studi Kasus Aplikasi E-Commerce). *JATISI (Jurnal Teknik Informatika Dan Sistem Informasi),* 9(2), 806-816.

Rumapea, Y. Y. (2017). Analisis Perbandingan Metode Algoritma Quick Sort dan Merge Sort dalam Pengurutan Data terhadap Jumlah Langkah dan Waktu. *METHODIKA: Jurnal Teknik Informatika dan Sistem Informasi,* 3(2), 5-9.

Zulfa, M. L (2022). Analisis Perbandingan Algoritma Bubble Sort, Shell Sort, dan Quick Sort dalam Mengurutkan Baris Data Acak Menggunakan Bahasa Java. *Jurnal Ilmiah Wahana Pendidikan,* 8(13), 237-246.