"MENINGKATKAN EFISIENSI ALGORITMA DENGAN MEMANFAATKAN HIGHER ORDER FUNCTION DALAM STRUKTUR DATA"

Elia Meylani Simanjuntak[1], Priska Silvia Ferantiana[2], Residen Nusantara[3], Anwar Muslim[4], Muhammad Zaky Zaiddan[5]
Program Studi Sains Data, Fakultas Sains, Institut Teknologi Sumatera
email : *elia.122450026@student.itera.ac.id*, *priska.122450053@student.itera.ac.id*,
residen.122450080@student.itera.ac.id, *anwar.122450117@student.itera.ac.id*,
muhammad.122450119@student.itera.ac.id

Abstrak

Penelitian ini ingin membuktikan bahwa penggunaan *higher order function* dalam struktur data dapat meningkatkan efisiensi algoritma. *Higher order function* merupakan salah satu konsep fundamental dalam bahasa pemrograman yang memungkinkan programmer untuk mengoperasikan fungsi-fungsi sebagai data, sehingga dapat memberikan fleksibilitas dalam penggunaan fungsi. Dengan penggunaan *higher order function*, waktu eksekusi dapat lebih ditekan dan pada algoritma tertentu dapat mengurangi kompleksitas kode sehingga lebih mudah dibaca. Setelah menggunakan konsep *higher order function* dalam menghitung jumlah elemen dengan kondisi tertentu, filtering kata, dan transformasi data dengan fungsi map, didapatkan bahwa hasil keluaran kode *higher order function* lebih efisien dan dapat disesuaikan dengan lebih fleksibel dibandingkan fungsi yang lebih sederhana. Ini juga menjelaskan bahwa penggunaan *higher order function* dapat meningkatkan efisiensi algoritma.

Kata Kunci: *Higher order function, Algoritma, Pemrograman berbasis fungsi*

PENDAHULUAN

1) Latar Belakang

Seiring kemajuan dan kebutuhan pemrosesan data, peningkatan performa algoritma menjadi sangat penting. Dalam beberapa tahun terakhir, telah banyak dikembangkan berbagai macam pendekatan pemrograman untuk memanipulasi dan mengelola data dengan lebih efisien, salah satunya adalah pendekatan pemrograman fungsional yang memiliki popularitas yang signifikan dalam pengembangan perangkat lunak. Pemrograman fungsional memiliki paradigma yang berbeda dalam menangani data, yaitu fokus pada fungsi yang tidak berubah (immutable) dan pemrosesan data yang deklaratif.

Dalam dinamika struktur data, penerapan higher order functions dapat menghasilkan kode yang lebih bersih dan lebih mudah dimengerti, secara bersamaan meningkatkan kinerja algoritma. Dengan memanfaatkan higher order functions, kita dapat memanipulasi data dengan cara yang lebih deklaratif, sehingga mengurangi kompleksitas dan meningkatkan kejelasan dari kode yang dibuat. Dalam meningkatkan efisien algoritma, high order function menggabungkan fungsi-fungsi menjadi suatu komponen yang dapat digunakan lagi. Oleh karena itu, high order function lebih fokus untuk memecahkan masalah yang lebih besar daripada masalah yang mendetail.

2) Rumusan Masalah

a. Bagaimana konsep dasar dari high order function?
b. Bagaimana penerapan konsep high order function dalam meningkatkan efisiensi algoritma struktur data?

3) Tujuan
    a. Mengetahui konsep dasar dari high order function.
    b. Menerapkan konsep high order function dalam meningkatkan efisiensi algoritma struktur data.

METODE

1) List Comprehension

List comprehension adalah teknik yang digunakan dalam pemrograman Python untuk membuat daftar baru dengan cara yang ringkas dan ekspresif. List comprehension memungkinkan kita membuat daftar dengan melakukan iterasi pada iterable (daftar, tupel, string, dll.) dan menerapkan ekspresi ke setiap elemen. Hal ini memungkinkan untuk membuat daftar baru yang difilter, dimodifikasi, atau diubah berdasarkan kriteria tertentu dalam satu baris kode yang mudah dibaca.

2) Filter

Metode filter adalah fungsi bawaan Python yang digunakan untuk memfilter elemen objek yang dapat diubah (seperti daftar, tuple, atau string) berdasarkan kondisi tertentu yang ditentukan dalam fungsi tersebut. Fungsi filter mengembalikan iterator yang hanya mengembalikan elemen yang memenuhi kondisi tertentu. Metode filter memungkinkan kita dengan mudah memfilter dan memilih elemen yang cocok dengan kriteria tertentu, seperti kondisi matematis atau kriteria yang ditentukan pengguna, memungkinkan untuk melakukan pemrosesan data yang efisien dan terstruktur dalam bahasa Python.

3) Map

Metode Map adalah fungsi bawaan Python yang digunakan untuk menerapkan fungsi ke setiap elemen dalam iterable (daftar, tupel, string, dll.) dan membuat iterator yang menghasilkan nilai yang merupakan hasil penerapan fungsi kembali. Metode map memungkinkan kita mengubah atau memodifikasi elemen dalam struktur data secara efisien dan kompak dengan menerapkan operasi yang sama ke setiap elemen. Hal ini memungkinkan kita melakukan transformasi data seragam dalam jumlah besar, serta meningkatkan kejelasan dan konsistensi kode dalam bahasa Python.

HASIL DAN PEMBAHASAN

1. Higher order function untuk menghitung jumlah elemen yang memenuhi kondisi tertentu.

```
Jumlah bilangan genap dalam data: 5
```

Gambar 1. Output HOF untuk menghitung jumlah elemen.

Kode yang diberikan pada output berikut adalah pengimplementasian dari fitur higher order dalam Python yang digunakan untuk menghitung jumlah bilangan genap dalam sebuah list. Dua parameter diberikan kepada fungsi hitung_elemen: data, yang merupakan daftar, dan kondisi, yang merupakan fungsi yang menentukan kondisi yang harus dipenuhi oleh elemen. Operator modulo digunakan dalam contoh ini untuk menentukan apakah suatu bilangan adalah genap atau tidak dengan menggunakan fungsi kondisi_genap. Setelah perhitungan selesai, hasilnya dicetak dan menampilkan jumlah bilangan genap dalam daftar data; dalam kasus ini, ini menghasilkan hasil "Jumlah bilangan genap dalam data: 5".

2. Filtering Berdasarkan Kriteria

```
Kata dengan panjang lebih dari 5 karakter: ['pisang', 'anggur']
```

Gambar 2. Output Metode Filtering

Kode ini adalah fungsi filter_data yang menerapkan filter pada list data sesuai dengan kriteria tertentu. Fungsi ini menerima dua argumen: data, yang merupakan list yang akan difilter, dan kondisi, yang merupakan fungsi atau ekspresi lambda yang menentukan kriteria filter. Beberapa string seperti "apel", "jeruk", "nanas", "pisang", dan "anggur" termasuk dalam data dalam kode tersebut. Dengan menggunakan ekspresi lambda x: len(x) > 5, fungsi filter_data dapat digunakan untuk memfilter kata-kata dalam list data yang memiliki panjang lebih dari lima karakter. Kode ini menghasilkan kata-kata dalam daftar data yang kode tersebut menghasilkan kata-kata dalam daftar data yang memiliki panjang lebih dari lima karakter, seperti "pisang" dan "anggur".

3. Transformasi Data dengan Mapping

```
Data setelah dipangkatkan dua: [1, 4, 9, 16, 25]
```

Gambar 3. Output Metode Map

Dalam kode ini, fungsi map_data mengubah setiap elemen dalam list data menggunakan fungsi atau ekspresi lambda yang ditetapkan oleh argumen transformasi. Dalam hal ini, list data terdiri dari bilangan bulat dari 1 hingga 5, dan fungsi map_data mengangkat setiap elemen dalam list tersebut dengan pangkat dua, yang didefinisikan oleh ekspresi lambda x: x ** 2. Kode tersebut menghasilkan daftar baru yang mengandung setiap elemen dari daftar data yang telah dipangkat dua, yaitu [1, 4, 9, 16, 25]. Karena fungsi map_data mengambil alih proses iterasi dan transformasi, kita dapat dengan mudah melakukan transformasi pada setiap elemen dalam daftar tanpa harus melakukan loop.

KESIMPULAN

Berdasarkan percobaan menghitung jumlah elemen dengan kondisi tertentu, filtering kata, dan transformasi data dengan fungsi map, didapatkan bahwa hasil keluaran menggunakan algoritma yang disertai *higher order function* lebih efisien dan dapat disesuaikan dengan lebih fleksibel dibandingkan fungsi yang lebih sederhana. Selain itu, kode yang menggunakan konsep *higher order function* cenderung lebih mudah dipelihara dan diperluas karena memungkinkan untuk penyesuaian fungsionalitas melalui parameter fungsi yang dapat diubah.

Dengan hasil keluaran yang didapatkan, ini menjelaskan sekali lagi bahwa penggunaan higher order function dapat meningkatkan efisiensi dari suatu algoritma. Dalam pemrograman fungsional, konsep dan penerapan fungsi tingkat tinggi dapat secara signifikan meningkatkan efisiensi algoritma struktur data. Dengan demikian, pemahaman yang baik tentang konsep dan penerapan fungsi tingkat tinggi akan memungkinkan kita untuk menghasilkan kode yang lebih bersih, mudah dimengerti, dan memiliki kinerja yang lebih baik dalam pemrosesan dan manipulasi data. Fungsi tingkat tinggi juga memungkinkan pendekatan deklaratif dalam pemrograman, yang mengurangi kompleksitas dan meningkatkan kejelasan kode.