**Perbandingan Efisiensi Algoritma Iteratif dan Rekursif untuk Menghitung Jumlah Digit**

**dalam Bilangan**

**Mata Kuliah Analisis Kompleksitas Algoritma**

**A logo with a book in the middle

Description automatically generated**

Disusun oleh:

1. Bramantya H. Yoga (103012300257)

2. M. Paksi Pratama (103012300432)

**PROGRAM STUDI S1 INFORMATIKA**

**FAKULTAS INFORMATIKA**

**UNIVERSITAS TELKOM**

**BANDUNG**

**2024**

1. **Deskripsi Studi Kasus**

Masalah yang diangkat dalam studi kasus ini adalah bagaimana menghitung jumlah digit dalam sebuah bilangan secara efisien menggunakan dua pendekatan algoritma, yaitu iteratif dan rekursif. Permasalahan ini sering ditemukan dalam pemrosesan data numerik, seperti analisis data statistik, penghitungan ID, atau data transaksi besar.

1. **Deskripsi Algoritma**
2. **Algoritma Iteratif**

Iterasi dalam algoritma adalah proses perulangan dalam suatu prosedur atau fungsi, terutama dalam bentuk loop. Fungsi ini memiliki ciri dimana adanya nilai variabel yang terus berubah selama terjadi perulangan. Fungsi iteratif berarti fungsi yang menggunakan konsep iterasi dalam prosesnya.

**Implementasi Algoritma Iteratif:**

// Fungsi iteratif untuk menghitung jumlah digit

int countDigitsIterative(int n) {

int count = 0;

while (n != 0) {

n /= 10;

count++;

}

return count;

}

1. **Algoritma Rekursif**

Dalam dunia pemrograman, rekursi dapat diimplementasikan dalam fungsi yang memanggil dirinya sendiri. Konsep pemrograman fungsi rekursif sebenarnya mirip dengan konsep fungsi rekursif dalam matematika yaitu sebagai berikut:

1. Menentukan kasus penyetop atau kasus dasar dimana pemanggilan rekursif tidak lagi diperlukan karena solusinya sudah diperoleh)  basis

2. Menerapkan suatu langkah untuk menggiring kasus kompleks ke kasus penyetopnya dengan metode yang mencerminkan fungsinya → rekurens.

**Implementasi Algoritma Rekursif:**

// Fungsi rekursif untuk menghitung jumlah digit

int countDigitsRecursive(int n) {

if (n == 0) {

return 0;

} else {

return 1 + countDigitsRecursive(n / 10);

}

}

1. **Grafik Perbandingan *Running Time***

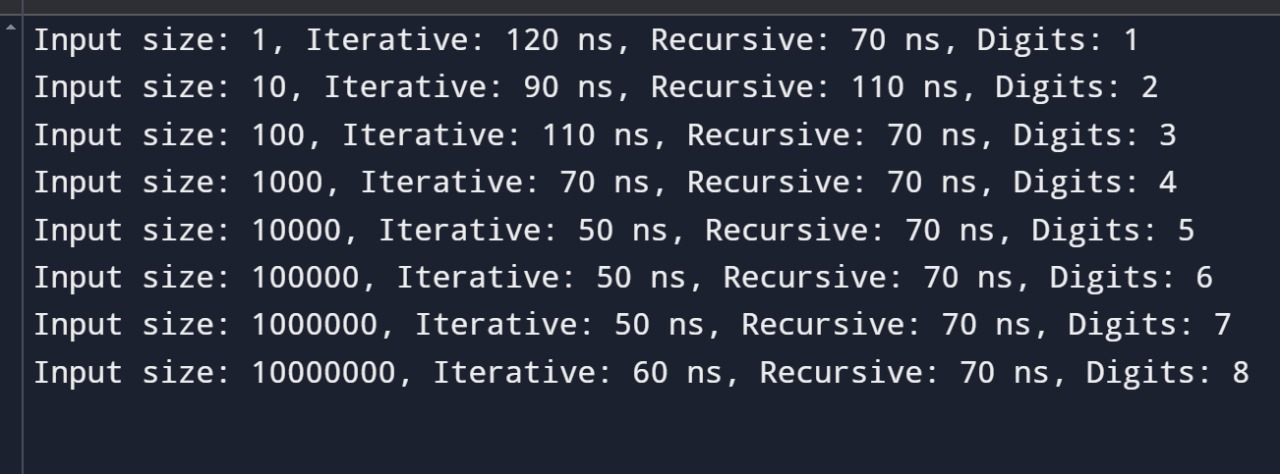
Pada bagian ini, dilakukan pengukuran dan perbandingan waktu eksekusi antara dua algoritma, yaitu algoritma iteratif dan algoritma rekursif, dalam menghitung jumlah digit sebuah bilangan. Setiap algoritma diuji pada berbagai ukuran input untuk melihat bagaimana perbedaan cara kerja kedua algoritma mempengaruhi waktu eksekusinya. Eksperimen ini mengukur waktu yang dibutuhkan oleh kedua algoritma untuk menghitung jumlah digit pada angka dengan jumlah digit yang berbeda. Waktu eksekusi diukur dalam satuan nanosekon untuk memberikan gambaran yang lebih detail mengenai performa masing-masing algoritma. Untuk melakukan eksperimen ini, digunakan kode berikut untuk mengukur waktu eksekusi kedua algoritma pada berbagai ukuran input:

A screen shot of a computer code

Description automatically generated

A screenshot of a computer code

Description automatically generated

******

**Grafik Running Time:**

**A graph with a line and a red line

Description automatically generated**

Grafik menunjukkan bahwa algoritma iteratif (biru) secara konsisten lebih cepat daripada algoritma rekursif (merah) untuk semua ukuran input. Perbedaan waktu eksekusi ini menjadi lebih terlihat seiring bertambahnya jumlah digit.

1. **Analisis Perbandingan Kedua Algoritma**

Jumlah digit dalam bilangan n dapat dihitung dengan ⌊⌋ + 1. Hal ini menunjukkan bahwa jumlah langkah atau iterasi yang diperlukan untuk menghitung jumlah digit bersifat logaritmik terhadap 𝑛. Oleh karena itu, kompleksitas waktu untuk algoritma iteratif maupun rekursif adalah: , di mana 𝑛 merupakan bilangan yang dihitung.

1. **Kompleksitas Waktu**

* **Iteratif:**

Dalam algoritma ini, kita melakukan operasi pembagian hingga n mencapai 0. Setiap pembagian mengurangi jumlah digit angka n. Oleh karena itu, jumlah iterasi yang diperlukan untuk mencapai 0 adalah O(), di mana n adalah angka yang dihitung. Ini berarti bahwa jumlah langkah yang dibutuhkan bergantung pada jumlah digit dalam angka n.

**Best Case**: Dalam **best case**, jika angka yang diberikan adalah 0, maka algoritma hanya akan melakukan satu langkah untuk keluar dari loop karena n langsung memenuhi kondisi n == 0. Jadi, dalam kasus ini, kompleksitas waktu adalah O(1), yang berarti hanya memerlukan waktu konstan.

**Worst Case**:

Dalam **worst case**, misalnya jika angka n adalah angka yang sangat besar seperti 999999, algoritma harus melakukan pembagian berulang hingga semua digit terhapus. Dalam hal ini, jumlah iterasi yang diperlukan adalah O(), di mana jumlah digit dalam angka n menentukan banyaknya langkah yang perlu dilakukan.

* **Rekursif:**

Sama seperti algoritma iteratif, algoritma rekursif juga akan melakukan pembagian hingga n menjadi 0, dan pada setiap pemanggilan fungsi mengurangi satu digit dari n. Oleh karena itu, kompleksitas waktu untuk algoritma rekursif juga O().

**Best Case**: Pada **best case**, ketika angka yang diberikan adalah 0, algoritma akan segera mencapai kondisi dasar dan berhenti tanpa melakukan pemanggilan rekursif lebih lanjut. Dalam hal ini, waktu eksekusinya adalah O(1), yang berarti hanya memerlukan satu langkah rekursif untuk mencapai hasil.

**Worst Case**: Pada **worst case**, jika n adalah angka yang sangat besar, algoritma akan terus memanggil dirinya sendiri untuk membagi n dengan 10 sampai akhirnya n menjadi 0. Jumlah pemanggilan rekursif yang diperlukan adalah O(), yang mengindikasikan bahwa algoritma ini juga memerlukan waktu yang tergantung pada jumlah digit dalam n.

1. **Grafik**

* Iteratif: Grafik menunjukkan bahwa waktu eksekusi algoritma iteratif meningkat lebih lambat dibandingkan dengan rekursif.
* Rekursif: Waktu eksekusi algoritma rekursif meningkat lebih tajam seiring dengan pertumbuhan ukuran input.

1. **Running Time**

* Iteratif: Hasil eksperimen menunjukkan waktu eksekusi yang lebih rendah dibandingkan rekursif, bahkan untuk input yang sangat besar. Hal ini dikarenakan iterasi menggunakan sumber daya lebih sedikit, tanpa perlu menyimpan informasi di stack.
* Rekursif: Waktu eksekusi meningkat lebih signifikan dibandingkan iteratif seiring bertambahnya ukuran input. Pemanggilan fungsi berulang kali menambah overhead di stack, sehingga memperlambat proses. Pada input yang sangat besar, algoritma ini berpotensi mengalami stack overflow.

1. **Penggunaan Memori**

* Iteratif: Menggunakan memori secara efisien karena seluruh operasi dilakukan dalam loop tunggal tanpa memanfaatkan call stack. Hal ini menjadikan iteratif lebih cocok untuk masalah yang membutuhkan pemrosesan data dalam skala besar.
* Rekursif: Memiliki overhead memori yang tinggi karena setiap pemanggilan fungsi memerlukan ruang tambahan di stack. Dalam kasus ekstrem, seperti input dengan jutaan digit, penggunaan algoritma rekursif dapat menyebabkan kehabisan memori.

1. **Referensi**
2. Munir, R. (2008). Makalah 0809-079: Analisis dan aplikasi algoritma.

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2008-2009/Makalah2008/Makalah0809-079.pdf>

1. Mulyadi, M. (2020). Analisis algoritma pencarian dengan pendekatan teori graf. Evolusi, 16(1), 1-8.

<https://ejournal.bsi.ac.id/ejurnal/index.php/evolusi/article/view/702>

1. Lutfiana, E., Inayati, N., & Saraswati, G. W. (2020). Analisis perbandingan kinerja metode rekursif dan metode iteratif dalam algoritma linear search. Komputika, 17(2), 123-130.

<https://ojs.unikom.ac.id/index.php/komputika/article/view/5493>