MAKALAH PEMILIHAN ALGORITMA UNTUK MENINGKATKAN KEEFISIENAN PROGRAM

Febriawan Ghally Ar Rahman 13519111
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13519111@stdd.stei.itb.ac.id

Abstract- Program memiliki ciri unik dimana setiap dari program memiliki kompleksitas waktu eksekusi yang tidak selalu sama bergantung dari bagaimana algoritma program tersebut dibuat. Program dengan kompleksitas waktu yang besar memiliki waktu eksekusi yang lama sehingga sangat rawan menimbulkan error dalam pengeksekusiannya. Dalam makalah ini penulis akan membahas sejumlah algoritma yang digunakan untuk menyelesaikan persoalan yang sama dan mencoba menentukan yang terbaik diantaranya.

Keywords -Algoritma,Kompleksitas waktu

I. PENDAHULUAN

Algoritme adalah metode efektif diekspresikan sebagai rangkaian terbatas dari instruksi-instruksi yang telah didefinisikan dengan baik untuk menghitung sebuah fungsi. Dimulai dari sebuah kondisi awal dan input awal (mungkin kosong), instruksi-instruksi tersebut menjelaskan sebuah komputasi yang, bila dieksekusi, diproses lewat sejumlah urutan kondisi terbatas yang terdefinisi dengan baik, yang pada akhirnya menghasilkan "keluaran" dan berhenti di kondisi akhir. Transisi dari satu kondisi ke kondisi selanjutnya tidak harus deterministik; beberapa algoritme, dikenal dengan algoritme pengacakan, menggunakan masukan acak.

Kompleksitas algoritma adalah waktu tempuh yang dilakukan oleh computer untuk mengeksekusi program. Semakin besar kompleksitasnya maka semakin lama pula waktu eksekusinya.

Dalam menyelesaikan permasalahan ada banyak cara pendekatan yang dapat dillakukan untuk menyelesaikannya tentunya kita juga menginginkan program tesebut dieksekusi secara efisien agar menghidari error yang terjadi yang diakibatkan eksekusi program terlalu lama

Kelompok Algoritma	Nama
O(1)	Konstan
O(log n)	Logaritmik
O(n)	Linear
O(n log n)	n log n
$O(n^2)$	Kuadratik
$O(n^3)$	Kubik
$O(2^n)$	Eksponensial
O(n!)	Faktorial

Tabel notasi Big O

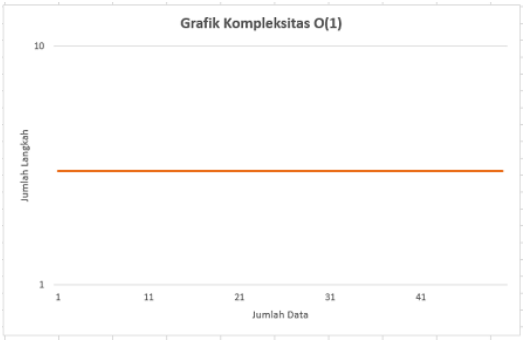
II. TEORI DASAR

Berdasarkan table notasi Big O atau notasi asimtotik terdapat beberapa notasi yang akan kita bahas.

1. O(1):Kompleksitas konstan

Adalah kompleksitas yang konstan dan tidak tumbuh berdasarkan ukuran dari masukan data atau masukan yang diterima oleh algoritma tersebut. Algoritma dengan kompleksitas konstan ini *memiliki jumlah langkah eksekusi yang sama untuk semua inputan*.

Contoh dari algoritma ini adalah implementasi rumus untuk menghitung luas dari sebuah persegi

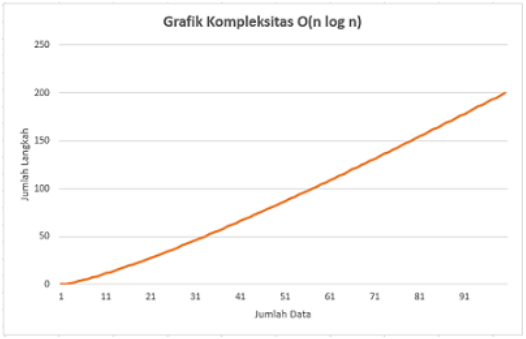


Grafik pertumbuhan O(1)

2. O(log n):Kompleksitas logaritmik

Algoritma dengan kompleksitas logaritmik merupakan algoritma yang menyelesaikan masalah dengan membagi-bagi masalah tersebut menjadi beberapa bagian, sehingga masalah dapat diselesaikan tanpa harus melakukan komputasi atau pengecekan terhadap seluruh masukan.

Contoh dari algoritma ini adalah pada penerapan *binary search*.

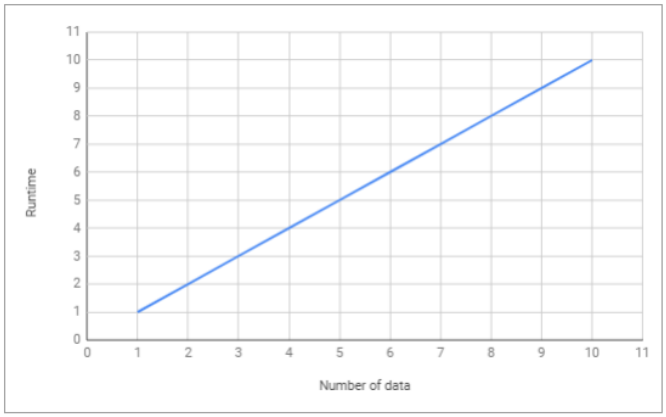


Grafik pertumbuhan O(log n)

3. O(n): Kompleksitas Linear

Algoritma ini bertumbuh selaras dan sebanding dengan meningkatnya ukuran data. Jika untuk suatu program memerlukan 10 langkah untuk input ukuran 10 dan 20 langkah untuk input ukuran 20, maka untuk input berukuran 100 memerlukan 100 langkah untuk mengeksekusinya

Contoh dari algoritma ini adalah algoritma dengan menggunakan iterasi sebanyak jumlah inputan

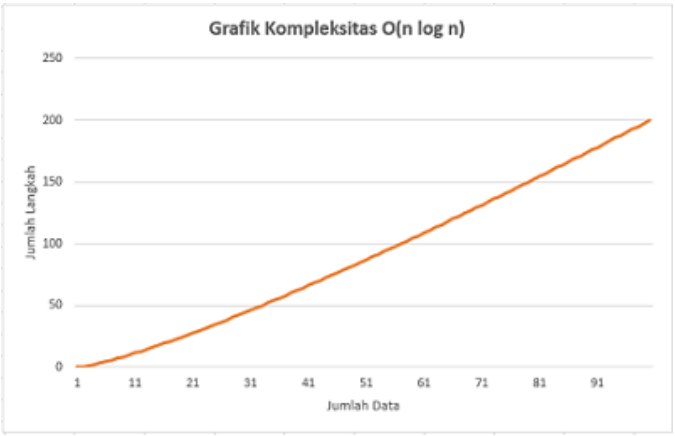


Grafik pertumbuhan O(n)

4. O(n log n)

Algoritma ini sebenarnya memiliki perhitungan log n yang sama seperti dengan sebelumnya, perbedaannya terletak pada algoritma log n diiterasi sebanyak n kali.

Contoh dari algoritma ini adalah iterasi binary search sebanyak n kali sesuai input.

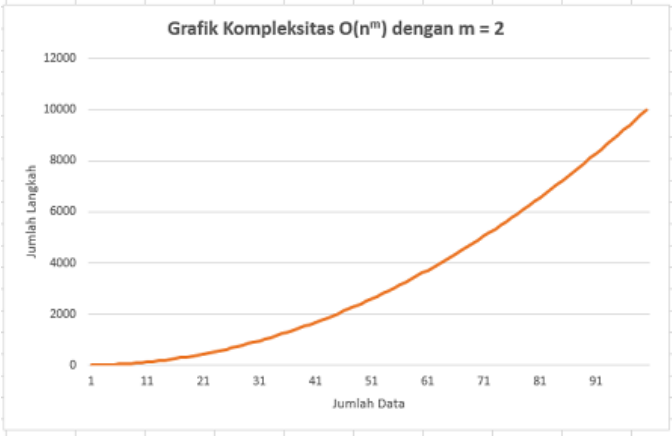


Grafik pertumbuhan O(n log n)

5. O(n^m):Kompleksitas Polinomial

Algoritma dengan kompleksitas polinomial merupakan salah satu kelas algoritma yang tidak efisien, karena memerlukan jumlah langkah penyelesaian yang jauh lebih besar daripada jumlah data.

Contoh dari algoritma ini adalah penggunaan nested loop



Grafik pertumbuhan $O(n^m)$

III. PEMBAHASAN

Pembahasan kali akan digunakan untuk membahas beberapa soal dari beberapa sumber yang saya temukan dan solusinya dengan menggunakan Bahasa c++.

1. Terdapat N tombol yang dinomori dari 1 hingga N dan sebuah lampu dalam keadaan mati. Apabila tombol ke-i ditekan, keadaan lampu akan berubah (dari mati menjadi menyala, atau sebaliknya) apabila N habis dibagi oleh i. Apabila masing-masing tombol ditekan tepat sekali, bagaimana keadaan lampu pada akhirnya?

Solusi sederhana adalah untuk mengiterasi setiap angka dari 1 sampai N dan mengubah keadaan lampu setiap kali N%i=0 kemudian mengecek keadaan diakhir dan menyatakan apakah lampu tersebut nyala/tidak, Kompleksitas algoritma ini adalah O(n).

```cpp
#include <iostream>

using namespace std;

int main() {
  long long N;
  cin >> N;

  int divisorCount = 0;
  for (long long i = 1; i <= N; i++) {
    if (N % i == 0) {
      divisorCount++;
    }
  }

  if (divisorCount % 2 == 0) {
    cout << "lampu mati" << endl;
  } else {
    cout << "lampu menyala" << endl;
  }
}
```


Implementasi dari solusi pertama

Solusi kedua adalah dengan melakukan faktorisasi dari n kemudian menghitung banyaknya factor dari n. kita hanya perlu mengecek bilangan sampai sqrt(n) sehingga kompleksitas algoritmanya adalah O(sqrt(n)).

```cpp
#include <iostream>

using namespace std;

int main() {
  long long N;
  cin >> N;

  long long num = N;
  int divisorCount = 1;
  for (long long i = 2; i*i <= num; i++) {
    int factorCount = 0;
    while (num % i == 0) {
      factorCount++;
      num /= i;
    }

    divisorCount *= (1 + factorCount);
  }
  if (num > 1) { // Sisa faktor
    divisorCount *= 2;
  }

  if (divisorCount % 2 == 0) {
    cout << "lampu mati" << endl;
  } else {
    cout << "lampu menyala" << endl;
  }
}
```

Implementasi solusi kedua

Solusi ketiga, kita mengetahui apabila tombol ditekan sebanyak n%2=1 maka lampu tersebut akan menyala, dengan memanfaatkan hal ini kita juga mengetahui bahwa banyaknya factor dari bilangan kuadrat sempurna adalah ganjil, maka kita hanya perlu mengecek bilangan tersebut ganjil atau tidak. Solusi ini memiliki Kompleksitas algoritma O(1)

```cpp
#include <iostream>
#include <cmath>

using namespace std;

int main() {
  long long N;
  cin >> N;

  long long s = round(sqrt(N));

  if (s * s != N) {
    cout << "lampu mati" << endl;
  } else {
    cout << "lampu menyala" << endl;
  }
}
```

Implementasi dari solusi ketiga

2. Contoh kedua ini akan membandingkan beberapa algoritma searching yang umum dipakai (*sequential search* dan *binary search*)

   a. Sequential search adalah metode searching dengan mencari secara linier keberadaan suatu elemen pada indeks tertentu Kompleksitas dari algoritma ini adalah O(n).

*Untitled - Notepad

File Edit Format View Help

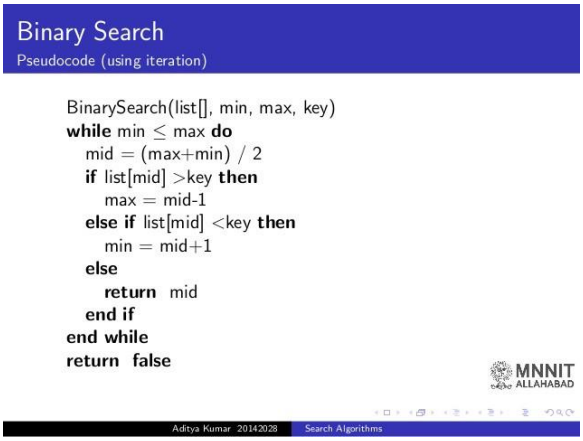
```
procedure search(a,X)

found <- 0

i <- 1
while(i < n and not(found)) do
        if(a[i]=X) then
                output(i)
                found <- 1
        i <- i+1
```

Pseudocode sequential search

   b. Binary search adalah algoritma searching yang dapat digunakan jika elemen-elemen pada pencarian terurut berdasarkan kriteria tertentu. Algoritma ini diimplementasikan dengan cara memotong ruang pencarian sebanyak setengah dari ruang pencarian sebelumnya sehingga akan mengemat cukup banyak waktu untuk ukuran input data yang besar.



Pseudocode dari Binary Search

3. Contoh ketiga ini akan membandingkan beberapa algoritma tentang pengurutan atau sorting.

   a. Bubble sort adalah algoritma sorting dengan menukar elemen yang bersebelahan satu persatu. Algoritma ini memiliki kompleksitas yang cukup buruk yaitu $O(n^2)$.

Pseudocode

```
procedure bubbleSort( A : list of sortable items ) defined as:
  do
    swapped := false
    for each i in 0 to length(A) - 2 inclusive do:
      if A[i] > A[i+1] then
        swap( A[i], A[i+1] )
        swapped := true
      end if
    end for
  while swapped
end procedure
```

   b. Selection sort adalah algoritma sorting dengan mencari nilai terkecil kemudian meletakkan pada posisi pertama, kemudian melanjutkan mencari nilai terkecil kedua dan meletakkannya pada posisi kedua dst.

Selection Sort – Pseudocode

Input: An array $A[1..n]$ of n elements.
Output: $A[1..n]$ sorted in descending order

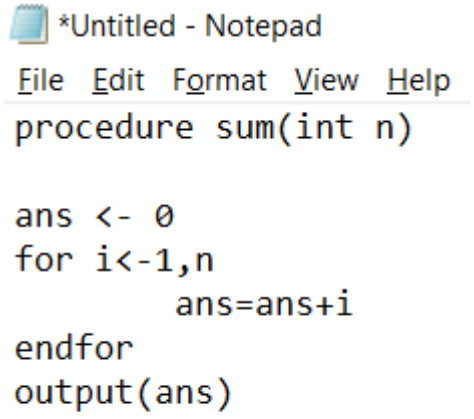
1. for $i \leftarrow 1$ to $n - 1$
2. min $\leftarrow i$
3. for $j \leftarrow i + 1$ to n {Find the i *th* smallest element.}
4. if $A[j] < A[min]$ then
5. min $\leftarrow j$
6. end for
7. if min $\neq i$ then interchange $A[i]$ and $A[min]$
8. end for

22

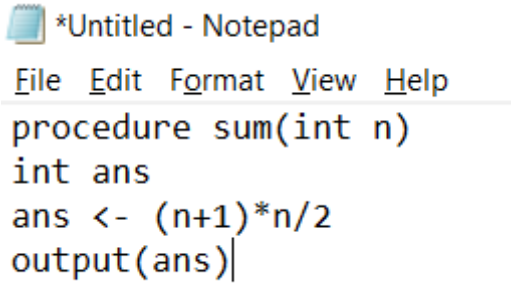
4. Pada contoh kali ini terdapat sebuah persoalan untuk menghitung bilangan dari 1 sampai n.

Terdapat 2 solusi yang umumnya akan dipakai yaitu:

a. Melakukan iterasi sebanyak n kemudian menjumlahkan masing-masing index iterasi. Algoritma ini memiliki kompleksitas waktu O(n)



b. Kita dapat mengimplementasikan rumus deret aritmatika secara langsung untuk memperoleh hasilnya. Algoritma ini memiliki kompleksitas waktu sebesar O(1).



IV. KESIMPULAN

Dengan memanfaatkan ilmu kompleksitas algoritma kita dapat membuat program dengan lebih efisien, meskipun masing-masing algoritma yang digunakan untuk menyelesaikan permasalahan memiliki tujuan sama tetapi kita dapat mengoptimalkan kerja dari program yang telah kita buat.

V. UCAPAN TERIMA KASIH

Pertama-tama penulis mengucapkan rasa syukur kepada Allah SWT dengan rahmatnya penulis bisa membuat dan menyelesaikan makalah ini. Kemudian ucapan terima kasih kepada keluarga karena telah memberi semangat. Serta kepada para dosen pengampu mata kuliah Matematika DiskritFariska Zakhralativa Ruskanda, S.T, M.T, Dr. Ir.Rinaldi Munir, M.T, Dra. Harlili S., M.Sc, dan Dr. Nur Ulfa Maulidevi S.T, M.Sc atas ilmu yang telah diberikan sehingga penulis dapat menyelesaikan makalah ini.

REFERENSI

[1] http://informatika.stei.itb.ac.id/~rinaldi.munir/. Diakses pada tanggal 11 Desember 2020.

[2] https://bertzzie.com/knowledge/analisis-algoritma/KompleksitasAlgoritma.html

Diakses pada tanggal 11 Desember 2020.

[3] Pemrograman Kompetitif Dasar Alham,William

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi

Sidoarjo,11 Desember 2020

Febriawan Ghally Ar Rahman 13519111