Penentuan Hubungan Kompleksitas Algoritma dengan Waktu Eksekusi pada Operasi Perkalian

Raymond Lukanta – 13510063[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13510063@std.stei.itb.ac.id

***Abstrak*—Algoritma adalah urutan langkah untuk menyelesaikan suatu persoalan. Kemangkusan suatu algoritma ditentukan oleh kompleksitas algoritma. Pada tulisan ini akan diulas hubungan antara kompleksitas algoritma dengan waktu eksekusi, khususnya hubungan perpangkatan. Algoritma yang menjadi uji coba adalah algoritma perkalian *divide and conquer* dengan algoritma Karatsuba.**

Setelah dilakukan analisis, ternyata diperoleh hasil bahwa kompleksitas tidak memiliki hubungan perpangkatan dengan waktu eksekusi. Selain itu juga diperoleh bahwa waktu eksekusi dapat digunakan untuk memprediksi waktu eksekusi apabila jumlah digit merupakan kelipatan dua.

***Kata kunci*—kompleksitas algoritma, divide and conquer, karatsuba, dan waktu eksekusi.**

I. Latar Belakang

Banyak respon yang dapat diekspresikan oleh orang yang berbeda ketika mendengar kata "kompleksitas". Bagi orang yang tidak berkecimpung di dunia informatika, kemungkinan besar belum pernah mendengar kata tersebut. Di lain pihak, kata "kompleksitas" sepatutnya menjadi makanan sehari-hari bagi orang yang bergelut di bidang Informatika. Namun, sayangnya banyak orang yang hanya mengetahui teori saja. Hal ini tentunya bukan merupakan rahasia lagi, karena praktik tentu saja tidak semudah teori. Lantas, bila praktik tidak semudah teori; apa jadinya bila teoripun tidak dapat dipahami. Hal tersebutlah yang melatarbelakangi pembuatan tulisan ini. Dengan adanya tulisan ini, diharapkan setiap pembaca menjadi lebih memahami makna dari kompleksitas.

Selain itu, tulisan ini juga dibuat sebagai laporan (*technical report*) atas eksperimen yang telah dilakukan. Eksperimen yang dilakukan adalah mencari hubungan antara kompleksitas dengan waktu eksekusi. Dengan mengetahui dampak kompleksitas terhadap waktu eksekusi, diharapkan pemahaman terhadap kompleksitas semakin menancap kuat. Karena penulis menilai bahwa waktu eksekusi lebih mudah dibayangkan daripada sekadar notasi matematika saja.

II. Tujuan

Tulisan ini dibuat untuk:

1. Meneliti apakah hubungan perpangkatan dapat diterapkan pada algoritma yang memiliki kompleksitas tertentu.
2. Mencari tahu hubungan apabila terdapat hubungan lain antara kompleksitas algoritma dengan waktu eksekusi.

III. Ruang Lingkup

Pada tulisan ini hanya dibahas keterhubungan antara kompleksitas dengan waktu eksekusi yang berbasiskan hubungan perpangkatan. Selain itu, algoritma yang diujicoba adalah algoritma perkalian *Divide and Conquer* dan algoritma Karatsuba.

IV. Kompleksitas Algoritma

Algoritma adalah urutan langkah-langkah yang harus dilakukan untuk memecahkan suatu persoalan. Dalam kehidupan sehari-hari, algoritma dikenal dengan istilah "prosedur" atau cara kerja. Namun, algoritma haruslah sistematis. Secara khusus, algoritma akan menentukan alur program yang dibuat. Apabila tidak sistematis, maka program yang dibuat tidak akan berjalan sesuai dengan yang diinginkan.

Selain sistematis, algoritma yang baik juga harus mangkus. Algoritma yang mangkus adalah yang hemat dan tepat guna dalam memakai sumber daya. Ada dua sumber daya utama, yaitu waktu dan ruang. Mangkus dari segi waktu berarti tidak banyak waktu yang terpakai untuk proses yang tidak perlu. Sedangkan mangkus dari segi ruang berarti penggunaan memori tidak boros. Kemangkusan algoritma merupakan hal yang penting agar semakin banyak proses yang dapat dilakukan oleh suatu komputer. Dengan kata lain, jumlah proses yang dieksekusi oleh komputer dapat menjadi lebih banyak tanpa harus melakukan penggantian perangkat keras.

Waktu eksekusi dapat diukur dengan menggunakan satuan waktu seperti detik, milidetik, mikrodetik, dan satuan waktu lainnya. Sedangkan ruang diukur dengan

menggunakan satuan *byte*, *kilobyte*, dan sebagainya. Saat ini, kompleksitas ruang tidak terlalu diperhatikan karena perkembangan perangkat keras (memori) sangat cepat. Sebagai dampaknya, kompleksitas waktulah yang menjadi fokus dalam penentuan kompleksitas.

Waktu eksekusi dapat diperoleh dengan menghitung setiap operasi yang terjadi. Operasi yang dimaksud dapat berupa operasi pengisian nilai (*assignment*), operasi aritmatika (tambah, kurang, kali, bagi, dan lain-lain), operasi logika (*and*, *or*, dan *not*), operasi perbandingan, dan operasi bit (*shifting*). Jika terdapat operasi yang dibungkus oleh kalang (*loop*), maka perhitungan operasi yang terjadi bergantung pada berapa kali proses pengulangan yang dilakukan.

Setiap algoritma memiliki komplesitasnya masing-masing. Algoritma yang berbeda, namun menghasilkan keluaran (*output*) yang sama, mungkin saja memiliki kompleksitas algoritma yang berbeda. Kompleksitas waktu biasa dilambangkan dengan dengan T(n), yang menunjukkan berapa waktu yang dibutuhkan untuk melaksanakan semua perintah yang ada dalam suatu kode program sampai selesai dengan ukuran masukan n. Selain T(n), kompleksitas juga dapat dilambangkan dengan notasi "O-Besar". Notasi O-Besar digunakan untuk menunjukkan kompleksitas waktu asimptotik. Makna dari notasi O-Besar adalah jika sebuah algoritma memiliki kompleksitas waktu asimptotik O(f(n)), maka bila n membesar menjadi sebesar apapun, waktu eksekusi tidak akan melebihi konstanta C dikali dengan f(n). Jadi, f(n) adalah batas atas dari T(n) untuk n yang besar. Dengan kata lain, T(n) memiliki orde paling besar f(n).

V. ALGORITMA DIVIDE AND CONQUER

A. Pengertian Divide and Conquer

Algoritma divide and conquer adalah metode untuk memecahkan masalah dengan cara membaga-bagi persoalan yang ada sehingga persoalan tersebut menjadi lebih kecil dan mudah diselesaikan. Secara etimologis, *divide* berarti memecah-mecah, sedangkan *conquer* memiliki arti menyelesaikan atau menaklukan. Algoritma ini terdiri dari tiga tahap, yaitu:

i. *Divide*: proses membagi-bagi masalah menjadi lebih kecil. Namun, masalah yang lebih kecil tersebut haruslah memiliki kemiripan dengan masalah awal.
ii. *Conquer*: menyelesaikan setiap masalah yang telah dipecah-pecah secara independen.
iii. *Combine*: proses penggabungan setiap hasil yang telah diperoleh dari proses conquer untuk mendapatkan solusi masalah awal secara keseluruhan.

Secara umum, algoritma *divide and conquer* lebih natural bila diimplementasikan dalam bentuk algoritma rekursif.

B. Algoritma Perkalian Divide and Conquer

Proses perkalian dengan *divide and conquer* dilakukan dengan membagi dua bilangan yang akan dikali. Berikut ini *pseudo-code* algoritma perkalian *divide and conquer* [2]:

```
function Kali(input X, Y :
LongInteger, n: integer) → LongInteger
{ Mengalikan X dan Y, masing-masing
panjangnya n digit dengan algoritma
Divide and Conquer.
  Masukan: X dan Y
  Keluaran: hasil perkalian X dan Y
}
Deklarasi
   a, b, c, d : LongInteger
   s : integer

Algoritma:
   if n = 1 then
      return X * Y
   else
      s←n div 2
      a←X div 10^s
      b←X mod 10^s
      c← Y div 10^s
      d← Y mod 10^s
      return Kali2(a, c, s)*10^2s +
Kali2(b, c, s)*10^s +
            Kali(a, d, s)*10^s +
Kali2(b, d, s)
  endif
```

Kompleksitas algoritma di atas adalah $T(n) = O(n^2)$.

C. Algoritma Karatsuba

Algoritma perkalian dengan *divide and conquer* pada upa bab sebelumnya diperbaiki oleh Anatolii Alexeevitch Karatsuba pada tahun 1962. Berikut adalah *pseudo-code* algoritma Karatsuba[2]:

```
  function Kali(input X, Y:
LongInteger, n: integer) → LongInteger

{ Mengalikan X dan Y, masing-masing
panjangnya n digit dengan algoritma
Divide and Conquer.
Masukan: X dan Y
Keluaran: hasil perkalian X dan Y
}
  Deklarasi
     a, b, c, d : LongInteger
     s : integer

  Algoritma:
     if n = 1 then
        return X * Y
     else
        s←n div 2
        a←X div 10^s
```

```
        b←X mod 10^s
        c← Y div 10^s
        d← Y mod 10^s
        p←Kali(a, c, s)
        q←Kali(b, d, s)
        r←Kali(a + b, c + d, s)
        return p*10^2s +(r −p −q)*10^s +q
    endif
```

Kompleksitas algoritma di atas adalah $O(n^{1,59})$

.

VI. METODE EKSPERIMEN

Eksperimen dilakukan dengan melakukan pengujian terhadap beberapa kasus pada kedua algoritma yang diuji, yaitu algoritma perkalian dengan *divide and conquer* dan algoritma Karatsuba. Pengujian dilakukan pada program yang dibuat dalam bahasa C++. Data uji yang dimasukkan adalah jumlah digit bilangan tertentu yang ingin dihitung waktu eksekusinya.

Sebelumnya, pada program sudah disisipi perintah untuk menghitung waktu eksekusi. Waktu eksekusi dihitung dalam satuan mikrosekon.

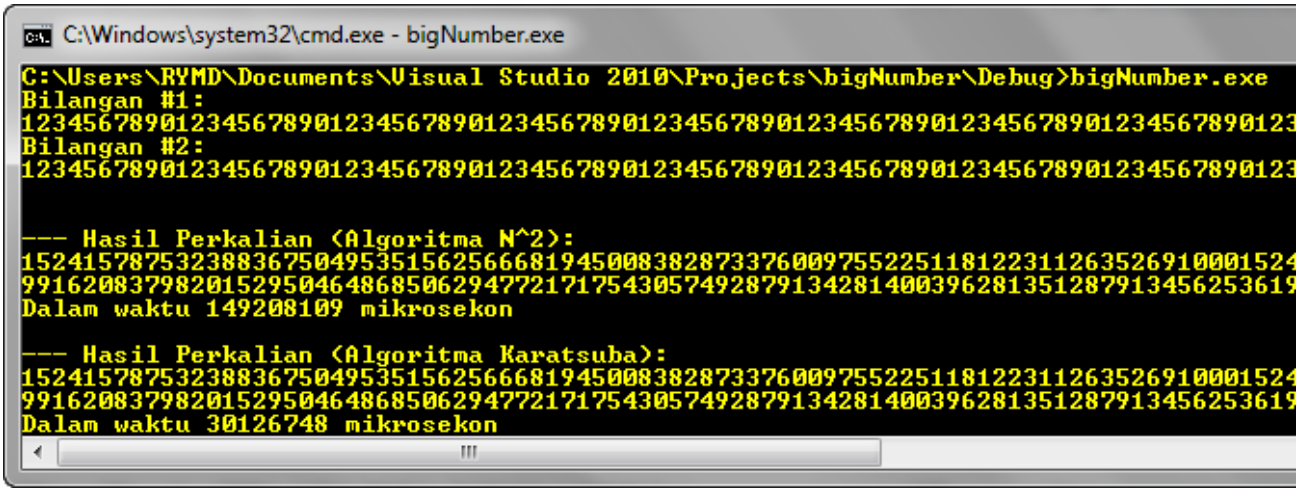
Setelah data uji dimasukkan ke program, waktu eksekusi yang merupakan keluaran dari program dicatat. Setelah data percobaan yang diperoleh dinilai cukup, selanjutnya dilakukan analisis terhadap data keluaran yang telah diperoleh. Analisis dilakukan dengan membandingkan waktu eksekusi yang diperoleh dari hasil percobaan dengan waktu eksekusi yang diharapkan. Waktu eksekusi yang diharapkan adalah sama dengan hasil yang diperoleh ketika suatu bilangan dihitung pada fungsi kompleksitas algoritma.

Tabel 1 Data Hasil Pengujian Hubungan Perpangkatan

	Pembanding		Hasil Uji		Pendekatan	Galat
	n^2	$n^{1,59}$	n^2	$n^{1,59}$		
1	1	1	553	505	2514	35461%
2	4	3.010493	2474	2250	16466	56555%
3	9	5.73618	7602	6770	65819	76581%
4	16	9.063071	11984	10818	118686	89037%
5	25	12.92297	28961	25500	348994	110505%
6	36	17.26873	33737	31503	455307	124958%
7	49	22.0651	39810	40380	622186	146289%
8	64	27.28432	44064	43727	687735	146076%
9	81	32.90376	117271	60193	1028034	77663%
10	100	38.90451	122511	79369	1455737	108825%
15	225	74.12848	165525	156043	3407083	195835%
20	400	117.1218	487368	226481	5443635	101695%
30	900	223.1633	639710	459194	13243653	197026%
40	1600	352.5944	1994418	731132	23773621	109201%
45	2025	425.2143	2126677	902096	30965919	135607%
50	2500	502.7618	2325133	1064810	38148228	154069%
60	3600	671.8317	2711578	1478939	57669173	202678%
80	6400	1061.483	8066565	2250222	97773370	111208%
100	10000	1513.561	9269292	3207932	152732342	154772%
150	22500	2883.933	31213554	6428560	366155054	107306%
200	40000	4556.566	37205495	9943884	633807385	160353%
300	90000	8682.06	124142161	19765532	1503984659	111150%
320	102400	9620.297	129777531	21438986	1665867879	118363%
400	160000	13717.51	149273291	30455764	2590721641	163556%

Waktu eksekusi dihitung dengan menggunakan perangkat keras berikut:

- Prosesor : Intel Core i7 1.7GHz Quad Core



Gambar 1 *Screenshot Program*

- RAM: 6 GB
- OS: Windows 7 Professional 64bit

VII. Data Hasil Pengujian

Data pembanding dan data hasil pengujian dapat dilihat pada tabel 1.Berikut penjelasan tabel 1:

- Kolom 1: digit angka yang dikalikan.
- Kolom 2: nilai keluaran dari fungsi kompleksitas algoritma perkalian divide and conquer.
- Kolom 3: nilai keluaran dari fungsi kompleksitas algoritma Karatsuba.
- Kolom 4: Waktu eksekusi (dalam µs) yang dihasilkan dari algoritma perkalian *divide and conquer*.
- Kolom 5: Waktu eksekusi (dalam µs) yang dihasilkan dari algoritma Kartsuba.
- Kolom 6: Hasil pendekatan $n^{1,59}$ terhadap n^2.
- Kolom 7: Perhitungan galat dari hasil pendekatan yang diperoleh.

VIII. Analisis

Analisis pertama dilakukan dengan melakukan pendekatan $n^{1,59}$ terhadap n^2. Pendekatan diperoleh dari perhitungan:

$$n^2 = n^{1,59\ X\ \frac{2}{1,59}}$$

Ternyata diperoleh bahwa hasil pendekatan tidak sesuai dengan hasil eksekusi algoritma dengan kompleksitas n^2. Untuk lebih jelas, perlu dilakukan perhitungan galat. Perhitungan tersebut diperoleh dari:

$$galat = \frac{p - a}{a} X100$$

Keterangan:
p: hasil pendekatan
a: waktu eksekusi algoritma dengan kompleksitas n^2.

Berdasarkan data yang diperoleh, ternyata galat cukup besar. Faktor yang dapat menyebabkan hal tersebut adalah ketidakstabilan kinerja komputer saat melakukan proses eksekusi. Selain itu, ada suatu faktor yang merupakan harga mati, yaitu bahwa kompleksitas tidak memiliki hubungan perpangkatan dengan waktu eksekusi.

Di sisi lain, bila dilakukan analisis jauh, ternyata data yang diperoleh dapat digunakan untuk melakukan prediksi waktu eksekusi untuk suatu nilai n tertentu. Hanya saja, hubungan ini berlaku hanya untuk n yang merupakan kelipatan dua dan hanya berlaku untuk algoritma perkalian *Divide and Conquer*. Berikut data hasil perhitungannya:

Tabel 2 Data Hasil Analisis

	n^2	Pendekatan	Galat (%)
2	2474	2212	10.59013743
4	11984	9896	17.42323097
6	33737	30408	9.86750452
8	44064	47936	-8.787218591
10	122511	115844	5.441960314
20	487368	490044	-0.549071749
30	639710	662100	-3.500023448
40	1994418	1949472	2.253589769
60	2711578	2558840	5.632808645
80	8066565	7977672	1.101993228
100	9269292	9300532	-0.337026819
200	37205495	37077168	0.344914105
300	124142161	124854216	-0.573580317
400	149273291	148821980	0.302338749

Pendekatan pada tabel di atas dilakukan dengan melakukan perhitungan

$$pendekatan = (\frac{n_1}{n_2})^2 X\ a$$

Keterangan:
$\frac{n_1}{n_2}$: perbandingan jumlah digit antara n1 dengan n2.
a: waktu eksekusi program dengan nilai $n=n_2$.

Dapat dilihat pada tabel 2 bahwa pendekatan yang diperoleh secara umum mendekati waktu eksekusi yang diperoleh. Agar lebih jelas, dapat dilihat dari galat yang diperoleh; galat yang diperoleh berkisar antara -8,79% sampai 17.423%.

IX. KESIMPULAN

Secara teoritis, kompleksitas algoritma berbanding lurus dengan jumlah data masukan yang diproses oleh algoritma tersebut. Namun percobaan yang dilakukan tidak berhasil membuktikan bahwa waktu eksekusi algoritma perkalian *divide and conquer* dan Karatsuba memiliki hubungan perpangkatan dengan kompleksitas algoritmanya. Ketidakberhasilan ini bisa saja diakibatkan karena ketidakstabilan CPU dalam menjalankan program. Karena itu diperlukan teknik pengukuran waktu yang tidak bergantung pada kestabilan CPU.

Untuk jumlah digit yang merupakan kelipatan dua, dapat dilakukan prediksi terhadap waktu eksekusi dengan digit yang merupakan kelipatan dua tersebut. Namun, hubungan ini hanya berlaku pada algoritma perkalian *divide and conquer*. Galat yang diperoleh berkisar antara -8,79% sampai 17.423%.

X. SARAN

Untuk menentukan hubungan perpangkatan antara kompleksitas dengan waktu eksekusi, perlu dilakukan pencarian teknik mengukur waktu yang tidak bergantung pada kinerja CPU. Selain itu, perlu juga pembuktian adanya hubungan perpangkatan pada algoritma yang lain.

REFERENSI

[1] Rinaldi Munir, Diktat Kuliah IF2091 Struktur Distrik, Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung.

[2] Rinaldi Munir, Diktat Kuliah IF3051 Strategi Algoritma, Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung.

PERNYATAAN

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

Bandung, 21 Desember 2012

Raymond Lukanta
13510063