Analisis Waktu Eksekusi Algoritma Perkalian Karatsuba dan Nikhilam

Felix
Universitas Sumatera Utara, Jl. Dr. T. Mansyur No. 9 Medan 20155
STMIK Mikroskil, Jl. Thamrin No. 112, 124, 140, Telp. (061) 4573767, Fax. (061) 4567789
Program Studi Magister Teknik Informatika, Universitas Sumatera Utara
Jurusan Teknik Informatika, STMIK Mikroskil, Medan
felixpandi@gmail.com

Abstrak

Sebagian perkalian dalam Ilmu Komputer melibatkan bilangan dengan jumlah digit yang banyak. Beberapa cabang ilmu yang banyak melibatkan perkalian bilangan dengan jumlah digit yang banyak adalah Kriptografi dan Kriptanalisis. Untuk melakukan perkalian tersebut, cara biasa yang dipelajari di tingkat SD sudah tidak memadai karena selain waktu komputasinya sangat lama, beberapa kalkulator pun tidak sanggup memberikan nilai yang akurat. Keakurasian tersebut merupakan hal pokok dalam Kriptografi dan Kriptanalisis. Oleh karena itu muncul algoritma perkalian Karatsuba pada tahun 1963 yang memiliki waktu komputasi yang lebih singkat. Algoritma ini selain menerapkan metode Divide and Conquer, juga menerapkan algoritma yang bersifat rekursif. Hal ini memang mampu menurunkan waktu eksekusi secara drastis, namun masih ada potensi untuk menurunkan waktu eksekusinya dengan menggabungkan algoritma Nikhilam ke dalam algoritma Karatsuba. Setelah penulis menggabungkan kedua algoritma tersebut, dihasilkan algoritma gabungan yang hanya membutuhkan kurang dari setengah kali waktu eksekusi algoritma Karatsuba Klasik.

Kata kunci*— waktu eksekusi, algoritma, perkalian, Karatsuba, Nikhilam*

Abstract

Some multiplication in Computer Science involved number with a lot of digits. Some branches which involved a lot of multiplication of number with a lot of digits are Cryptography and Cryptanalysis. To do the multiplication, the usual way which is learned in Primary is no longer enough because not only the computation time becomes very long, some calculators are also unable to give accurate result. Accuracy is a main thing in Cryptography and Cryptanalysis. Therefore Karatsuba multiplication algorithm was introduced in 1963 which has a shorter computation time. This algorithm implements Divide and Conquer method, and also implements recursive algorithm. This can reduce the execution time drastically, but there are still potential to reduce the execution time by combining Nikhilam algorithm into Karatsuba algorithm. After the writer combined both algorithms, a combined algorithm is produced which only needs less than half of the execution time of Classical Karatsuba algorithm.

Keywords*— execution time, algorithm, multiplication, Karatsuba, Nikhilam*

1. PENDAHULUAN

Operasi perkalian merupakan operasi yang penting dalam Ilmu Komputer (Eyupoglu, 2015) [1]. Hal ini disebabkan karena operasi perkalian banyak digunakan dalam berbagai bidang Ilmu Komputer. Koç (2015) memasukkan pertanyaan, "Apa algoritma tercepat untuk perkalian dari dua buah bilangan dengan n-digit?" ke dalam masalah yang belum terpecahkan dalam bidang Ilmu Komputer/Teknik Informatika (*Computer Science*) [2].

Beberapa riset utama mengenai algoritma perkalian bila disusun secara kronologis adalah yaitu *Karatsuba Algorithm* (1963); *Toom-Cook Multiplication* (1966); *Schönhage–Strassen Algorithm* (1971); *Nikhilam Algorithm* (1992); *Fürer Algorithm* (2007); *David Harvey, Joris van der Hoevena,*

and Grégoire Lecerf Algorithm (2014); dan *Svyatoslav Covanov and Emmanuel Thomé Algorithm* (2015). [3, 4, 5, 6, 7, 8, 9]

Adapun beberapa riset yang berkaitan dengan *Karatsuba Algorithm* yang berhasil diteliti oleh praktisi atau akademisi dari bidang ilmu *Computer Engineering, Computer Science, Mathematics and Computing* dan serumpunnya yaitu Tudor Jebelean (1997) menggunakan *Parallel Karatsuba Algorithm* untuk perkalian dan pembagian *Long Integer* [10]; Farij Ohmar Ehtiba dan Azman Samsudin (2004) menggunakan Montgomery dan Karatsuba untuk perkalian dan perpangkatan *Integer* [11]; Francis Tsang (2004) melakukan perbandingan antara algoritma perkalian standar, Karatsuba, dan Fourier untuk *Big Integer* [12]; Juliano B. Lima dan rekan-rekannya (2010) melakukan riset untuk menghitung perkalian polinomial dalam bentuk Chebyshev dengan menggunakan Algoritma Karatsuba [13]; Marco Bodrato dan Alberto Zanoni (2011) memanfaatkan metode Karatsuba dan Toom-Cook untuk menghitung *Multivariate Polynomial* [14]; dan Can Eyupoglu (2015) menganalisis performansi Algoritma Karatsuba dengan panjang bit yang berbeda-beda [1].

Sedangkan, hasil penelitian yang berkaitan dengan *Nikhilam Algorithm* yaitu Shri Prakash Dwivedi (2013) menjabarkan algoritma perkalian yang efisien dengan menggunakan *Nikhilam Method* [15] dan Can Eyopoglu (2015) menginvestigasi performansi dari Algoritma Nikhilam [16].

Setiap algoritma perkalian memiliki nilai batas yang membuatnya unggul ataupun lemah dibanding algoritma lainnya (Dwivedi, 2013) [15]. Algoritma Karatsuba akan unggul ketika nilai yang akan dikalikan memiliki jumlah digit yang besar (Karatsuba, 1992) [17]. Sedangkan Algoritma Nikhilam unggul ketika kedua bilangan yang akan dikalikan memiliki selisih yang kecil atau bahkan sama nilainya (Dwivedi, 2013) [15].

Algoritma Karatsuba memang lebih unggul dibandingkan algoritma Standar karena memiliki kompleksitas waktu yang lebih baik (Karatsuba, 1992) [17]. Namun Algoritma Karatsuba menerapkan Algoritma *Divide and Conquer* yang bersifat rekursif (Karatsuba, 1992) [17]. Dengan demikian, ketika bilangan yang hendak dikalikan menjadi besar, maka semakin sering terjadi pemanggilan fungsi secara rekursif. Hal ini memicu terjadinya pemborosan waktu dan memori sehingga waktu eksekusinya menjadi semakin lama.

Algoritma Nikhilam dapat mereduksi jumlah perkalian inti yang terjadi dengan menggunakan beberapa buah operasi pengurangan di dalamnya. Hal ini mampu menghemat waktu komputasi karena kompleksitas waktu operasi pengurangan jauh lebih rendah dibandingkan kompleksitas waktu perkalian standar (Dwivedi, 2013) [15]. Kompleksitas waktu pengurangan dengan notasi Big O adalah O(n) yang artinya peningkatan waktu dibandingkan dengan peningkatan input bersifat linear. Sedangkan untuk kompleksitas waktu perkalian Standar membutuhkan $O(n^2)$ yang berarti meningkatnya waktu dibandingkan dengan meningkatnya input adalah bersifat kuadratik (Eyupoglu, 2015) [1].

Dari hasil analisis tersebut, maka akan dilakukan analisis waktu eksekusi algoritma perkalian Karatsuba dan Nikhilam.

2. METODE PENELITIAN

Metode penelitian yang dilakukan adalah:

1. Menentukan jenis bilangan, panjang bilangan, dan bilangan yang akan dilakukan perkalian yang disebut juga sebagai data input.
2. Memasukkan data tersebut ke dalam algoritma perkalian Karatsuba dan algoritma perkalian gabungan Karatsuba dan Nikhilam dan diperiksa keakuratan hasil perkalian beserta durasi atau lamanya waktu ekseksusi tersebut.
3. Dari data output yang terkumpul disajikan dalam bentuk grafik garis.
4. Dari hasil pencatatan waktu eksekusi pada langkah 2 sebelumnya, dihitung rata-rata waktu eksekusi untuk masing-masing kategori panjang bilangan atau jumlah digitnya.
5. Hasil rata-rata tersebut dibuatkan lagi menjadi grafik garis yang berbeda.
6. Dari grafik garis yang dihasilkan dilakukan analisis dan penarikan kesimpulan.

Pada penelitian ini data yang diteliti adalah perkalian antara bilangan asli atau bilangan bulat positif saja. Bilangan yang dikalikan akan memiliki jumlah digit yang sama. Adapun jumlah digit atau panjang bilangan yang diteliti adalah 0, 2000, 4000, 6000, 8000, dan 10000. Dimana bilangan yang dikalikan adalah 111...111 x 111...111, 555...555 x 444...444, dan 999...999 x 999...999.

Penelitian dilakukan dengan menggunakan bahasa pemrograman Python 2.7 dan komputer dengan spesifikasi Compaq Presario CQ42 dengan prosesor Core i3.

Adapun beberapa algoritma yang digunakan dalam penelitian ini adalah algoritma perkalian Karatsuba, algoritma perkalian Nikhilam, dan algoritma perkalian gabungan Karatsuba dan Nikhilam yang diajukan oleh penulis.

2.1. Algoritma Perkalian Karatsuba

Algoritma Karatsuba merupakan algoritma pertama yang menerapkan konsep *Divide and Conquer*. Algoritma Karatsuba digunakan oleh Intel dan perusahaan lainnya untuk menghasilkan perkalian yang lebih cepat karena membutuhkan jumlah langkah yang lebih sedikit (Dwivedi, 2013) [15]. Selain itu, algoritma ini bisa diimplementasikan secara rekursif maupun iteratif (Eyupoglu, 2015) [1]. Adapun salah satu versi algoritma Karatsuba dapat dilihat pada Gambar 1 berikut ini.

```
 1: procedure karatsuba(num1, num2)
 2:   if (num1 < 10) or (num2 < 10)
 3:     return num1*num2
 4:   m = max(size_base10(num1), size_base10(num2))
 5:   m2 = m/2
 6:   high1, low1 = split_at(num1, m2)
 7:   high2, low2 = split_at(num2, m2)
 8:   z0 = karatsuba(low1,low2)
 9:   z1 = karatsuba((low1+high1),(low2+high2))
10:   z2 = karatsuba(high1,high2)
11:   return (z2*10^(2*m2))+((z1-z2-z0)*10^(m2))+(z0)
```

Gambar 1. Algoritma Perkalian Karatsuba

Dari Gambar 1 dapat dilihat pseudocode untuk Algoritma Perkalian Karatsuba. Baris pertama akan menerima input dua buah bilangan. Bilangan yang dapat diterima adalah bilangan bulat positif dengan tipe data *integer*, *long integer*, *big integer* dan sejenisnya. Bilangan negatif, pecahan, berkoma tidak diterima.

Selanjutnya pada baris kedua dan ketiga, kedua bilangan tersebut akan diperiksa apakah salah satunya sudah lebih kecil dari 10. Bila sudah, maka akan dilakukan perkalian antara bilangan pertama dan bilangan kedua dengan cara yang standar. Bila belum, maka bari keempat hingga baris kesepuluh akan dijalankan.

Baris keempat berfungsi untuk menentukan jumlah digit dari bilangan pertama dan kedua berdasarkan basis bilangan desimal. Jumlah digit terbesar yang diambil. Setelah itu dibagi dua dan disimpan di variabel m2 sesuai dengan baris kelima dari algoritma tersebut.

Baris keenam dan ketujuh memiliki kemiripan, bedanya adalah pada baris keenam, operasi pemisahan dilakukan pada bilangan pertama sedangkan pada baris ketujuh dilakukan pada bilangan kedua. Misalnya bilangan pertama adalah 9753 yang ditentukan secara acak, maka bilangan tersebut akan dipisahkan menjadi 97 dan 53. Nilai 97 disimpan oleh variabel high1 dan nilai 53 akan disimpan oleh variabel low1. Hal yang sama dilakukan juga pada bilangan kedua.

Pada baris kedelapan adalah z0 = karatsuba(low1, low2). Nilai low1 dan low2 yang diperoleh akan dimasukkan kembali pada procedure karatsuba(num1, num2) pada baris pertama secara rekursif. Hasilnya akan disimpan di variabel z0. Pseudocode baris kesembilan dan kesepuluh juga menerapkan cara yang sama dengan baris kedelapan. Bedanya adalah z1 = karatsuba((low1 + high1), (low2 + high2)) dan z2 = karatsuba(high1, high2). Perhitungan z1 dan z2 juga menerapkan cara rekursif dengan memanggil procedure baris pertama.

Setelah nilai z0, z1, dan z2 diperoleh, maka akan dijalankan baris kesebelas dan selesailah pseudocode tersebut.

2.2. Algoritma Perkalian Nikhilam

Algoritma Perkalian Nikhilam adalah salah satu algoritma yang dibahas pada jurnal ini. Terdapat dua jenis algoritma perkalian Nikhilam yang dinyatakan sebagai Nikhilam I dan Nikhilam II.

Perbedaan kedua algoritma tersebut adalah Nikhilam I digunakan apabila nilai yang akan dikalikan lebih kecil dari *Nearest base* dan mendekati nilai *Nearest base*, sedangkan Nikhilam II digunakan ketika nilai yang akan dikalikan sedikit lebih besar dari *Nearest base* dan mendekati nilai *Nearest base*. Nilai *Nearest base* adalah perpangkatan dari bilangan 10 yang mendekati nilai yang akan dikalikan.

Misalnya bila nilai yang dikalikan adalah 99 dan 98, maka lebih sesuai untuk menggunakan Nikhilam I karena mendekati *Nearest base* = 100 dan masih lebih kecil dari *Nearest base* = 100, sedangkan jika nilai yang dikalikan adalah 101 dan 102, maka lebih sesuai untuk menggunakan Nikhilam II karena mendekati *Nearest base* = 100 dan sedikit lebih besar dari *Nearest base* = 100.

Pada Gambar 2 berikut tertera algoritma perkalian Nikhilam I.

```
1: def nikhilam(a,b):
2:     A = nearestBase - a
3:     B = nearestBase - b
4:     C = A * B
5:     D = a - B = b - A
6:     result = nearestBase * D + C
7:     return result
```

Gambar 2. Algoritma Perkalian Nikhilam I

Pada algoritma perkalian Nikhilam I yang tertera di Gambar 2 terdapat 6 langkah utama yaitu:

a. Input dua buah bilangan yang mau dikalikan misalnya a dan b
b. Hitung nilai A dimana A = nearestBase – a. Nilai nearestBase diperoleh dari bilangan perpangkatan 10 yang terdekat dengan nilai a. Misalnya a = 97, maka nearestBase = 100. Bila a = 978, maka nearestBase = 1000.
c. Nilai B dihitung dengan cara yang sama dengan perhitungan nilai A.
d. Nilai A dan B dikalikan dengan algoritma standar dan disimpan di variabel C.
e. Nilai a – B akan menghasilkan nilai yang setara dengan b – A kemudian disimpan di variabel D.
f. Hasil akhir diperoleh dengan rumus Result = nearestBase * D – C.

```
1: def nikhilam(a,b):
2:     A = a - nearestBase
3:     B = b - nearestBase
4:     C = A * B
5:     D = a + B = b + A
6:     result = nearestBase * D + C
7:     return result
```

Gambar 3. Algoritma Perkalian Nikhilam II

Algoritma Perkalian Nikhilam II yang tertera di Gambar 3 memiliki kemiripan dengan algoritma perkalian Nikhilam I. Perbedaannya terletak pada langkah kedua dan ketiga dimana nilai A dan B dihitung dengan rumus A = a – Nearest base dan B = b – Nearest base. Perbedaan lain terletak pada langkah kelima dimana nilai D = a + B = b + A.

Pada jurnal ini, setiap penyebutan algoritma Nikhilam tanpa mencantumkan I atau II maka maksudnya adalah algoritma Nikhilam I.

2.3. Algoritma Perkalian Gabungan Karatsuba dan Nikhilam

Pada penelitian ini, penulis mengajukan algoritma perkalian gabungan. Algoritma yang digabungkan adalah Karatsuba dan Nikhilam I. Penulis memiliki hipotesis bahwa dengan algoritma gabungan ini dapat mengurangi waktu komputasi dibandingkan dengan algoritma Karatsuba klasik. Adapun algoritma perkalian gabungan Karatsuba dan Nikhilam dapat dilihat pada Gambar 4 berikut ini.

```
 1: procedure nikhilam (num1, num2)
 2:     A = 100 - num1
 3:     B = 100 - num2
 4:     C = A * B
 5:     D = num1 - B
 6:     return (100 * D + C)
 7:
 8: procedure karatsuba (num1, num2)
 9:     if (num1 < 100) or (num2 < 100)
10:         nikhilam (num1, num2)
11:     m = max (size_base10 (num1), size_base10 (num2))
12:     m2 = m / 2
13:     high1, low1 = split_at (num1, m2)
14:     high2, low2 = split_at (num2, m2)
15:     z0 = karatsuba (low1, low2)
16:     z1 = karatsuba ((low1 + high1), (low2 + high2))
17:     z2 = karatuba (high1, high2)
18:     return (z2 * 10 ^ (2 * m2)) + ((z1 - z2 - z0) * 10 ^ (m2)) + z0
```

Gambar 4. Algoritma Perkalian Gabungan Karatsuba dan Nikhilam

3. HASIL DAN PEMBAHASAN

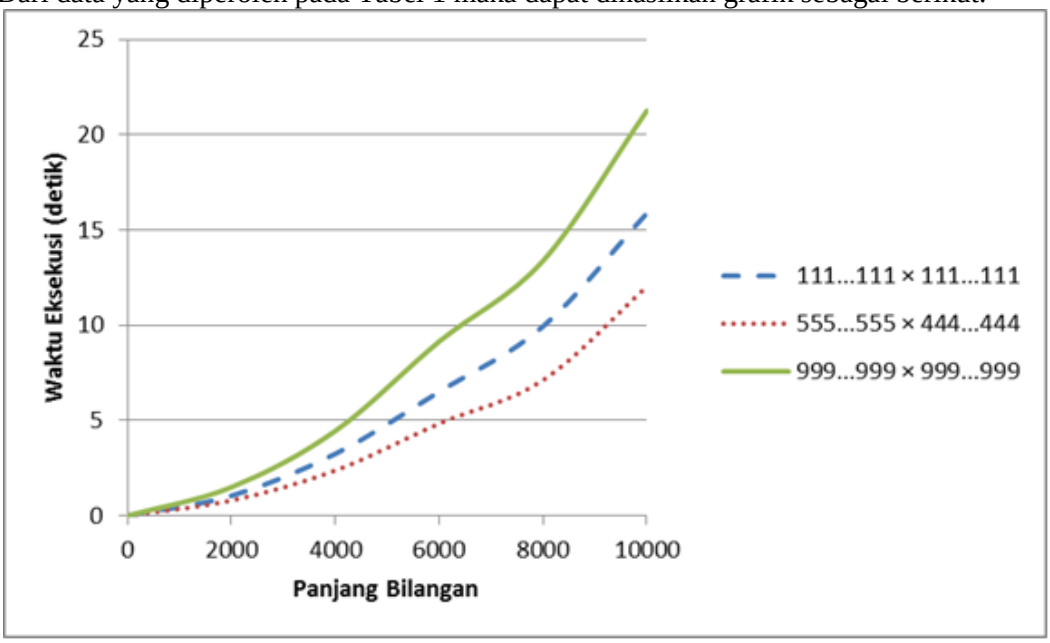
3.1. Hasil

Pada penelitian ini dilakukan pencatatan lamanya waktu eksekusi pada algoritma perkalian Karatsuba dan algoritma perkalian gabungan Karatsuba dan Nikhilam. Adapun waktu eksekusi dari algoritma perkalian Karatsuba dapat dilihat pada Tabel 1.

Tabel 1. Waktu Eksekusi Algoritma Karatsuba

Panjang Bilangan	Bilangan yang Dikalikan		Waktu Eksekusi (detik)
	Bilangan I	Bilangan II	
0	111...111	111...111	0.000
	555...555	444...444	0.000
	999...999	999...999	0.000
2000	111...111	111...111	1.045
	555...555	444...444	0.796
	999...999	999...999	1.497
4000	111...111	111...111	3.245
	555...555	444...444	2.371
	999...999	999...999	4.446
6000	111...111	111...111	6.489
	555...555	444...444	4.836
	999...999	999...999	9.141
8000	111...111	111...111	9.937
	555...555	444...444	7.114
	999...999	999...999	13.385
10000	111...111	111...111	15.881
	555...555	444...444	12.028
	999...999	999...999	21.263

Dari data yang diperoleh pada Tabel 1 maka dapat dihasilkan grafik sebagai berikut.



Gambar 5. Grafik Perbandingan Waktu Eksekusi Algoritma Karatsuba

Pada Gambar 5, panjang bilangan dinyatakan pada sumbu-x, dan waktu eksekusi dinyatakan pada sumbu-y dengan satuan detik. Pada bagian legenda di sebelah kanan menyatakan bilangan I yang dikalikan dengan bilangan II.

Panjang bilangan 2000 yang tertera pada sumbu-x menyatakan bahwa dilakukan perkalian antara bilangan I dengan bilangan II dimana bilangan I memiliki jumlah digit sebanyak 2000 buah dan bilangan II juga memiliki jumlah digit yang sama yaitu sebanyak 2000 buah.

Dari grafik pada Gambar 5 terlihat bahwa perbedaan bilangan yang digunakan dalam perkalian menghasilkan waktu eksekusi yang berbeda. Kecilnya bilangan yang dikalikan tidak memiliki korelasi kuat dengan waktu eksekusinya. Hal tersebut dapat terlihat ketika membandingkan perkalian 111...111 × 111...111 dengan 555...555 × 444...444. Pada tiap percobaan, terlihat bahwa perkalian 555...555 × 444...444 justru memiliki waktu eksekusi yang lebih cepat dibandingkan dengan 111...111 × 111...111. Meskipun demikian, perkalian 999...999 × 999...999 tetap memiliki waktu eksekusi yang lebih lama dibandingkan dengan 111...111 × 111...111.

Grafik pada Gambar 5 juga menunjukkan bahwa semakin panjang bilangan yang dikalikan, maka terdapat perbedaan waktu eksekusi yang semakin lama. Terdapat perbedaan yang lebih signifikan untuk perkalian yang panjang bilangannya 10000 digit dibandingkan dengan perkalian yang panjang bilangan 2000 digit. Terdapat selisih waktu eksekusi sebesar 9.235 detik antara perkalian bilangan 555...555 dengan 444...444 dan perkalian bilangan 999...999 dengan 999...999 untuk panjang bilangan 10000 digit. Namun, hanya 0.701 detik antara perkalian bilangan 555...555 dengan 444...444 dan perkalian bilangan 999...999 dengan 999...999 untuk panjang bilangan 2000 digit.

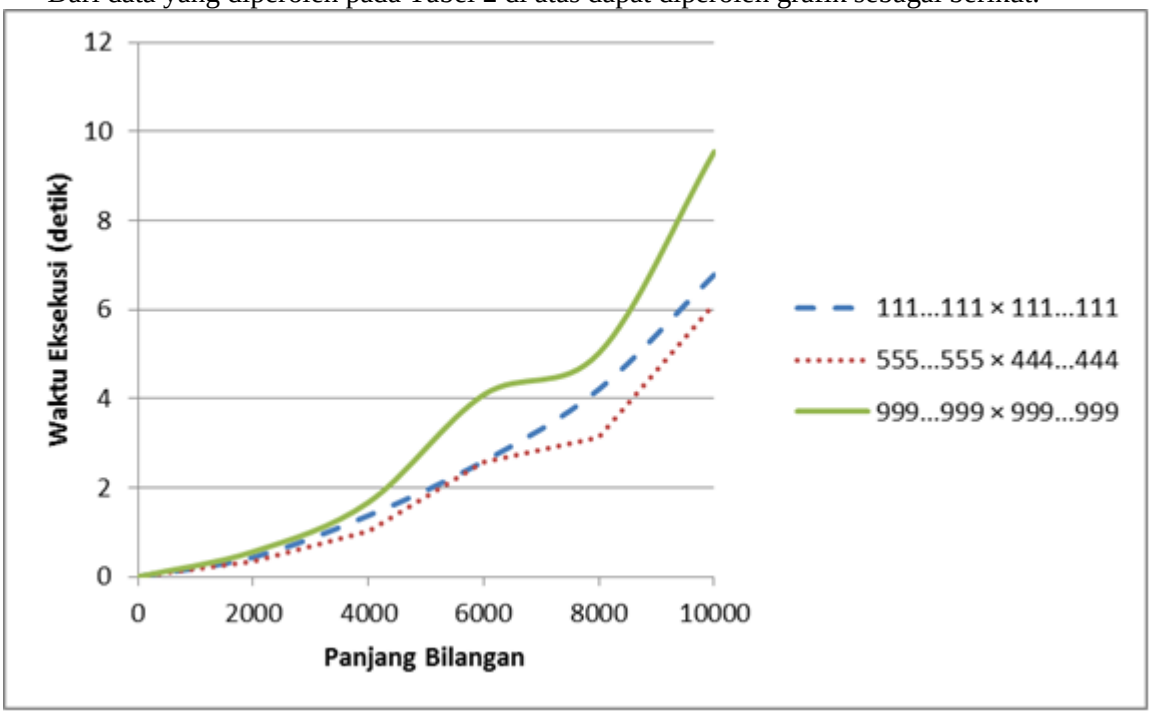
Waktu eksekusi untuk algoritma perkalian gabungan Karatsuba dan Nikhilam dapat dilihat pada Tabel 2 berikut ini.

Tabel 2. Waktu Eksekusi Algoritma Perkalian Gabungan Karatsuba dan Nikhlam

Panjang Bilangan	Bilangan yang Dikalikan		Waktu Eksekusi (detik)
	Bilangan I	Bilangan II	
0	111...111	111...111	0.000
	555...555	444...444	0.000
	999...999	999...999	0.000

2000	111...111	111...111	0.437
	555...555	444...444	0.343
	999...999	999...999	0.562
4000	111...111	111...111	1.373
	555...555	444...444	1.029
	999...999	999...999	1.669
6000	111...111	111...111	2.590
	555...555	444...444	2.558
	999...999	999...999	4.087
8000	111...111	111...111	4.212
	555...555	444...444	3.120
	999...999	999...999	5.024
10000	111...111	111...111	6.786
	555...555	444...444	6.115
	999...999	999...999	9.547

Dari data yang diperoleh pada Tabel 2 di atas dapat diperoleh grafik sebagai berikut.



Gambar 6. Grafik Perbandingan Waktu Eksekusi Algoritma Gabungan Karatsuba dan Nikhilam

Pada Gambar 6, panjang bilangan dinyatakan pada sumbu-x, dan waktu eksekusi dinyatakan pada sumbu-y dengan satuan detik. Pada bagian legenda di sebelah kanan menyatakan bilangan I yang dikalikan dengan bilangan II.

Panjang bilangan 2000 yang tertera pada sumbu-x menyatakan bahwa dilakukan perkalian antara bilangan I dengan bilangan II dimana bilangan I memiliki jumlah digit sebanyak 2000 buah dan bilangan II juga memiliki jumlah digit yang sama yaitu sebanyak 2000 buah.

Dari grafik pada Gambar 6 terlihat bahwa perbedaan bilangan yang digunakan dalam perkalian menghasilkan waktu eksekusi yang berbeda. Kecilnya bilangan yang dikalikan tidak memiliki korelasi kuat dengan waktu eksekusinya. Hal tersebut dapat terlihat ketika membandingkan perkalian 111...111 × 111...111 dengan 555...555 × 444...444. Pada tiap percobaan, terlihat bahwa perkalian 555...555 × 444...444 justru memiliki waktu eksekusi yang lebih cepat dibandingkan dengan 111...111 × 111...111.

Meskipun demikian, perkalian 999...999 × 999...999 tetap memiliki waktu eksekusi yang lebih lama dibandingkan dengan 111...111 × 111...111.

Grafik pada Gambar 6 juga menunjukkan bahwa semakin panjang bilangan yang dikalikan, maka terdapat perbedaan waktu eksekusi yang semakin lama. Terdapat perbedaan yang lebih signifikan untuk perkalian yang panjang bilangannya 10000 digit dibandingkan dengan perkalian yang panjang bilangan 2000 digit. Terdapat selisih waktu eksekusi sebesar 3.432 detik antara perkalian bilangan 555...555 dengan 444...444 dan perkalian bilangan 999...999 dengan 999...999 untuk panjang bilangan 10000 digit. Namun, hanya 0.219 detik antara perkalian bilangan 555...555 dengan 444...444 dan perkalian bilangan 999...999 dengan 999...999 untuk panjang bilangan 2000 digit.

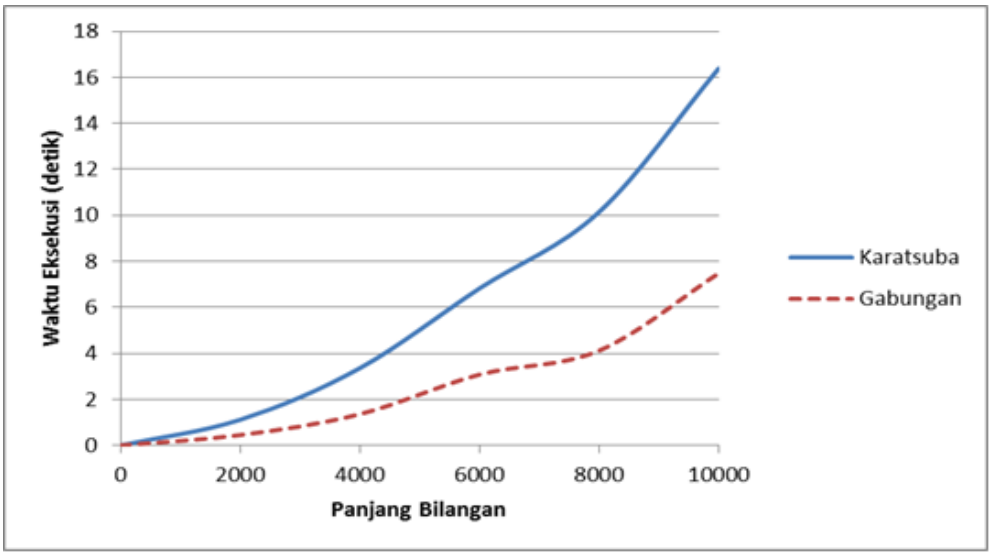
3.2. Pembahasan

Untuk membandingkan waktu eksekusi algoritma Karatsuba dan algoritma Gabungan maka waktu eksekusi untuk masing-masing panjang bilangan dirata-ratakan dan dicatat seperti pada Tabel 3 berikut ini. Adapun nilai yang dihitung dalam mencari rata-ratanya diambil dari data pada Tabel 1 dan Tabel 2.

Tabel 3. Perbandingan Waktu Eksekusi Algoritma Karatsuba dan Gabungan Karatsuba dan Nikhilam

Panjang Bilangan	Rata-rata Waktu Eksekusi (detik)	
	Karatsuba	Gabungan Karatsuba dan Nikhilam
0	0.000	0.000
2000	1.113	0.447
4000	3.354	1.357
6000	6.822	3.078
8000	10.145	4.119
10000	16.390	7.483

Pada Tabel 3 terlihat bahwa rata-rata waktu eksekusi algoritma Gabungan Karatsuba dan Nikhilam I jauh lebih baik dibandingkan dengan algoritma Karatsuba Klasik. Hal ini dapat dilihat dari waktu eksekusi algoritma Gabungan adalah lebih kecil dari setengah waktu eksekusi algoritma Karatsuba Klasik. Pada Gambar 7 dapat terlihat perbandingan waktu eksekusi antara kedua algoritma tersebut.



Gambar 7. Grafik Perbandingan Waktu Eksekusi Algoritma Karatsuba dan Gabungan Karatsuba dan Nikhilam

Pada Gambar 7 terlihat bahwa rata-rata waktu eksekusi algoritma Gabungan Karatsuba dan Nikhilam I jauh lebih baik dibandingkan dengan algoritma Karatsuba Klasik. Hal ini dapat dilihat dari waktu eksekusi algoritma Gabungan adalah lebih kecil dari setengah waktu eksekusi algoritma Karatsuba Klasik.

4. KESIMPULAN

Dalam penelitian ini, setelah peneliti mengkaji dan membandingkan dengan mengimplementasikan algoritma perkalian Karatsuba Klasik, algoritma perkalian Nikhilam I, dan algoritma perkalian Gabungan Karatsuba dan Nikhilam, didapatkan:

1. Algoritma perkalian Gabungan Karatsuba dan Nikhilam memiliki waktu eksekusi yang lebih singkat dibandingkan dengan algoritma perkalian Karatsuba Klasik.
2. Waktu eksekusi algoritma Gabungan Karatsuba dan Nikhilam mencapai kurang dari setengah kali waktu eksekusi algoritma Karatsuba Klasik.

5. SARAN

Penulis menyarankan untuk penelitian selanjutnya agar meneliti dan menggabungkan algoritma perkalian lainnya seperti algoritma Schönhage–Strassen, Fürer, Tiryakbhyam, dan lain-lain. Selain itu, penulis juga menyarankan supaya algoritma perkalian yang diteliti bisa mengalikan bilangan pecahan, desimal, dan negatif. Algoritma perkalian matriks juga bisa menjadi topik penelitian yang dibahas di kemudian hari.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Prof. Dr. Muhammad Zarlis, Prof. Dr. Herman Mawengkang, Dr. Erna Budhiarti Nababan, M.IT., dan Dr. Syahril Efendi, S.Si, M.IT yang telah memberikan saran dan bimbingan kepada penulis selama melakukan penelitian ini.

DAFTAR PUSTAKA

[1] Eyupoglu, C. 2015. Performance Analysis of Karatsuba Multiplication Algorithm for Different Bit Lengths. World Conference on Technology, Innovation and Entrepreneurship. Elsevier Procedia - Social and Behavioral Sciences Journal, Volume 195. pp 1860-1864.
[2] Koç, Ç. K. 2015. Open Problems in Mathematics and Computational Science. Springer.
[3] Karatsuba, A., & Ofman, Y. 1963. Multiplication of multidigit numbers on automata, English Translation in Soviet Physics Doklady.
[4] Stephen A. Cook, On the minimum computation time of functions, PhD thesis, Harvard University, Cambridge, MA, USA 1966, URL: http://cr.yp.to/bib/entries.html#1966/cook.
[5] Schönhage, A., & Strassen,V. 1971. Schnelle Multiplikation großer Zahlen. Computing, 7(3-4):281–292.
[6] Tirthaji, B. K. M. 1992. Vedic mathematics. Motilal Banarsidass Publication.
[7] Fürer, M. 2007. Faster integer multiplication. STOC '07 Proceedings of the thirty-ninth annual ACM symposium on Theory of computing. pp 57-66.
[8] Harvey, D., van der Hoeven, J., & Lecerf, G. 2014. Even faster integer multiplication.
[9] Covanov, S. & Thome, E. 2015. Fast arithmetic for faster integer multiplication. HAL CCSD.
[10] Jebelean, T. 1997. Using the Parallel Karatsuba Algorithm for long integer multiplication and division. Euro-Par'97 Parallel Processing. Springer.
[11] Ehtiba, F.O. & Samsudin, A. 2004. Multiplication and Exponentiation of Big Integers with Hybrid Montgomery and Distributed Karatsuba Algorithm. IEEE.
[12] Lima, B.J, Panario, D. & Wang, Q. 2013. A Karatsuba-based Algorithm for polynomial multiplication in Chebyshev Form. IEEE Transactions on Computers.
[13] Tsang, F. 2004. A comparison of Traditional, Karatsuba and Fourier big integer multiplication. Disertasi Ph.D. University of Bath.

[14] Bodrati, M. & Zanoni, A. 2011. Karatsuba and Toom-Cook methods for multivariate polynomials. Acta Universitatis Apulensis.
[15] Dwivedi, S. P. 2013. An efficient multiplication algorithm using Nikhilam method. Fifth International Conference on Advances in Recent Technologies in Communication and Computing (ARTCom), 20-21 Sept, Bangalore, IET, ISBN: 978-1-84919-842-4, 223-228.
[16] Eyupoglu, C. 2015. Investigation of the performance of Nikhilam Multiplication Algorithm. World Conference on Technology, Innovation and Entrepreneurship. Elsevier Procedia – Social and Behavioral Sciences Journal, Volume 195. pp 1959 – 1965.