Perbandingan Algoritma Brute Force dan Backtracking dalam Pencarian Triple Pythagoras

Albertus Djauhari Djohan/13515054
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13514016@std.stei.itb.ac.id

***Abstrak*—Ada berbagai teorema dalam bidang matematika yang banyak digunakan dalam berbagai cabang ilmu lain. Akan tetapi, banyak dari teorema tersebut yang penerapannya sulit jika tidak dikerjakan oleh algoritma yang mangkus. Sulit di sini berarti memakan waktu yang lama untuk jumlah data yang besar. Beberapa contoh dari persoalan ini adalah mencari bilangan prima ke-n untuk n yang cukup besar, mencari tiga pasang bilangan yang merupakan triple Pythagoras untuk batas bilangan yang cukup besar, dll. Persoalan-persoalan semacam ini jika dikerjakan dengan algoritma brute force tidaklah efisien dan akan memakan waktu yang lama. Salah satu dari beberapa persoalan semacam ini yang akan dibahas dalam makalah ini adalah pencarian tiga pasang bilangan yang merupakan triple Pythagoras.**

Kata Kunci—triple Pythagoras, brute force, backtracking, segitiga

I. PENDAHULUAN

Teorema Pythagoras merupakan salah satu teorema klasik dalam bidang geometri. Teorema ini telah banyak dikenal oleh masyarakat dan telah diajarkan sejak tingkat pendidikan dasar. Cara untuk membuktian teorema ini pun cukup beragam. Inti dari teorema ini adalah suatu segitiga dengan panjang sisi a, b, dan c adalah suatu segitiga siku-siku jika dan hanya jika jumlah dari kuadrat dua sisinya sama dengan kuadrat dari sisi yang lain.

Teorema ini banyak digunakan dalam cabang ilmu lain selain matematika, salah satu contohnya adalah fisika, yaitu dalam mekanika dan gerak. Teorema Pythagoras juga berperan dalam lahirnya aturan sinus, cosinus, dan tangen (aturan trigonomtri) yang memberikan sumbangan besar dalam bidang kalkulus.

Dari tak hingga banyaknya tiga pasang bilangan real positif yang memenuhi teorema Pythagoras, ada pasangan khusus yang semua anggotanya adalah bilangan bulat positif. Pasangan semacam ini disebut triple Pythagoras. Triple Pythagoras memiliki beberapa kegunaan, salah satunya adalah dalam bidang kriptografi.

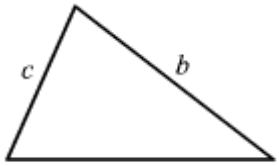
Ada beberapa metode untuk mencari bilangan-bilangan yang termasuk ke dalam triple Pythagoras. Salah satu metode yang paling mudah untuk diimplementasikan adalah metode brute force. Akan tetapi, penggunaan metode ini memakan waktu yang lama karena menelusuri semua kemungkinan yang ada, padahal ada kasus dimana penelusuran ke suatu cabang tidak perlu dilakukan. Salah satu perbaikan dari algoritma brute force adalah algoritma backtracking yang akan dibahas dalam makalah ini. Algoritma ini lebih mangkus dan waktu eksekusinya lebih singkat dibandingkan algoritma brute force.

II. DASAR TEORI

2.1 Segitiga

Segitiga adalah bangun datar paling sederhana dalam ilmu geometri. Segitiga merupakan poligon yang terdiri dari tiga titik sudut dan tiga sisi. Meskipun sederhana, konsep dan teori tentang segitiga memberikan banyak sumbangan dalam perkembangan matematika.

Salah satu teorema penting tentang segitiga adalah teorema ketaksamaan segitiga (*Triangle Inequality Theorem*) yang berbunyi hasil penjumlahan panjang dari dua sisi segitiga selalu lebih besar dari panjang sisi ketiga. Hal ini mudah dipahami karena jika penjumlahan panjang dua sisi segitiga lebih kecil atau sama dengan panjang sisi ketiga, maka tidak akan terbentuk segitiga, melainkan hanya terbentuk garis lurus. Secara matematis, teorema ini dapat dilihat pada gambar 1.



$$a + b > c$$

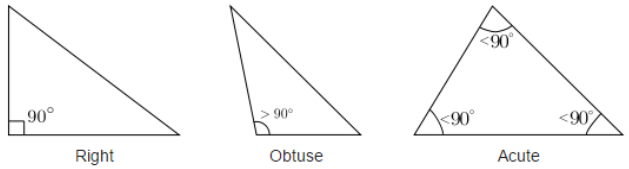
$$a + c > b$$

$$b + c > a$$

Gambar 1 : Teorema ketaksamaan segitiga
Sumber : https://www.varsitytutors.com/hotmath/hotmath_help/topics/triangle-inequality-theorem

Secara sederhana, segitiga dapat digolongkan berdasarkan dua propertinya, yaitu berdasarkan panjang sisi-sisinya dan berdasarkan besar sudut dalamnya.

Berdasarkan panjang sisinya, segitiga dapat digolongkan menjadi tiga macam, yaitu segitiga sama sisi (*equilateral triangle*), segitiga sama kaki (*isosceles triangle*), dan segitiga sembarang (*scalene triangle*). Segitiga sama sisi adalah segitiga yang panjang semua sisinya sama. Segitiga ini memiliki sifat khusus, yaitu besar semua sudut dalamnya sama, yaitu 60^0. Segitiga sama kaki memiliki dua sisi dengan panjang yang sama. Sifat khusus dari segitiga ini adalah dua sudut yang berhadapan dengan dua sisi yang sama panjang memiliki besar yang sama. Segitiga sembarang memiliki panjang sisi yang berbeda-beda. Hal ini juga menyebabkan semua sudut dalamnya memiliki besar yang berbeda.

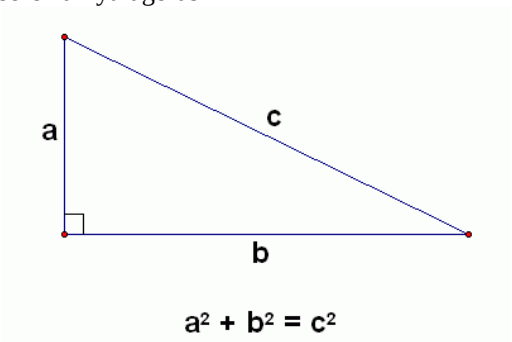


Gambar 2 : Penggolongan segitiga berdasarkan besar sudutnya

Sumber : Weisstein, Eric W. "Triangle". MathWorld.

Berdasarkan besar sudut dalamnya, segitiga dapat digolongkan menjadi tiga macam, yaitu segitiga lancip, segitiga siku-siku, dan segitiga tumpul. Segitiga lancip adalah segitiga yang semua sudutnya kurang dari 90^0. Segitiga ini memenuhi ketaksamaan $a^2+b^2 > c^2$. Segitiga siku-siku adalah segitiga yang salah satu sudutnya bernilai 90^0. Segitiga ini memenuhi persamaan $a^2 + b^2 = c^2$ (c adalah sisi terpanjang segitiga) , yang juga dikenal sebagai teorema Pythagoras dan akan dijelaskan pada bagian 2.2. Segitiga tumpul adalah segitiga yang salah satu sudutnya lebih dari 90^0. Segitiga ini memenuhi ketaksamaan $a^2 + b^2 < c^2$ (c adalah sisi terpanjang segitiga).

2.2 Teorema Pythagoras

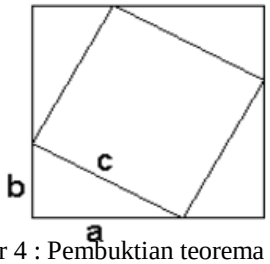


Gambar 3 : Segitiga yang memenuhi teorema Pythagoras

Sumber : http://ncalculators.com/number-conversion/pythagoras-theorem.htm

Teorema Pythagoras memiliki nama yang berasal dari seorang matematikawan Yunani yang menemukannya, yaitu Pythagoras (569 SM – 500 SM). Teorema ini menyatakan bahwa untuk semua segitiga siku-siku, jumlah kuadrat dari dua sisi segitiga yang membentuk sudut siku-siku sama dengan kuadrat dari sisi miringnya (hipotenusa). Hubungan ini secara matematis dapat dilihat pada gambar 3.

Ada sangat banyak cara untuk membuktikan kebenaran dari teorema Pythagoras. Salah satu cara yang mudah dapat dilihat pada gambar 2.



Gambar 4 : Pembuktian teorema Pythagoras

Sumber : http://www.cut-the-knot.org/pythagoras/

Pada gambar 4, terdapat sebuah persegi besar dengan panjang sisi a + b. Di dalam persegi tersebut diletakkan sebuah persegi yang berukuran lebih kecil dengan panjang sisi c seperti pada gambar 4. Ada du acara untuk menghitung luas persegi yang berukuran besar. Cara pertama adalah dengan menggunakan rumus luas persegi, yaitu sisi dikali sisi.

$$L_{\text{persegi besar}} = (a + b).(a + b)$$
$$= (a + b)^2 \ldots\ldots\ldots (1)$$

Cara kedua adalah dengan menjumlahkan semua bangun berukuran kecil yang terdapat di dalam persegi berukuran besar, yaitu sebuah persegi dengan panjang sisi c dan empat buah segitiga siku-siku dengan panjang sisi a, b, dan c.

$$L_{\text{persegi besar}} = L_{\text{persegi kecil}} + 4.(L_{\text{segitiga}})$$
$$= c.c + 4.a.b/2$$
$$= c^2 + 2ab \ldots\ldots\ldots (2)$$

Dengan menyamakan persamaan (1) dan (2) diperoleh bukti yang diperlukan.

$$(1) = (2)$$
$$(a + b)^2 = c^2 + 2ab$$
$$a^2 + b^2 + 2ab = c^2 + 2ab$$
$$a^2 + b^2 = c^2$$

(terbukti)

2.3 Algoritma Brute Force

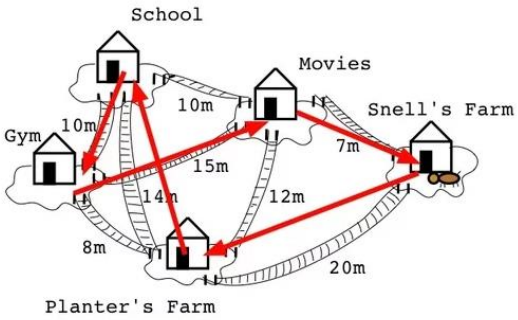
Algoritma brute force adalah metode pemecahan masalah yang memiliki pendekatan yang bersifat langsung (*straightforward*) dan berdasarkan pada pernyataan dan definisi pada persoalan yang akan dipecahkan. Karenanya, algoritma ini sederhana dan mudah dimengerti.

Algoritma brute force memiliki beberapa kelebihan. Metode ini dapat digunakan untuk memcahkan hampir sebagian besar masalah (*wide applicability*) karena pada dasarnya metode brute force mencoba semua kemungkinan yang mungkin untuk

menyelesaikan suatu permasalahan. Banyak algoritma penting yang didasarkan pada brute force, contohnya adalah pencarian elemen dalam larik dan senarai, pengurutan elemen dalam larik, pencocokan string, perkalian matriks, dan sebagainya.

Akan tetapi, algoritma brute force jarang dipakai untuk banyak persoalan karena waktu komputasinya yang lama. Salah satu contoh persoalan yang memakan waktu lama jika menggunakan algoritma brute force adalah TSP (*Travelling Salesman Problem*).

Bunyi dari persoalan ini adalah sebagai berikut. Seorang pedagang ingin melakukan perjalanan ke beberapa kota. Dia ingin mengunjungi semua kota tepat sekali dan kembali ke kota keberangkatan. Jika dia mengetahui jarak antar kota, tentukan rute dengan total jarak terpendek. Persoalan ini merupakan salah satu persoalan yang sangat terkenal dalam bidang *computer science* karena belum ditemukan algoritma untuk menyelesaikannya dengan kompleksitas waktu polinomial.



Gambar 5 : Persoalan TSP

Sumber : https://www.dogonews.com/2012/9/30/scientists-learn-how-bumblebees-solve-complex-traveling-salesman-problem-so-effortlessly

Dengan menggunakan algoritma brute force, persoalan ini dapat diselesaikan sebagai berikut.

a) Enumerasikan semua sirkuit Hamilton yang mungkin dari graf lengkap pada persoalan.

b) Hitung bobot total dari setiap sirkuit Hamilton yang diperoleh dari langkah a.

c) Pilih sirkuit Hamilton yang mempunyai bobot terkecil sebagai solusi.

Algoritma ini memiliki *cost* yang mahal. Jika terdapat n simpul pada graf lengkap, maka banyaknya cara untuk memilih simpul berikutnya dari simpul keberangkatan adalan n - 1 cara (terdapat n - 1 simpul pilihan karena 1 simpul merupakan simpul keberangkatan). Dari simpul kedua ini, terdapat n – 2 cara untuk memilih simpul berikutnya (karena dua simpul sudah terpilih, maka tidak boleh dipilih kembali), dan seterusnya hingga simpul terakhir. Dengan demikian, banyaknya rute yang ada adalah (n – 1)! rute.

Kompleksitas waktu untuk menghitung banyaknya kemungkinan sirkuit Hamilton adalah O(n – 1)! seperti yang telah dijelaskan pada paragraf sebelumnya. Kompleksitas waktu untuk menghitung bobot total dari suatu sirkuit Hamilton adalah O(n). Dengan demikian, kompleksitas total untuk algoritma ini adalah O(n.n!). Jika terdapat 20 simpul pada graf lengkap dan butuh waktu satu detik untuk menghitung satu sirkuit Hamilton, perlu waktu sekitar 1,6 trilliun tahun untuk menemukan solusinya.

2.4 Algoritma Backtracking

Backtracking (runut balik) adalah suatu algoritma untuk mencari sebagian atau semua solusi dari suatu persoalan yang memenuhi konstrain tertentu. Tidak seperti brute force, algoritma ini relatif lebih cerdas karena jika suatu pilihan tidak mungkin untuk mengarah ke solusi, maka pilihan tersebut tidak dieksplorasi lebih lanjut.

Pemangkasan simpul-simpul yang tidak mungkin menuju ke solusi membuat algoritma ini lebih cepat dibandingkan brute force karena tidak perlu menelusuri semua kemungkinan. Akan tetapi, pada kasus terburuk algoritma ini harus menelusuri semua kemungkinan sehingga waktu eksekusinya sama dengan algoritma brute force.

Algoritma backtracking memiliki tiga properti utama, yaitu sebagai berikut.

a) Solusi persoalan, dinyatakan sebagai vektor dengan *n-tuple*, yaitu X : ($x_1, x_2, x_3, \ldots, x_n$). Sebagai contoh, untuk persoalan TSP, x_i menyatakan simpul mana yang diambil. Untuk graf lengkap dengan lima simpul, salah satu contoh solusinya adalah X: (3, 2, 4, 5, 1). Artinya adalah rute yang dipilih dimulai dari simpul 1(simpul keberangkatan) ke simpul 3, 2, 4, 5, dan kembali lagi ke simpul 1.

b) Fungsi pembangkit nilai x_k, dinyatakan dengan predikat *T(k)*. *T(k)* membangkitkan nilai x_k yang merupakan komponen vektor solusi.

c) Fungsi pembatas, dinyatakan dengan predikat $B(x_1, x_2, x_3, \ldots, x_k)$. *B* bernilai *true* jika $(x_1, x_2, x_3, \ldots, x_k)$ mengarah ke solusi. Jika *true*, maka pembangkitan nilai untuk x_{k+1} dilanjutkan. Sebaliknya, jika *false* maka $(x_1, x_2, x_3, \ldots, x_k)$ dibuang. Fungsi inilah yang memangkas pilihan yang tidak mungkin mengarah ke solusi yang membuat algoritma backtracking lebih mangkus dibandingkan algoritma brute force.

Semua kemungkinan solusi dari persoalan disebut dengan ruang solusi (*solution space*). Ruang solusi diorganisasikan ke dalam struktur pohon. Tiap simpul pohon menyatakan status (*state*) persoalan, sedangkan sisi (cabang) dilabeli dengan nilai-nilai x_i. Lintasan dari akar ke daun menyatakan solusi yang mungkin. Seluruh lintasan dari akar ke daun membentuk ruang solusi. Pengorganisasian pohon ruang solusi diacu sebagai pohon ruang status (state space tree).

Dalam algoritma *backtracking*, solusi dicari dengan membentuk lintasan dari akar ke daun. Aturan pembentukan yang dipakai adalah mengikuti aturan *depht-first order* (DFS). Simpul-simpul yang sudah dilahirkan dinamakan simpul hidup

(*live node*), sedangkan Simpul hidup yang sedang diperluas dinamakan simpul-E (*Expand-node*). Tiap kali simpul-E diperluas, lintasan yang dibangun olehnya bertambah panjang. Jika lintasan yang sedang dibentuk tidak mengarah ke solusi, maka simpul-E tersebut "dibunuh" oleh fungsi pembatas sehingga menjadi simpul mati (*dead node*). Simpul mati ini tidak akan diperluan lagi. Pembentukan lintasan yang berakhir dengan simpul mati akan memicu *backtrack* ke simpul aras di atasnya. Pencarian dilanjutkan dengan membangkitkan simpul anak yang lain. Simpul ini akan menjadi simpul-E yang baru. Pencarian dihentikan jika sudah mencapai simpul tujuan.

III. IMPLEMENTASI

3.1 Implementasi Algoritma Brute Force

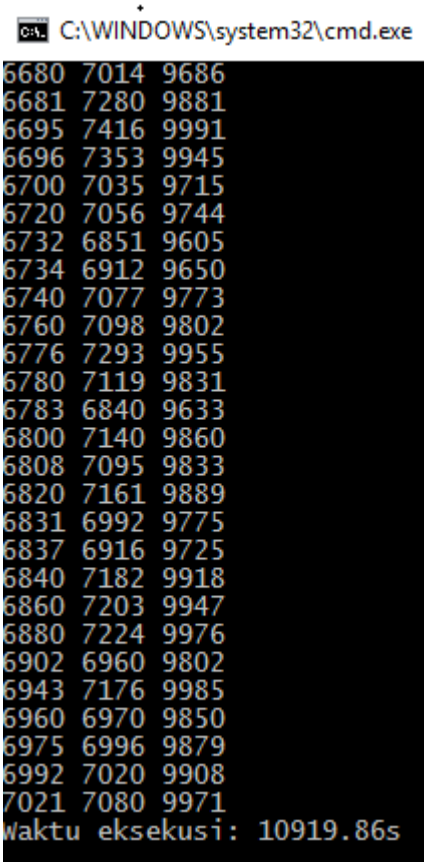
Implementasi algoritma brute force pada pencarian triple Pythagoras cukup mudah. Program melakukan loop dengan tiga tingkat untuk menelusuri semua kemungkinan tiga pasang angka yang memenuhi triple Pythagoras.

Berikut ini adalah *pseudocode* dari algoritma brute force. (untuk makalah ini, program ditulis dalam bahasa C++).

```
for(nilai a dari 1 sampai dengan n){
  for(nilai b dari 1 sampai dengan n){
    for(nilai c dari 1 sampai dengan n){
      Jika a, b, dan c memenuhi
      persamaan Pythagoras, maka cetak
      a, b, dan c ke layar
    }
  }
}
```

Program yang ditulis untuk keperluan eksperimen untuk makalah ini menerima input suatu bilangan bulat n, kemudian menampilkan semua bilangan antara 1 sampai n yang memenuhi triple Pythagoras dan menampilkan waktu eksekusinya.

Berikut ini adalah *screenshot* hasil eksekusi program untuk n = 10000. Bagian yang ditampilkan hanyalah sebagian kecil dari output program karena jika ditampilkan semuanya, ruang yang tersedia tidak cukup.



Gambar 6 : Hasil eksekusi program dengan algoritma brute force untuk input n = 10000

3.2 Implementasi Algoritma Backtracking

Implementasi algoritma *backtracking* memiliki kemiripan dengan implementasi algoritma brute force. Program juga melakukan loop dengan tiga tingkat. Perbedaannya, tidak semua kemungkinan ditelusuri. Berdasarkan dasar teori pada bagian 2.3 , berikut ini adalah properti utama dari algoritma yang digunakan pada program.

a) Solusi persoalan pada algoritma ini berupa vektor 3-tupple yang menyatakan 3 angka dalam triple Pythagoras. Salah satu contoh solusi adalah (3, 4, 5).

b) Fungsi pembangkit dalam algoritma ini adalah *loop for* yang melakukan inkremen terhadap nilai a, b, dan c yang menjadi angka yang akan membentuk triple Pythagoras.

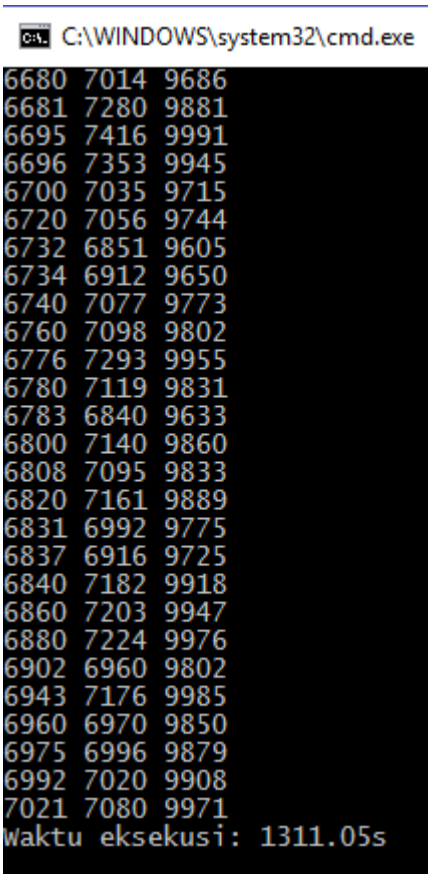
c) Ada beberapa fungsi pembatas dalam algoritma ini. Pertama, jika suatu triple Pythagoras (a, b, c) sudah ditemukan, pencarian tidak diteruskan dengan menginkremen nilai c, melainkan diteruskan dengan menginkremen nilai b. (loop yang pertama adalah loop terhadap a, kemudian di dalam loop a terdapat loop b, dan loop paling dalam adalah loop c). Hal ini karena untuk suatu

triple Pythagoras dengan nilai a dan b tertentu, hanya terdapat nilai c tunggal yang memenuhi persamaan Pythagoras. Kedua, pencarian solusi untuk suatu pilihan akan dihentikan jika sampai pada keadaan dimana a + b <= c belum ditemukan solusi. Dari sini akan dilakukan backtrack ke loop b untuk mencari solusi dengan nilai b yang lain. Hal ini didasarkan pada kenyataan bahwa nilai a, b, dan c yang membentuk triple Pythagoras merupakan nilai sisi – sisi segitiga. Sisi-sisi segitiga memenuhi ketaksamaan a + b > c. Hal ini tentu masuk akal karena jika a + b = c maka akan terbentuk garis lurus dan bukan segitiga. Demikian pula jika a + b < c, maka tidak akan terbentuk segitiga. Selain itu, pencarian untuk a dimulai dari 3, b dimulai dari a + 1, dan c dimulai dari b + 1. Alasannya, (3, 4, 5) adalah triple Pythagoras terkecil. Karena tidak ingin ada pengulangan triple yang sama, maka nilai b harus lebih dari a dan nilai c harus lebih dari b. Jika diijinkan bernilai sama, akan diperoleh beberapa triple Pythagoras yang sebenarnya sama. Contohnya (3,4,5), (3,5,4),(4,3,5),(4,5,3),(5,3,4),(5,4,3).

Berikut ini adalah *pseudocode* dari implementasi algoritma *backtracking* pada permasalahan ini. Sama seperti pada algoritma brute force, program untuk makalah ini juga ditulis dalam C++.

```
for(nilai a dari 3 sampai n){
  for(nilai b dari a+1 sampai n){
    for(nilai c dari b+1 sampai n){
      if(nilai a, b, c memenuhi persamaan Pythagoras){
        cetak nilai a, b, dan c
        lakukan backtrack
      }
      else if(a + b <= c){
        lakukan backtrack
      }
    }
  }
}
```

Berikut ini adalah *screenshot* hasil eksekusi program dengan algoritma *backtrack*. Hasil yang ditampilkan hanya sebagian kecil karena jika ditampilkan semuanya, ruang yang tersedia tidak cukup.



Gambar 7 : Hasil eksekusi program dengan algoritma *backtracking* untuk input n = 10000

IV. ANALISIS

Dari hasil eksperimen, diperoleh bahwa waktu eksekusi untuk program dengan algoritma brute force adalah 10919.86 detik. Jika dikonversi ke jam, maka waktu eksekusinya sekitar tiga jam. Sementara itu, hasil eksperimen untuk waktu eksekusi program dengan algoritma backtracking adalah 1311.05 detik. Jika dikonversi ke menit, maka waktu eksekusinya sekitar 22 menit. Perbedaan waktu eksekusi antara keduanya sangat besar. Perbandingan waktu eksekusi antara algoritma brute force dengan algoritma backtracking sekitar 8 banding 1.

Seperti telah dijelaskan pada bagian sebelumnya, perbedaan waktu yang besar dari kedua algoritma disebabkan karena adanya fungsi pembatas pada algoritma *backtracking*. Fungsi pembatas ini menyebabkan tidak semua kemungkinan ditelusuri. Sebagai contoh, pada kasus a = 3 dan b = 4. Pada algoritma brute force, terjadi n buah pemeriksaan, yaitu dari 1 sampai dengan n (dalam eksperimen ini, nilai n adalah 10000). Pada algoritma backtracking, jumlah pemeriksaan yang terjadi adalah satu kali. Pemeriksaan nilai c selalu dimulai dari b + 1, yang dalam hal ini nilai c adalah 5. Karena (3,4,5) adalah solusi, maka pencarian dihentikan dan dilanjutkan dengan b = 5. Ini adalah contoh kasus ekstrim dengan jumlah komputasi pada algoritma brute force adalah 10000 dan jumlah komputasi pada algoritma backtracking adalah 1.

Contoh kasus lain yang menunjukkan perbedaan signifikan antara kedua algoritma adalah saat a = 6 dan b = 7. Jumlah pencarian nilai c pada algoritma brute force adalah n kali, yaitu dari 1 sampai n. Jumlah pencarian pada algoritma backtracking adalah 6 kali, yaitu dari 8 sampai 13. Pencarian dihentikan saat mencapai c = 13 karena 6 + 7 = 13. Mulai dari sini, tuple (a, b ,c) tidak mungkin membentuk solusi lagi karena tidak mungkin membentuk segitiga dari nilai-nilai ini.

V. KESIMPULAN

Berdasarkan hasil eksperimen, dapat disimpulkan bahwa untuk persoalan pencarian triple Pythagoras, algoritma backtracking terbukti lebih mangkus dibandingkan dengan algoritma brute force. Hal ini dapat dilihat dari waktu eksekusi program dengan algoritma backtracking jauh lebih cepat dibandingkan dengan waktu eksekusi program dengan algoritma brute force.

Eksperimen ini tidak menjamin bahwa algoritma backtracking yang digunakan dalam program ini adalah algoritma yang paling mangkus untuk persoalan pencarian triple Pythagoras. Bisa jadi ada algoritma lain yang lebih mangkus, misalnya dengan memanfaatkan sifat dari triple Pythagoras bahwa kelipatan dari suatu triple Pythagoras juga merupakan triple Pythagoras. Sebagai contoh, (3,4,5) adalah triple Pythagoras. Demikian juga dengan (6,8,10) yang merupakan kelipatan dari (3,4,5).

ACKNOWLEDGMENT

Terima kasih kepada Tuhan Yang Maha Esa karena tanpa berkat dari-Nya penulis tidak dapat menyelesaikan makalah ini. Terima kasih juga kepada Ir Rinaldi Munir M.T. karena tanpa ilmu dari beliau, makalah ini tidak dapat dirampungkan. Terima kasih juga kepada semua pihak yang terkait dalam pembuatan makalah ini.

REFERENCES

Berikut ini adalah daftar pustaka yang digunakan dalam proses penyusunan makalah ini.

[1] https://www.ualr.edu/lasmoller/pythag.html

[2] http://faculty.simpson.edu/lydia.sinapova/www/cmsc250/LN250_Leviti n/L05-BruteForce.htm

[3] https://see.stanford.edu/materials/icspacs106b/H19-RecBacktrackExamples.pdf

[4] http://www.geeksforgeeks.org/category/algorithm/backtracking/

[5] https://www.varsitytutors.com/hotmath/hotmath_help/topics/triangle-inequality-theorem

[6] Weisstein, Eric W. "Triangle". MathWorld.

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, 29 April 2012

Albertus Djauhari Djohan/13515054