Aplikasi Algoritma Damerau–Levenshtein Distance dalam Mendeteksi Potensi Penipuan

Mohammad Ridwan Hady Arifin
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung, Indonesia
13517007@std.stei.itb.ac.id

***Abstrak*—Perkembangan internet yang sangat pesat mendorong munculnya banyak jenis bisnis baru. Salah satu bisnis yang sangat digandrungi adalah toko daring, karena kepraktisannya. Namun dalam praktiknya, terjadi banyak celah untuk melakukan tindak kejahatan penipuan. Dengan menggunakan Algoritma *Damerau-Levenshtein Minimum Edit Distance*, penulis mencoba menawarkan sebuah skema untuk mendeteksi adanya potensi penipuan. Potensi penipuan yang dimaksud, termasuk dalam jenis typosquatting. Keluaran dari algoritma yang penulis tawarkan adalah sebuah daftar nama – nama toko daring yang berpotensi melakukan penipuan. Dalam hal penindakan dan verifikasi benar tidaknya, akan dilakukan oleh tenaga manusia sendiri. Dalam makalah ini juga ditunjukan dengan beberapa contoh pseudocode agar mudah dipahami oleh pembaca.**

Kata Kunci—string; Damerau-Levenshtein; typosquatting; minimum; penyuntingan; algoritma.

I. PENDAHULUAN

Dalam beberapa tahun terakhir, internet berkembang dengan sangat pesat. Sebagai salah satu efek dari berkembangnya internet adalah munculnya model - model bisnis baru yang bergantung pada internet. Di antara bisnis – bisnis baru yang muncul, salah satu jenis bisnis yang paling populer dan digandrungi baik dari mata pebisnis maupun pengguna adalah toko daring. Beberapa toko daring yang berkembang di Indonesia seperti halnya Bukalapak, BliBli, Tokopedia, Shopee, JD.id, OLX, Hijup hingga Zalora.

***Gambar 1.** Beberapa toko daring di Indonesia*

Dengan banyaknya media yang membantu penjual membuka toko secara virtual, jumlah penjual/toko pada setiap media memiliki jumlah yang luar biasa. Tentu untuk melakukan verifikasi keaslian toko dan penjual, diperlukan sumber daya yang amat sangat besar. Hal ini, memunculkan banyak toko toko palsu dengan nama - nama yang dimiripkan toko sebenarnya. Misalnya ada 2 toko, Samsung store (asli) dan Samsunk store (palsu). Untuk orang – orang yang mungkin lalai dalam mengecek kembali nama toko dari barang yang akan dibelinya, bisa saja tertipu.

Modus penipuan dengan membuat nama yang seolah olah mirip, atau sengaja dimunculkan saat terjadi typo, disebut dengan typosquatting. Hal ini sangat sering ditemui dalam proses transaksi di Indonesia. Tidak hanya pada toko daring, modus penipuan juga terjadi melalui pengiriman pesan singkat dengan mengatasnamakan suatu instansi tertentu, namun dengan nama yang sengaja ditulis sedikit salah.

Dari hal tersebut, penulis mencoba membuat sebuah aplikasi untuk mendeteksi adanya potensi penipuan, dengan model memiripkan nama toko. Hasil dari pendeteksian potensi penipuan, tidak akan langsung diberi tindakan, namun hanya akan dimasukan ke dalam sebuah laporan, untuk kemudian diverifikasi secara langsung oleh sumber daya manusia. Hal ini dilakukan untuk menjamin target potensi penipuan tidak salah sasaran.

II. DASAR TEORI

A. *Pemrograman Dinamis*

Pemrograman dinamis ditemukan oleh seorang matematikawan berkebangsaan Amerika bernama Richard Bellman pada tahun 1950an yang pada awalnya digunakan untuk mengoptimasi proses pengambilan keputusan yang berjenjang. Kata pemrograman awalnya merujuk pada perencanaan, ketimbang pada pemrograman yang dikenal dalam dunia ilmu komputer. Kemudian, karena dirasa cocok untuk diterapkan dalam dunia komputer, kemudian berkembanglah algoritma algoritma yang berbasis pada konsep pemrograman dinamis.

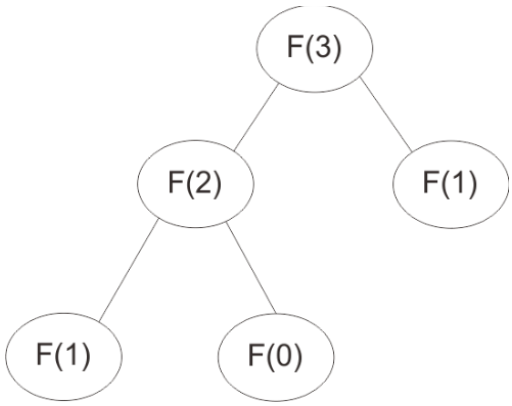
Pemrograman dinamis adalah sebuah teknik untuk menyelesaikan masalah yang memiliki upamasalah yang saling tumpeng tindih satu sama lain. Daripada harus menyelesaikan masalah yang saling tumpeng tindih berkali kali, pemrograman

dinamis menawarkan solusi untuk menyimpan hasil perhitungan pada sebuah tabel, sehingga ketika dibutuhkan kembali, tidak perlu adanya komputasi yang berulang.

Kunci dari pemrograman dinamis terletak pada sub-masalah yang saling tumpeng tindih satu sama lain, dan sistem tabel yang dibuat untuk memangkas jumlah komputasinya. Contoh paling mudah untuk hal ini adalah algoritma untuk menghitung bilangan fibonacci. Dalam Algoritma rekurens normal, misalnya untuk menghitung fibonacci ketiga, diperlukan sebanyak 4 komputasi, sedangkan dengan menggunakan pemrograman dinamis, cukup dilakukan dengan 3 komputasi.

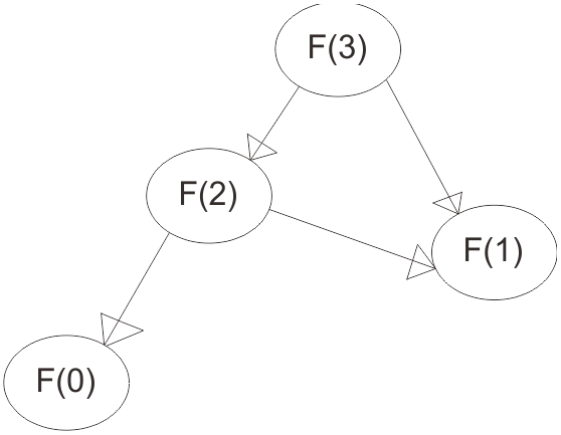
$$F(n) = F(n-1) + F(n-2)\,, n > 1$$

Dengan F(0) = 1 dan F(1) = 1, pada algoritma rekurens akan dihasilkan pemanggilan sebagai berikut :



***Gambar 2**. Fibonacci biasa*

Sedangkan dengan menggunakan program dinamis, hanya akan dihasilkan pemanggilan berikut:



***Gambar 3**. Fibonacci dengan metode program dinamis*

Dalam mengisi tabel, terdapat 2 metode utama yang bisa digunakan, yaitu *Memoization (top-down cache filling)* dan *Tabulation (bottom-up cache filling).*

Untuk *memorizatio*n, algoritma penghitungan Fibonacci akan seperti ini

```
memFib(n) {
       if (mem[n] is undefined)
         if (n < 2) result = n
         else result = memFib(n-2) + memFib(n-1)
         mem[n] = result
       return mem[n]
}
```

Sedangkan untuk *tabulation*, akan di dapat kode seperti berikut:

```
tabFib(n) {
       mem[0] = 0
       mem[1] = 1
       for i = 2...n
           mem[i] = mem[i-2] + mem[i-1]
       return mem[n]
}
```

B. Damerau-Levenshtein Distance

Damerau-Levensthein Distance, dalam ilmu komputer, adalah sebuah algoritma yang dikembangkan oleh Frederick J. Damerau dan Vladimir Levenshtein sebagai sebuah bentuk improvisasi dari algoritma *Levenshtein* distance. Baik *Damerau-Levensthein* maupun Levensthein, adalah sebuah algoritma yang digunakan untuk mencari operasi minimal untuk melakukan penyuntingan dari 2 *string*, agar menjadi sama. Dalam algoritma awalnya (Levenshtein), terdapat 3 operasi yaitu *insertions* (insersi), *deletions* (penghapusan), dan *subtitutions* (subtitusi). Sedangkan dalam versi penyempurnaan yang diusulkan oleh *Damerau*, beliau menambahkan operasi transposisi dari dua karakter yang bersebelahan.

Dalam sebuah paper yang Damerau tulis, dengan judul *"A technique for computer detection and correction of spelling errors"*, beliau berpendapat bahwa lebih dari 80% kesalahan penulisan disebabkan oleh keempat tipe diatas.

Dari hal tersebut, Algoritma *Damerau-Levenshtein* Distance dapat dirumuskan sebagai berikut:

$$DL_{(a,b)}(i,j) = min \begin{cases} 0 & jika\ i = j = 0 \\ DL_{(a,b)}(i-1,j) + 1 & jika\ i > 0 \\ DL_{(a,b)}(i,j-1) + 1 & jika\ j > 0 \\ DL_{(a,b)}(i-1,j-1) + 1_{(ai \neq bj)} & jika\ i,j > 0 \\ DL_{(a,b)}(i-2,j-2) + 1 & jika\ i,j > 1\ dan \\ & a[i] = b[j-1]\ dan\ a[i-1] = b[j] \end{cases}$$

Dengan a dan b merupakan 2 buah *string* yang akan dibandingkan.

Masing masing kasus diatas berkorespondensi sebagai berikut:

$DL_{(a,b)}(i-1,j) + 1$, *deletions*
$DL_{(a,b)}(i,j-1) + 1$, *insertions*
$DL_{(a,b)}(i-1,j-1) + 1_{(ai \neq bj)}$, *subtitutions*
$DL_{(a,b)}(i-2,j-2) + 1$, *tranpositions*

Dari rumus yang telah diturunkan diatas, berikut adalah pseudocode dari Algoritma *Damerau-Levenshtein*:

```
DL-distance(string a, string b)
  da := new array of |Σ| integers
  for i := 1 to |Σ| inclusive do
    da[i] := 0

  let d[0..length(a), 0..length(b)] be a 2-d
array of integers, dimensions length(a)+1,
length(b)+1

  for i := 0 to length(a) inclusive do
    d[i, 0] := i
  for j := 0 to length(b) inclusive do
    d[0, j] := j

  for i := 1 to length(a) inclusive do
    for j := 1 to length(b) inclusive do
      if a[i] = b[j] then
         cost := 0
      else
          cost := 1
      d[i, j] := minimum(d[i-1, j] + 1,     //
deletion
                d[i, j-1] + 1,     // insertion
                d[i-1, j-1] + cost)  //
substitution
      if i > 1 and j > 1 and a[i] = b[j-1] and
a[i-1] = b[j] then
      d[i, j] := minimum(d[i, j],
                d[i-2, j-2] + cost)  //
transposition
    return d[length(a), length(b)]
```

Dalam mempermudah pencarian hasil *Damerau-Levenshtein minimum edit distance*, kita bisa menggunakan sebuah metrik yang merepresentasikan jumlah edit yang diperlukan. Misalnya kita memiliki dua buah *string* dengan panjang M dan N. Untuk membuat metrik yang akan membantu mencari operasi penyuntingan minimum, kita perlu melakukan terlebih dahulu inisialisasi matriks dengan ukuran [0..M][0..N]. Indeks 0 baik pada kolom maupun baris akan berisi nilai *NULL*. Lalu inisialisasi setiap baris 0 dan kolom 0 dengan indeks yang bersesuaian. Berikut adalah contoh tabel inisialisasi dengan 2 buah *string* yaitu "AC" dan "CBA":

***Tabel 1**. Tabel inisialisasi penyuntingan*

3-A	3		
2-B	2		
1-C	1		
0-null	0	1	2
	0-null	1-A	2-C

Setelah inisialisasi dilakukan, selanjutnya, isi sisa tabel yang masih kosong sesuai dengan rumus yang telah dijabarkan sebelumnya. Dengan mengikuti kaidah rumus diatas, maka akan didapat tabel berikut.

***Tabel 2**. Tabel hasil akhir penyuntingan*

3-A	3	2	**3**
2-B	2	2	2
1-C	1	1	1
0-null	0	1	2
	0-null	1-A	2-C

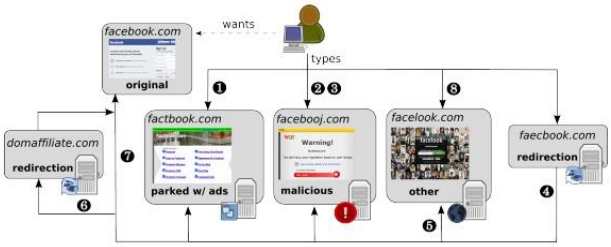
Operasi minimum, sebagai output, ditunjukan pada indeks ke M,N yang dalam hal ini bernilai 3. Hal tersebut dapat dijelaskan sebagai berikut:

AC -> C -> CB -> CBA

Awalnya, string AC akan mengalami pertukaran dan menjadi CA, kemudian dilakukan insersi, sehingga menjadi CBA, jumlah operasi minimum yang dibutuhkan adalah 3.

C. Typosquatting

Typosquatting adalah sebuah prilaku spekulatif dengan memanfaatkan penamaan pada internet, untuk mendapatkan keuntungan dari kesalahan penulisan atau ejaan dari pengguna. Praktik typosquatting banyak ditemui di dalam penulisan domain. Para pelakunya, akan membeli sebanyak mungkin domain yang sekiranya akan terakses ketika seseorang menulis domain dengan salah, misalnya adalah facebool.com. Para pelaku biasanya akan memasang iklan secara massal pada laman yang diakses, selain itu, juga ada kemungkinan adanya praktik phising atau malware.

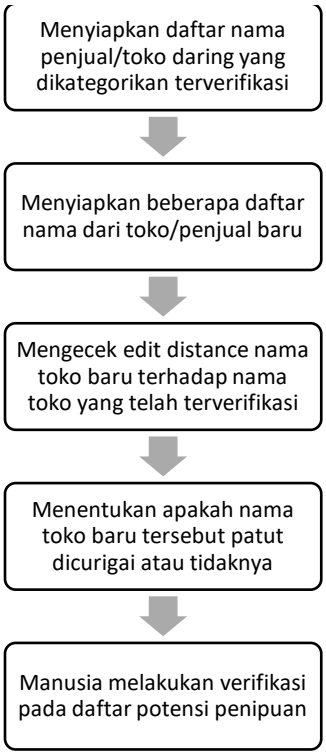


***Gambar 4**. Ilustrasi typosquatting pada facebook.com*
Sumber gambar : Ilustasi pada paper berjudul The Long "Taile" of Typosquatting Domain Names

Selain pada praktik penulisan domain, typosquatting juga dapat ditemui pada penipuan untuk mengatasnamakan sebuah perusahaan yang besar, contohnya adalah berusahaan toko jual beli di Indonesia dengan nama Bukalapak.com, bisa saja seseorang akan mengatasnamakan Bukalapaks.com untuk mengelabui orang lain. Atau misalnya, salah satu perusahaan Mie Instan terbesar di Indonesia, PT Indofood. Seseorang yang berniat curang akan mengatasnamakan PT Indofoos atau PT Indifood untuk menipu.

III. ANALISIS

Untuk mendapatkan hasil yang bisa dianalisis, penulis pertama – tama akan membuat sebuah skema yang menjadi garis besar haluan percobaan ini. Berikut adalah skema secara garis besar, langkah langkah yang akan digunakan untuk melakukan pengambilan data.



Gambar 4. *Skema pengujian*

Kita akan mengamsusikan, bahwa sebuah nama toko/penjual akan dikategorikan sebagai berpotensi penipuan apabila, nama toko tersebut memiliki *Damerau-Levenshtein* edit distance sebanyak <= 3.

A. Menyiapkan daftar nama toko yang telah terve

Pada mulanya, kita akan menyiapkan daftar nama toko yang sekiranya populer, toko yang populer tersebut, di dapatkan melalui situs pencarian seperti halnya *google, bing,* atau *duck duck go*. Selain manual menggunakan *google*, kita dapat menggunakan AI seperti Alexa untuk mencari nama - nama toko daring yang populer.

Dalam makalah ini, penulis akan menggunakan beberapa nama berikut, dengan asumsi bahwa nama – nama toko berikut telah terverifikasi pada sistem yang dimiliki penulis:

```
String[] verified = {
   “Samsung Store”,
   “Xiaomi Official Store”,
   “LG Indonesia”,
   “Oppo Official Store Indonesia”,
   “Under-Armour Store”,
   . . .,
   “Official Nike Indonesia”
}
```

B. Menyiapkan beberapa daftar nama dari toko/penjual baru

Dalam hal menyiapkan nama - nama baru, untuk media uji coba perlu diperhatikan beberapa hal agar hasil uji yang dihasilkan terbilang representatif dan mampu menutupi semua kasus. Maka kita penulis mengelompokan menjadi beberapa jenis. Beberapa jenis yang dimaksud adalah berikut:

```
String[] candidate ={
  “Samsunk Store”,
  “Xioami Officials Store”,
  “LG Indo”,
  “Opo Oficial Stores Indo”,
  “Upper-Amuor Store”,
  . . .
  “Officials Nile Indonesia” }
```

C. Pengecekan

Untuk melakukan pengujian, pertama kita akan pilih salah satu dari *string* pada tabel candidate. Dalam hal ini, penulis ingin memiliki *string* “Samsunk Store” untuk kemudian dicocokan dengan beberapa *string* pada tabel verified.

Untuk Samsung Store dibandingkan dengan Samsung Store. Berikut adalah metrik penyuntingan yang bisa dihasilkan:

Tabel 3*. Hasil tabulasi minimum penyuntingan dari “Samsunk Store” terhadap “Samsung Store”.*

	null	S	a	m	s	u	n	g		s	t	o	r	e
null	0	1	2	3	4	5	6	7	8	9	10	11	12	13
S	1	0	1	2	3	4	5	6	7	8	9	10	11	12
a	2	1	0	1	2	3	4	5	6	7	8	9	10	11
m	3	2	1	0	1	2	3	4	5	6	7	8	9	10
s	4	3	2	1	0	1	2	3	4	5	6	7	8	9
u	5	4	3	2	1	0	1	2	3	4	5	6	7	8
n	6	5	4	3	2	1	0	1	2	3	4	5	6	7
k	7	6	5	4	3	2	1	1	2	3	4	5	7	7
	8	7	6	5	4	3	2	2	1	2	3	4	5	6
S	9	8	7	6	5	4	3	3	2	1	2	3	4	5
t	10	9	8	7	6	5	4	4	3	2	1	2	3	4
o	11	10	9	8	7	6	5	5	4	3	2	1	2	3
r	12	11	10	9	8	7	6	6	5	4	3	2	1	2
e	13	12	11	10	9	8	7	7	6	5	4	3	2	**1**

Dari metrik tersebut, dapat dilihat bahwa operasi minimum yang dibutuhkan untuk mengubah "Samsunk Store" menjadi "Samsung Store" hanyalah 1. Sedangkan jika terhadap *string* yang lainnya, pasti lebih dari 1, namun nilai minimum dari semua nilai minimum edit terhadap *string* – *string* pada tabel verified adalah 1, maka program yang penulis buat akan memasukan *string* "Samsunk Store" ke dalam list yang berpotensi menyebabkan penipuan.

Untuk melihat *string* lainnya, yang tidak akan masuk ke dalam list yang berpotensi menyebabkan penipuan, kita akan meninjau *string* "LG Indo" terhadap "LG Indonesia".

Dengan metode yang sama seperti digunakan sebelumnya, maka akan dihasilkan tabel berikut.

***Tabel 4**. Hasil tabulasi minimum penyuntingan dari "LG Indo" terhadap "LG Indonesia".*

	null	L	G		I	N	D	O
Null	0	1	2	3	4	5	6	7
L	1	0	1	2	3	4	5	6
G	2	1	0	1	2	3	4	5
	3	2	1	0	1	2	3	4
I	4	3	2	1	0	2	2	3
N	5	4	3	2	1	0	1	2
D	6	5	4	3	2	1	0	1
O	7	6	5	4	3	2	1	0
N	8	7	6	5	4	3	2	1
E	9	8	7	6	5	4	3	2
S	10	9	8	7	6	5	4	3
I	11	10	9	8	7	6	5	4
A	12	11	10	9	8	7	6	**5**

Dari tabel tersebut, didapatkan bahwa minimum operasi yang dibutuhkan untuk mengubah "LG Indo" menjadi "LG Indonesia" adalah 5. Dan dari asumsi yang telah penulis buat sebelumnya, suatu *string* nama penjual akan dikategorikan sebagai berpotensi menimbulkan penipuan, apabila ditemukan minimum penyuntingan kurang dari 3. Maka toko LG Indo akan dinyatakan aman.

Kemudian kita akan melakukan proses yang sama, terhadap semua *string* baik pada list Candidate terhadap semua *string* pada list Verified.

Berikut adalah tabel secara lengkap, dari daftar *string* Candidate dibandingkan dengan setiap *string* yang ada pada list Verified.

***Tabel 4**. Hasil Pencarian Operasi Penyuntingan minimum dari setiap string.*

	Samsung Store	Xiaomi Official Store	LG Indo-nesia	Oppo Official Store Indonesia	Under-Armour Store	Offi-cial Nike Indonesia	minimum
Samsunk Store	1	13	12	23	10	18	1
Xioami Officials Store	13	3	18	17	15	20	3
LG Indo	11	18	5	23	17	17	5
Opo Ofic-ial Sto-res Indo	17	12	17	8	18	14	8
Upper-Am-uor Store	10	15	16	21	2	19	2
Officials Nile Indonesia	18	20	13	9	20	2	2

Dari hasil tersebut maka kita akan mendapati sebuah list yang berisi nama - nama toko berpotensi adalah sebagai berikut.

```
String[] fraudPotential = {
  "Samsunk Store",
  "Xioami Officials Store",
  "Upper-Amuor Store",
  . . .
  "Officials Nile Indonesia"
}
```

Hasil tersebut kemudian akan dievaluasi sendiri oleh seseorang, sehingga kita dapat membuat keputusan yang tepat apakah sebuah toko tersebut benar – benar memiliki niatan menipu atau tidak. Namun di sisi lain, toko seperti LG Indo dan Opo Official Stores Indonesia, yang lolos dari algoritma ini juga belum tentu merupakan toko daring yan aman terverifikasi. Diperlukan adanya pengecekan berjenjang, menggunakan Susunan algoritma algoritma lainnya untuk benar – benar mendapatkan hasil dengan tingkat akurasi lebih tinggi. Dari 6

contoh yang dibuat, setidaknya terdapat 4 dari 6 yang berhasil ditangkap oleh algoritma ini.

IV. Kesimpulan dan Saran

A. Kesimpulan

Dari bagian – bagian yang telah dituliskan di atas, dapat ditarik kesimpulan bahwa Algoritma *Damerau-Levenshtein*, dapat digunakan sebagai salah satu alternative untuk mendeteksi adanya potensi penipuan. Selain itu, melihat bagaimana cara kerja dari Algoritma *Damerau-Levenshtein*, kita dapat menggolongkan Algoritma tersebut ke dalam golongan pemrograman dinamis karena memiliki sifat "memori". Untuk kompleksitas waktu yang dimiliki oleh Algoritma tersebut setiap kali membandingkan 2 *string*, adalah $O(MN)$, sedangkan untuk kompleksitas ruang yang digunakan adalah $O(MN)$, dengan M adalah panjang dari *string* 1 dan N adalah panjang dari *string* 2.

B. Saran

Sebagai saran dari penulis, Algoritma ini tidak bisa benar – benar efektif 100% mendeteksi adanya potensi penipuan, diperlukan banyak penyempurnaan dengan menggunakan Algoritma – algoritma lainnya. Penulis berharap dengan adanya Algoritma ini, tingkat penipuan yang ada di Indonesia bisa semakin berkurang.

Ucapan Terima Kasih

Ucapan terima kasih penulis haturkan kepada Allah SWT, berkat rahmat-Nya penulis dapat menyelesaikan makalah ini tepat pada waktunya. Penulis juga ingin mengucapkan terima kasih kepada Ibu Dosen Mata Kuliah Strategi Algoritma Teknik Informatika ITB yang telah membimbing penulis selama satu semester sehingga mampu menerima begitu banyak manfaat dan pengetahuan baru, yaitu kepada Dr. Nur Ulfa Maulidevi, ST, M.Sc. Juga kepada semua orang dan sumber yang telah membantu proses pembuatan makalah ini.

References

[1] Anany V Levitin, "Introduction to the Design and Analysis of Algorithms", Addison-Wesley Longman Publishing Co., Inc., 2012, pp 283.

[2] Szurdi et.al, "The Long "Taile" of Typosquatting Domain Names", Chicago : University of Illinois, 2014.

[3] Damerau, Fred J. (March 1964), "A technique for computer detection and correction of spelling errors", Communications of the ACM, 7 (3): 171–176

[4] https://itnext.io/dynamic-programming-vs-divide-and-conquer-2fea680becbe diakses pada 25 April 2019.

[5] Munir, Rinaldi. 2006. Diktat Strategi Algoritma.

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, 26 April 2019

Mohammad Ridwan Hady Arifin
13517007