Penggunaan Pohon dan Permutasi dalam Pencarian Kata yang Valid dan Sub-Anagram Kata pada Aplikasi Scrabble Expert

Adrian Mulyana Nugraha 13515075
Program Studi Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13515075@std.stei.itb.ac.id

***Abstrak*--- Kosakata yang luas akan memudahkan seseorang dalam berkomunikasi, karena orang tersebut akan mengetahui kata yang tepat agar penyampaian berita menjadi lebih efektif. Salah satu cara untuk memperluas kosakata adalah dengan bermain Scrabble. Namun, setiap kata yang dimainkan dalam Scrabble belum tentu kata yang valid. Makalah ini akan menjelaskan mengenai bagaimana aplikasi Scrabble Checker mencari kata yang valid/sah dan sub-anagram yang bisa dibuat dalam database kata Scrabble dengan menggunakan pohon sebagai bentuk pencarian.**

***Kata Kunci*--- Scrabble, Pohon, Validasi, Permainan**

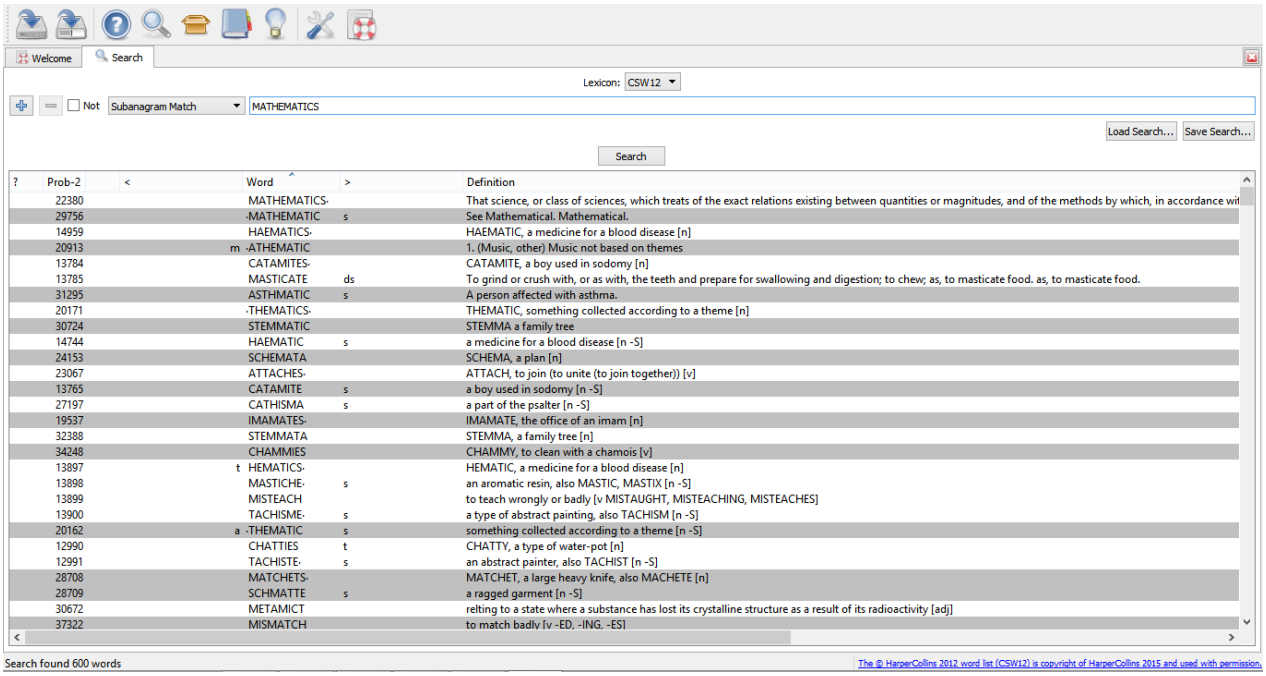
I. PENDAHULUAN

Scrabble merupakan permainan menyusun kata yang tersedia dari 7 kepingan (*tile*) huruf untuk disusun dalam sebuah papan kotak-kotak berukuran 15x15 keping. Pemain terdiri dari 2-4 orang, dan berlomba untuk mengumpulkan poin berdasarkan nilai dari tiap kata. Permainan Scrabble pertama kali diciptakan oleh Alfred Mosher Butts dari New York pada tahun 1933 dengan nama LEXIKO. Alfred terinspirasi dari permainan *board game* yang mengandalkan peluang, dan juga permainan kata/anagram yang tersedia dalam teka-teki silang. Dengan menggabungkan kedua elemen tersebut, maka terbentuklah *board game* legendaris, yang kini dimainkan jutaan orang di seluruh dunia, diadaptasi ke berbagai bahasa termasuk bahasa Indonesia, dan dilombakan di berbagai kalangan, mulai dari SD, SMP, hingga tingkat kuliah dan umum.

Gambar 1.1. Scrabble (sumber : amazon.co.uk)

Dalam permainan Scrabble sendiri, kata yang sah (*valid*) biasanya mengacu pada kamus yang tersedia. Di Indonesia mengacu pada KBBI, sedangkan di negara lain mengacu pada kamusnya masing-masing. Tetapi, untuk standar internasional (dengan bahasa Inggris tentunya), kamus yang menjadi acuan adalah CSW (*Collins Scrabble Words*) edisi 2015. Tercatat, ada 276.663 kata yang terdaftar dalam kamus CSW15, yang terdiri dari 2-15 huruf. Untuk menentukan valid atau tidaknya sebuah kata, selain mencari pada kamus yang membutuhkan waktu yang cukup lama dan ketelitian yang relatif tinggi, pemain dapat menggunakan aplikasi untuk membantu memudahkan proses pengecekan, seperti Scrabble Expert di gawai Android, Collins Zyzzyva di komputer, serta Zarf! di gawai pengguna iOS.

Gambar 1.2. Tampilan Scrabble Expert di Android dengan kata VERITAS untuk pengujian. (Sumber : Scrabble Expert)



Gambar 1.3. Tampilan Collins Zyzzyva pada Komputer dengan kata MATHEMATICS untuk pengujian (Sumber: Collins Zyzzyva 5.0.1)

II. LANDASAN TEORI

Teori yang menjadi dasar dari makalah ini adalah penggunaan pohon, terutama pohon berakar n-ary, dan teori permutasi bentuk umum.

1. Pohon

Pohon merupakan graf tak-terarah terhubung yang tidak mengandung sirkuit. Setiap pasang simpul dalam pohon terhubung dengan lintasan tunggal. Ada 2 macam pohon, yakni pohon merentang (yang berhubungan dengan pohon merentang minimum), dan pohon berakar, yang terdiri dari pohon terurut (bila urutan anak-anak pohon berpengaruh), dan pohon n-ary (bila pohon mempunyai 2 anak, pohon binary. Bila lebih banyak dari 2, pohon n-ary).

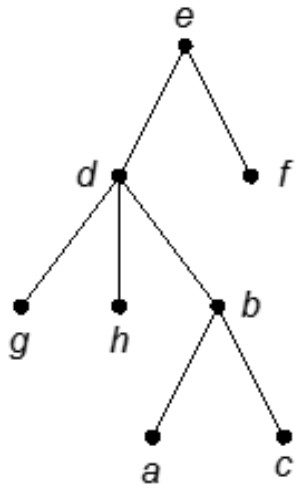
Dalam teori pohon berakar, ada berbagai istilah yang digunakan, yakni:

a. Anak (*child*) dan orang tua (*parent*)

Sebuah simpul dikatakan sebagai anak dari simpul orang tua apabila ada sisi yang menghubungkan antara simpul orang tua dengan anak. Sebagai contoh, pada gambar 2.1, simpul **d** dan **f** merupakan anak simpul **e**, simpul **g**, **h**, dan **b** merupakan anak simpul **d**, simpul **a** dan **c** merupakan anak simpul **b**, dan simpul **a** dan **c** tidak mempunyai anak.

b. Lintasan

Lintasan adalah jarak yang harus ditempuh dari satu simpul orang tua ke simpul anak yang dihitung dari jumlah sisi yang harus dilewati, atau jumlah simpul yang dilewati dikurangi 1. Sebagai contoh, dari gambar 2.1, lintasan e-h melewati simpul **e**, **d**, dan **h**. Panjang lintasan tersebut adalah (3-1 simpul) = 2.



Gambar 2.1. Pohon Berakar (sumber: buku Matematika Diskrit, Ed. 3, Rinaldi Munir)

c. Saudara kandung (*sibling*)

Layaknya saudara kandung dalam pohon keluarga, saudara kandung dari simpul adalah simpul lain yang berderajat sama dan memiliki simpul *parent* yang sama. Dari gambar 2.1, simpul **h** dan **b** merupakan saudara kandung simpul **g**.

d. Leluhur (*ancestor*) dan Keturunan (*descendant*)

Suatu simpul dikatakan sebagai keturunan dari simpul lain dan sebaliknya, apabila ada lintasan dalam pohon yang menghubungkan kedua simpul tersebut. Dalam gambar 2.1, simpul **e** merupakan leluhur simpul **h**, dan sebaliknya, simpul **h** merupakan keturunan simpul **e**.

e. Upapohon (*subtree*)

Upapohon adalah sebuah pohon yang mengandung semua anak dari simpul *parent* utama, dengan akar yang berasal dari simpul anak. Sebagai contoh, pada gambar 2.1, simpul **b** merupakan sebuah upapohon dari pohon dengan simpul akar **e**.

f. Derajat (*degree*)

Derajat menunjukkan jumlah anak atau jumlah upapohon dari suatu simpul.

g. Daun (*leaf*)

Daun merupakan simpul yang tidak memilki keturunan. Sebagai contoh, pada gambar 2.1, simpul **f**, **g**, **h**, **a**, dan **c** adalah daun, karena tidak memiliki anak.

h. Simpul Dalam (*internal nodes*)

Simpul dalam adalah kebalikan dari daun, yakni simpul yang mempunyai anak.

i. Aras / Tingkat (*level*)

Aras dihitung dari panjang lintasan yang dibutuhkan dari simpul akar menuju simpul tersebut, dengan aras simpul akar dimulai dari 0. Aras maksimum dari suatu pohon disebut sebagai tinggi (*height*) atau kedalaman (*depth*) dari pohon.

2. Permutasi

Permutasi merupakan cara penyusunan setiap elemen pada benda, dimana perbedaan urutan penempatan elemen menghasilkan nilai yang berbeda juga. Sebagai contoh, seluruh permutasi dari bilangan **135** adalah **153**, **315**, **351**, **513**, dan **531**. Jika dilihat, maka setiap elemen memiliki nilai yang berbeda. Menurut aturan perkalian, permutasi dari **n** buah objek dapat dirumuskan sebagai:

(**n**)(**n**-1)(**n**-2)...(1) = **n**!

Jika permutasi dari **n** buah objek diletakkan dalam **r** tempat, dimana **n**>**r**, maka hasil permutasi dirumuskan sebagai berikut:

$$P(\mathbf{n},\mathbf{r}) = \frac{(\boldsymbol{n}!)}{(\boldsymbol{n}-\boldsymbol{r})!}$$

Dan permutasi bentuk umum adalah permutasi terhadap suatu himpunan ganda dengan beberapa elemen yang sama. Hasil permutasi **n** buah elemen dengan $\mathbf{n_1}$, $\mathbf{n_2}$, dan seterusnya sampai $\mathbf{n_x}$ elemen sama dirumuskan sebagai berikut:

$$P(\mathbf{n},\mathbf{n_1},\mathbf{n_2},\ldots,\mathbf{n_x}) = \frac{P(\boldsymbol{n}.\boldsymbol{n})}{n_1!\,n_2!\ldots n_x!} = \frac{n!}{n_1!\,n_2!\ldots n_x!}$$

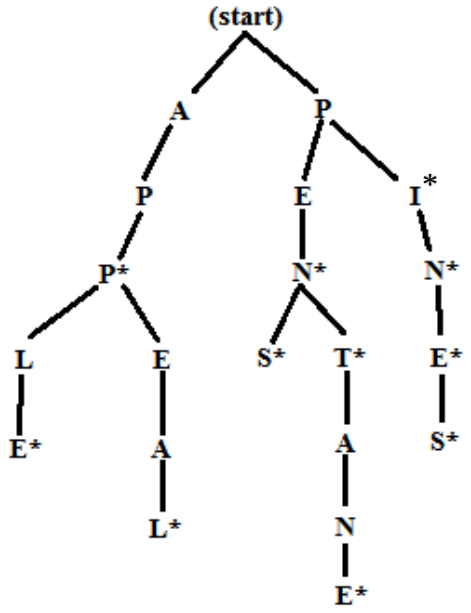
III. CARA PENCARIAN KATA YANG VALID PADA APLIKASI SCRABBLE EXPERT

Salah satu bentuk yang dapat digunakan dalam pencarian kata yang valid dalam permainan Scrabble, adalah dengan membentuk database dari kamus menjadi sebuah pohon. Tetapi, apabila seluruh isi dari kamus CSW15, dengan jumlah 276.663 kata, dijadikan sebagai pohon, maka tentu saja pohon yang dihasilkan akan sangat luas. Secara garis besar, pohon tersebut memiliki derajat maksimum 26 (sebanyak 26 huruf dalam alfabet), dan kedalaman pohon sebesar 15 (panjang maksimum string dalam permainan Scrabble), dan aras minimum 2, karena kata terpendek dalam Scrabble terdiri dari 2 huruf (ada 124 kata yang memiliki panjang 2 huruf).

Sebagai gambaran dari seluruh pohon raksasa tersebut, asumsikan ada daftar kata sebagai berikut (sebagian kecil sekali dari daftar kata CSW15):

APPLE
APPEAL
PEN
PENS
PENTANE
PINES

Apabila daftar kata tersebut dibuat menjadi pohon, maka penampilan pohon tersebut akan menjadi seperti gambar di bawah ini.



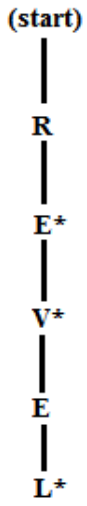
Gambar 3.1. Pohon dari daftar kata

Pada gambar, masing-masing huruf membentuk hubungan anak dan orang tua, dan setiap kata membentuk satu lintasan. Tetapi, ada yang membedakan pada pohon yang dibentuk, yakni dengan adanya beberapa huruf yang diberikan bintang. Fungsi dari bintang tersebut adalah sebagai penanda stop, yang berarti jika pencarian dihentikan pada titik tersebut, maka kata yang dibentuk merupakan kata yang valid meskipun bukan merupakan kata yang ingin dibentuk. Kata yang dibentuk tersebut merupakan *sub-anagram*, yakni sebuah kata baru yang dapat dibentuk dari huruf-huruf yang tersedia dengan panjang yang tidak harus sama dengan kata utama yang

dicari. Sebagai contoh, pada lintasan **PINES**, ada 3 *sub-anagram* yang dapat dibentuk, yakni **PI**, **PIN**, dan **PINE**, sebelum mencapai akhir dari kata **PINES**. Tentu saja, ini tidak melambangkan seluruh *sub-anagram* yang ada dari kata PINES karena masih terpengaruh pada urutan huruf, sedangkan *sub-anagram* yang ada seluruhnya merupakan permutasi dari setiap huruf pembentuk kata yang kemudian divalidasi. Ini akan dibahas pada bab selanjutnya.

Jika dihubungkan dengan bentuk pohon diatas, maka proses validasi dari sebuah kata menggunakan aplikasi Scrabble Checker dapat kita bentuk. Misalkan ada sebuah kata yang ingin divalidasi, yakni **DISCRETE**. Maka pohon akan menelusuri lintasan DISCRETE, yang dimulai dari huruf **D**. Jika aplikasi menemukan anak **I** dari *parent* **D**, maka pencarian akan berlanjut. Jika tidak menemukan anak, maka aplikasi akan mengembalikan nilai *false*, atau kata tidak valid. Kemudian aplikasi akan melakukan pencarian anak **S** dari *parent* **I**, anak **C** dari *parent* **S**, dan seterusnya, hingga pencarian anak terakhir (dalam kasus ini anak **E** dari *parent* **T**). Kemudian, akan dilakukan pengecekan apakah ada tanda bintang dari anak hasil pengecekan terakhir. Jika ada tanda bintang maka kata tersebut valid, dan program akan mengembalikan nilai *true*. Jika tidak ada tanda bintang, maka program akan mengembalikan nilai *false*, atau kata tidak valid.

Tanda bintang menjadi penting dalam pencarian kata valid. Karena, ada *sub-anagram* kata yang sebenarnya tidak valid meskipun akhir dari kata tersebut valid, sehingga apabila dilakukan pencarian terhadap *sub-anagram* kata tersebut akan menghasilkan nilai *false*. Sebagai contoh pencarian kata **REVEL** akan menghasilkan pohon sebagai berikut.



Gambar 3.2. Pencarian dengan pohon dari kata REVEL

Dalam pohon yang ditampilkan, **RE** dan **REV** merupakan *sub-anagram* dari **REVEL**, tetapi **REVE** bukan merupakan *sub-anagram*, dilihat dari tidak adanya tanda bintang pada huruf **E** kedua. Sehingga, apabila dilakukan pencarian terhadap kata **REVE**, akan menghasilkan nilai *false*.

IV. PENCARIAN *SUB-ANAGRAM* KATA MELALUI PROSES PERMUTASI DAN VALIDASI

Menurut kamus CSW15, *anagram* adalah sebuah kata atau frasa yang dibuat dari penyusunan ulang / transposisi huruf-huruf sebuah kata atau frasa lain dengan panjang kata yang sama. Misalnya kata **STAINER** merupakan *anagram* dari kata **RETINAS**. Sedangkan *sub-anagram* sama seperti *anagram*, tapi kata yang dibentuk tidak harus memiliki panjang kata yang sama dengan kata utama.

Untuk mencari seluruh *sub-anagram* yang ada pada suatu kata, karena dengan metode pencarian pohon tidak akan menghasilkan banyak variasi (karena terpaut urutan huruf), maka metode pencarian pohon digabungkan dengan permutasi bentuk umum, sehingga sebuah kata dengan panjang **n** akan menghasilkan **n!** jumlah permutasi dari kata tersebut, bila diasumsikan tidak ada huruf yang sama dalam kata tersebut. Jika ada huruf yang sama, maka jumlah permutasi akan mengikuti rumus permutasi bentuk umum, dengan permutasi yang sama diabaikan. Kemudian akan dilakukan proses pencarian pohon pada seluruh permutasi yang ada, dengan sedikit perbedaan. Pada proses validasi kata di bab sebelumnya, proses pencarian hanya akan mengembalikan nilai *true* apabila pencarian sudah mencapai akhir kata. Dengan kata lain, validasi kata sebelumnya mengabaikan *sub-anagram*. Tapi, dalam pencarian *sub-anagram*, pencarian pohon akan mengembalikan setiap kata yang mungkin, dengan melihat tanda bintang pada anak dari simpul, meskipun pencarian belum mencapai akhir kata. Pencarian kemudian akan dilanjutkan hingga mencapai akhir kata, dan kemudian dilanjutkan dengan permutasi berikutnya. Jika ada permutasi yang menghasilkan kata *sub-anagram* yang sama, maka hasil pencarian tersebut boleh diabaikan.

Sebagai contoh, kata **TEAR** memiliki panjang kata 4, dan tidak ada huruf yang sama. Maka, akan ada 4! = 24 permutasi dari kata **TEAR**, yang dapat dituliskan dalam daftar berikut.

AERT AETR ARET ARTE ATER ATRE
EATR EART ERTA ERAT ETRA ETAR
RAET RATE REAT RETA RTAE RTEA
TARE TAER TERA TEAR TREA TRAE

Pencarian kemudian akan dilakukan dengan dimulai dari permutasi pertama, yakni **AERT**, dan setiap *sub-anagram* yang terbentuk akan dikembalikan ke pengguna. Dalam kasus ini, permutasi **AERT** menghasilkan *sub-anagram* **AE**. Kemudian dilanjutkan dengan permutasi kedua, **AETR**. Karena permutasi ini juga menghasilkan *sub-anagram* **AE**, maka hasil pencarian ini diabaikan. Pencarian dilakukan dengan permutasi berikutnya, hingga mencapai permutasi terakhir (**TRAE** dalam kasus ini). Seluruh *sub-anagram* yang terbentuk akan dikembalikan ke pengguna. Dari 24 permutasi yang disebutkan di atas,

terbentuk 25 *sub-anagram* dari kata **TEAR**, yang dibuktikan dengan gambar berikut.

Gambar 4.1. Hasil pencarian *sub-anagram* kata TEAR di aplikasi Scrabble Expert (Sumber : Scrabble Expert)

Sebagai contoh kedua, akan dilakukan pencarian terhadap kata **GIGA**. Karena ada huruf yang sama pada kata tersebut (2 huruf G), maka jumlah permutasi yang terbentuk menjadi

P(4,2) = $\frac{P(4,4)}{2!} = \frac{4!}{2!}$ = 12 permutasi, dengan cara mengabaikan permutasi yang hasilnya sama, dituliskan sebagai berikut.

AGGI AGIG AIGG GAIG GAGI GGIA

GGAI GIGA GIAG IAGG IGGA IGAG

Kemudian dilakukan metode pencarian yang sama dengan metode pencarian sebelumnya, dimulai dari permutasi pertama hingga terakhir. Dari 12 permutasi diatas, terdapat 7 *sub-anagram* dari kata **GIGA**, yang dibuktikan dengan gambar dibawah ini.

Gambar 4.2. Hasil pencarian *sub-anagram* kata GIGA di aplikasi Scrabble Expert (sumber : Scrabble Expert)

V. KESIMPULAN

Permainan Scrabble adalah permainan papan yang telah ada sejak tahun 1933, dan kini telah menjadi salah satu permainan paling favorit di dunia, dengan banyak adaptasi ke bahasa lokal, dan banyak turnamen yang diadakan baik skala lokal, nasional, maupun internasional.

Dalam permainan Scrabble biasanya menggunakan sebuah kamus sebagai acuan kata, yakni Collins Scrabble Words 2015 (CSW15). Ada lebih dari 270 ribu kata yang tercantum dalam kamus tersebut. Untuk memudahkan pencarian kata, pemain Scrabble dapat menggunakan aplikasi untuk melakukan pencarian kata yang valid, serta seluruh *sub-anagram* dari kata tersebut.

Salah satu metode yang dapat digunakan dalam mencari kata yang valid adalah dengan membentuk seluruh kata dalam kamus CSW15 menjadi sebuah pohon berakar, kemudian melakukan penelusuran lintasan dari pohon tersebut sesuai dengan kata yang ingin dibuktikan. Jika ada lintasan yang dicari dan diakhiri dengan tanda bintang, maka kata tersebut valid.

Pencarian *sub-anagram* kata mirip dengan pencarian kata yang valid, dengan menggunakan permutasi bentuk umum untuk menghasilkan permutasi dari huruf-huruf penyusun kata, kemudian dilakukan proses validasi dari setiap permutasi tersebut.

Penulis mengakui kalau metode pencarian *sub-anagram* yang digunakan termasuk yang kurang efektif, karena metode ini termasuk cara *brute force*, yakni mencari dari seluruh kemungkinan yang ada. Untuk kata yang panjang ($\geq$ 10 huruf), waktu pencarian *sub-anagram* akan relatif lebih lama karena jumlah permutasi yang besar. Ada sebuah metode yang mungkin lebih efektif, yakni menggunakan GADDAG Data Structure, yakni menyimpan seluruh prefiks dan sufiks yang mungkin dalam pohon berakar, setelah kata-kata dalam kamus CSW15. Proses ini memakan lebih banyak memori tetapi menghasilkan waktu pencarian yang lebih cepat.

VI. UCAPAN TERIMA KASIH

Penulis mengucapkan syukur kepada Tuhan Yang Maha Esa, karena makalah ini akhirnya dapat tersusun dengan baik. Penulis berterima kasih kepada Dr. Ir. Rinaldi Munir, yang telah memberikan materi dan bimbingan dalam kuliah Matematika Diskrit. Penulis juga mengucapkan terima kasih kepada seluruh pihak yang telah memberikan dukungan baik secara fisik maupun moral sehingga penyusunan makalah ini dapat selesai tepat waktu.

DAFTAR PUSTAKA

[1] Munir, Rinaldi. 2006. *Matematika Diskrit*. Bandung:Penerbit Informatika

[2] http://stackoverflow.com/questions/880559/algorithm-to-get-a-list-of-all-words-that-are-anagrams-of-all-substrings-scrabb, Diakses 7 Desember 2016 pk. 19.21

[3] http://stackoverflow.com/questions/2497986/writing-an-algorithm-for-scrabble, Diakses 7 Desember 2016 pk. 20.00

[4] http://www.scrabble-assoc.com/info/history.html, Diakses 7 Desember 2016 pk. 22.10

[5]http://www.harpercollins.com.au/9780007589166/collins-official-scrabble-words-fourth-edition, Diakses 8 Desember 2016 pk. 22.08

[6]https://www.merriam-webster.com/dictionary/anagram, Diakses 9 Desember 2016 pk. 08.25

[7] http://textmechanic.com/text-tools/combination-permutation-tools/permutation-generator/, Diakses 9 Desember 2016 pk. 08.46

[8] https://nullwords.wordpress.com/2013/02/27/gaddag-data-structure/, Diakses 9 Desember 2016 pk. 10.45

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, 9 Desember 2016

Adrian Mulyana Nugraha

13515075