# CHEAT ENGINE SCRABLE GAME DENGAN ALGORITMA GADDAG

Antonius Widjaja
Program Studi Teknik Informatika, Unika Soegijapranata Semarang
antoniuskom@gmail.com

## Abstract

*Scrabble is a word game that tests the player's vocabulary. The difficulty of this game depends on the player's vocabulary knowledge. The more vocabularies known by the players, the easier it is to play. This project aims to help players who only know a few vocabularies by building a cheat engine application which implements GADDAG Algorithm. All vocabularies stored in trie data structure as a dictionary. This dictionary will be processed using GADDAG Algorithm to find all vocabularies that can be formed. This application will help player to find the vocabulary with the highest value.*

***Keywords:*** *Cheat Scrabble Game,GADDAG,Trie*

## Pendahuluan

Scrabble adalah permainan menyusun kata. Scrabble terdiri dari 15x15 kotak yang memiliki bonus nilai. Jumlah huruf pada permainan Scrabble memiliki jumlah yang berbeda-beda tiap versi bahasa. Pada jurnal ini menggunakan versi bahasa inggris. Scrabble dapat dimainkan 2 sampai 4 pemain.

Pada permainan Scrabble pemain sering mendapat kesulitan saat menyusun kata. Ini karena kurangnya pengetahuan kosa kata pemain. Untuk mengatasi permasalahan tersebut dibutuhkan sebuah program yang berguna sebagai cheat.

Cara kerja dari cheat tersebut adalah menemukan kata terbaik dari kondisi papan dan huruf ditangan. Kata terbaik ditentukan dari jumlah bonus tertinggi dan jumlah huruf yang paling banyak.

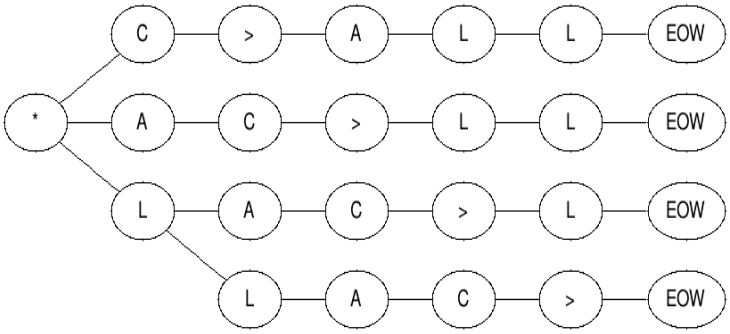
## Landasan Teori

- Data Struktur
  - Trie

    Data struktur yang cocok untuk menyimpan kumpulan kata adalah Trie. Data struktur Trie merupakan data struktur yang serupa dengan Tree. Setiap huruf dari sebuah kata akan disimpan dalam sebuah *Node* yang terhubung dengan *Node* yang bernilai huruf berikutnya.

- Algoritma
  - GADDAG

    Algoritma GADDAG adalah algoritma yang menggunakan metode DAWG yang dikembangkan untuk mempercepat pencarian sebuah kata dalam permainan Scrabble. Algoritma GADDAG membentuk sebuah kata menjadi beberapa bentuk yang memiliki pola. Polanya adalah menyimpan setiap huruf dari sebuah kata menjadi awal kata (*prefix*).




*Gambar 1: Pola GADDAG di Data Struktur Trie*

*source : https://nullwords.wordpress.com/2013/02/27/gaddag-data-structure/*

Pada gambar di atas adalah contoh menyimpan sebuah kata “CARE” Setiap bentuk diberi tanda “>” yang berguna untuk sebagai penanda huruf awal sebuah kata. Dengan penyimpanan seperti ini akan membutuhkan memori yang besar maka perlu dilakukan *compressing.* Pada algoritma GADDAG ada 2 procedure yang berjalan secara recursive yaitu Goon dan Gen. Gen berguna untuk memasangkan huruf ke koordinat tertentu dari huruf di tangan. Huruf tersebut akan dicek melalui method Goon. Goon berfungsi untuk mengecek apakah suatu huruf dapat dipasangkan pada huruf tertentu.

Pencarian diawali dengan mencari koordinat yang memungkinkan untuk dipasang huruf (Anchor Square). Setiap Anchor Square memiliki huruf parent yaitu huruf yang berada di sekitarnya. Dari setiap Anchor Square dilakukan pengecekan. Langkah – langkah pencarian di setiap Anchor Square kata sebagai berikut :

1. Cari kaki dari root ke huruf parent, jika ditemukan lanjut ke lankah 2 dan set root sebagai node dari kaki root. Posisi awal adalah 0 yaitu tepat di koordinat Anchor Square.

2. Pasang huruf pada dari rack ke papan, jika memiliki jalur lanjut ke langkah 3, hilangkan huruf tersebut yang ada di rack
3. jika memiliki arti tambahkan ke record
4. jika bernilai “>” maka posisi bernilai 1
5. Jika posisi <=0 geser posisi ke depan , Jika posisi > 0 posisi bergeser ke belakang
6. Lanjut ke langkah 2

Cara yang sama digunakan untuk mencari kata yang dapat terbentuk dalam posisi vertical.

- Scrabble

Scrabble memiliki beberapa versi. Versi yang digunakan pada jurnal ini adalah versi bahasa Inggris. Setiap pemain mendapatkan 7 huruf yang dimiliki (*Rack*). Scrable memliki beberapa bonus ditiap kotak, yaitu:

- DL = *Double Letter*, Nilai sebuah huruf dikali 2
- DW = *Double Word*, Nilai dari sebuah kata dikali 2
- TL = *Triple Letter*, Nilai sebuah huruf dikali 3
- TW = *Triple Word*, Nilai sebuah kata dikali 3.

Nilai sebuah kata didapat dari penjumlahan total semua huruf sebuah kata.

[1]*Scrabble edisi Bahasa Inggris memiliki 108 huruf, dengan pembagian sebagai berikut :*

- *1 point: E ×12, A ×9, I ×9, O ×8, N ×6, R ×6, T ×6, L ×4, S*
- *×4, U ×4*
- *2 points: W ×5, D ×4, G ×3*
- *3 points: B ×3, C ×3, M ×3, P ×3*
- *4 points: F ×2, H ×2, V ×2, Y ×2*
- *5 points: K ×2*
- *8 points: J ×1, X ×1*
- *10points: Q ×1, Z ×1*

1 https://en.wikipedia.org/wiki/Scrabble_letter_distributions#English, accessed on 05 September 20165

## Metodologi Penelitian

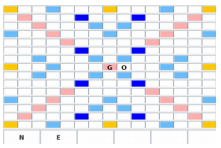
1. Analisa, penelitian dimulai dengan mempelajari permainan Scrabble. Scrabble memiliki beberapa versi dengan peraturan yang berbeda-beda. Setelah menentukan versi yang akan digunakan.
2. Pengumpulan Data, data yang digunakan dalam jurnal ini adalah sebuah kumpulan kosa kata (1000 kata). Kumpulan kosa kata ini berguna sebagai kamus. Setelah menemukan kumpulan kata menentukan bahasa pemrograman yang akan digunakan (pada jurnal ini menggunakan bahasa pemrograman *Java*).
3. Desain, sebelum masuk ke proses pembuatan program terlebih dahulu membuat desain program. Desain program dibuat dalam bentuk *DFD*, *Flowchart*, dan *Class Diagram.* Setelah itu masuk ke proses pemrograman. Proses desain diperlukan untuk mempermudah proses pemrograman.

   Tes dan Implementasi, Setelah proses pemrograman selesai masuk ke proses tes dan implementasi. Program akan di tes kemampuan mencari kosa kata dengan game Scrabble.

## Hasil dan Pembahasan

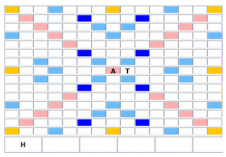
Berikut adalah beberapa hasil percobaan untuk mencari kata yang dapat terbentuk dengan nilai bonus tertinggi dari beberapa kondisi papan. Percobaan pertama adalah menambahkan huruf di belakang kata pada papan, di depan kata pada papan dan di depan dan belakang kata pada papan.

Tabel 1 : Percobaan Menambah Huruf di belakang kata

| Kondisi Papan | | Kondisi Rack | POLA GADDAG | Hasil |
|---|---|---|---|---|
|  | GO (Anchor Square di baris 7 kolom 6) | ENZXCLK | <NE |  |

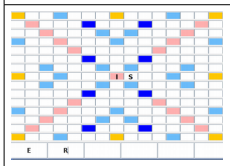
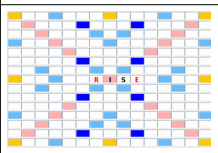
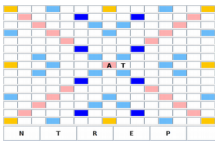
| | | | | |
|---|---|---|---|---|
|  | NO (Anchor Square di baris 7 kolom 6) | RXTCHZK | <ORTH |  |

Tabel 2 : Percobaan Menambah Huruf didepan kata

| Kondisi Papan | | Kondisi Rack | POLA GADDAG | Hasil |
|---|---|---|---|---|
|  | AT (Anchor Square di baris 7 kolom 6) | HZXCVQW | H |  |
|  | OR (Anchor Square di baris 7 kolom 6) | OZXCPMV | OOP>R |  |

Pada percobaan 2 tidak ditemukan tanda > pada pola GADDAG. Hal itu terjadi karena huruf yang terbentuk berada di depan dari kata. Contohya kata POOR memiliki pola P>OOR, OP>OR, OOP>R, dan ROOP. Huruf pada papan adalah OR dimana pengecekan dimulai dari huruf O. Maka pola yang dipakai adalah yang memiliki awalan O yaitu OOP>R dan OP>OR. OOP>R valid karena

Tabel 3 : Percobaan Menambah Huruf didepan dan di belakang kata

| Kondisi Papan | | Kondisi Rack | POLA GADDAG | Hasil |
|---|---|---|---|---|
|  | IS (Anchor Square di baris 7 kolom 6) | RZXCVQE | R>SE |  |
|  | AT (Anchor Square di baris 7 kolom 6) | TERNPMV | P>TTERN |  |

## Kesimpulan

Menggunakan algoritma GADDAG dapat mencari semua kemungkinan kata yang dapat terbentuk dengan cepat dan dalam berbagai posisi. Tetapi algoritma GADDAG memiliki kelemahan yaitu membutuhkan memori yang sangat besar, karena sebuah kata disimpan menjadi beberapa bentuk. Dengan algortma GADDAG dapat dimanfaatkan sebagai lawan bermain.

## Daftar Pustaka

[1] GADDAG, journal Implementasi Algoritma GADDAG dan Negascout Untuk Optimalisasi Computer Player Dalam Permainan Scrabble Renisa Surya hadikusumah, Rosa Ariani Sukamto, M.T., Harsa Wara Prabawa, M.Pd Program Studi Ilmu Komputer, Fakultas Pendidikan Matematika dan Ilmu Pengetahuan Alam,Universitas Pendidikan Indonesia, 2014

[2] A Faster Scrabble Move Generation Algorithm Steven A. Gordon Department of Mathematics, East Carolina University, Greenville, NC 27858,U.S.A,1994

[3] Bansal, J. C. , Singh, P. K. , Saraswat, M. , Verma, A. , Jadon, S.S. , & Abraham, A. (2011).

[4] Inertia Weight Strategies in Particle Swarm Optimization. *Third World Congress on Nature and Biologically Inspired Computing.* 640 – 647.