Menyelesaikan Permainan Wordament Menggunakan Algoritma Backtracking

Krisna Fathurahman/13511006
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13511006@std.stei.itb.ac.id

Wordament merupakan sebuah permainan yang menjadi trend di kalangan pengguna smartphone saat ini. Permainan ini dimainkan dengan cara mencari kata yang dapat dibentuk dari grid karakter berukuran 4x4. Semakin panjang kata yang dipilih dan semakin banyak kata yang berhasil ditemukan, akan semakin besar poin yang kita dapatkan. Secara teori, penelusuran dalam membentuk kata dapat dilakukan dengan DFS (*Depth-First Search*). Maka dari itu, algoritma backtracking sangat mungkin dilgunakan pada permainan ini.

Indeks : Wordament, grid, DFS, backtracking.

I. PENDAHULUAN

Wordament adalah sebuah permainan pada smartphone bersistem operasi iOS, Android, dan Windows Phone. Dikarenakan permainan ini dapat dimainkan hampir di semua jenis device, maka tak dapat dipungkiri permainan ini menjadi permainan favorit di kalangan remaja hingga dewasa saat ini.

Gameplay dari Wordament sendiri menyerupai permainan word puzzle pada umumnya seperti Scrabble atau Boggle. Tetapi yang membuat permainan ini menjadi menarik adalah sistem *real-time scoreboard* yang aktual berskala dunia. Dengan sistem itu membuat kita dapat bertanding dengan orang lain dari seluruh dunia secara tidak langsung. Saya pun mencoba memainkan Wordament, saya pun ikut tertarik dengan gameplay yang disajikan.

Setelah saya cukup lama bermain Wordament, saya menemukan hipotesis bahwa dalam memainkan permainan ini kita dapat menelusuri seluruh isi dari grid karakter berukuran 4x4 yang ada pada permainan ini menggunakan DFS (*Depth-First Search*). Sehingga hipotesis saya pun dapat berkembang menjadi "dapatkah saya menggunakan algoritma backtracking dalam menemukan kata di permainan ini?". Maka dari itu, saya akan menguji hipotesis saya melalui makalah ini.

II. LANDASAN TEORI

2.1 Wordament

Wordament™ adalah bentuk yang unik dari permainan

kata-turnamen kata-dimana anda akan bertanding dengan koneksi internet yang mungkin menjadi pencari kata terbaik diseluruh permainan.[1]

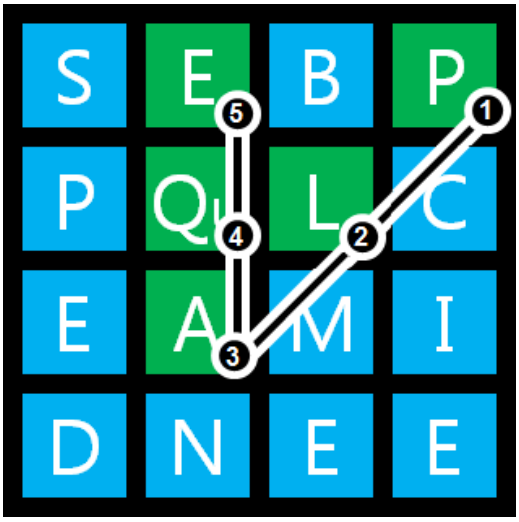
Tujuan dari permainan Wordament adalah menjadi pemain yang berhasil mencetak total skor kata tertinggi dalam satu permainan berdurasi dua menit. Total skor kata adalah akumulasi dari skor kata yang berhasil dibentuk dari grid karakter berukuran 4x4.[1]

Gambar 2-1 Tampilan utama permainan Wordament tiap ronde[1]

Pada setiap awal ronde permainan anda akan disajikan sebuah grid board yang terdiri dari 16 tile (ukuran 4x4) berisi karakter tunggal (contohnya "A" atau "B") atau karakter ganda (contohnya "Qu", "Th" atau "Pro"). Setiap tile yang berisi karakter tersebut memiliki skor yang bernilai antara 1-10. Buatlah kata dari karakter-karakter yang ada di dalam game board. Suatu kata dibentuk dari

berbagai sequence terhubung antar tile dari berbagai arah, tetapi kita hanya boleh menggunakan satu tile tersebut satu kali.[1]

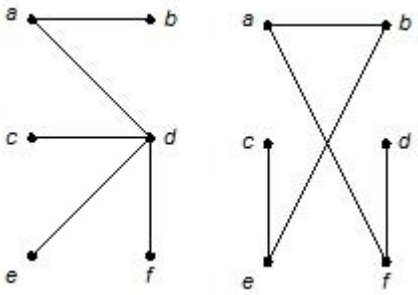
Untuk menambah sebuah kata ke dalam list kata yang anda miliki, sentuh tile pertama dari kata tersebut lalu drag jari anda di sekitar board dengan menghubungkan tiap tile sebanyak-banyaknya hanya satu kali. Jika anda berhasil menyelesaikan kata, angkat jari anda dari tile terakhir. Saat anda memilih (drag) tiap tile, tile tersebut akan terhighlight dan anda akan melihat kata yang sedang dibangun di atas game board. Ketika anda mengangkat jari anda dari layar, kata yang dibentuk akan berwarna hijau (jika benar), kuning (jika telah ditemukan sebelumnya), atau merah (kata yang tidak valid). Jika kata yang dibentuk baru dan benar/valid, kata anda akan dihitung berapa skornya dan total skor anda akan diupdate secara otomatis. Tetap mengumpulkan kata-kata secepat mungkin hingga waktu habis. Membentuk kata yang lebih panjang dan menggunakan tile dengan value lebih besar akan menghasilkan skor yang lebih besar.[1]



Gambar 2-2 Contoh membentuk kata pada Wordament[1]

2.2 Pohon

Pohon adalah graf tak berarah terhubung yang tidak mengandung sirkuit.[2]



Gambar 2-3 Pohon[2]

2.2.1 Sifat-sifat Pohon

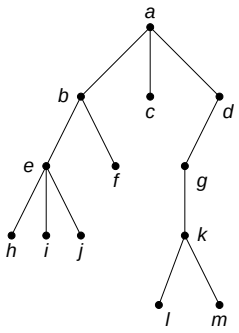
Teorema :

Misalkan G = (V, E) adalah graf tak-berarah sederhana dan jumlah simpulnya *n*. Maka, semua pernyataan di bawah ini adalah ekivalen :

1. G adalah pohon.
2. Setiap pasang simpul di dalam G terhubung dengan lintasan tunggal.
3. G terhubung dan memiliki $m = n - 1$ buah sisi.
4. G tidak mengandung sirkuit dan memiliki $m = n - 1$ buah sisi.
5. G tidak mengandung sirkuit dan penambahan satu sisi pada graf akan membuat hanya satu sirkuit.
6. G terhubung dan semua sisinya adalah jembatan.

Teorema di atas dapat dikatakan sebagai definisi lain dari pohon. [2]

2.2.2 Pohon berakar



Pohon yang satu buah simpulnya diperlakukan sebagai akar dan sisi-sisinya diberi arah sehingga menjadi graf berarah dinamakan pohon berakar (*rooted tree*).[2]

2.2.2.1 Terminologi pada Pohon Berakar

Anak (*child* atau *children*) dan Orangtua (*parent*). *b,c*, dan *d* adalah anak-anak simpul *a*, *a* adalah orangtua dari anak-anak itu. Lintasan dari *a* ke *j* adalah *a,b,e,j*. Panjang lintasan dari *a* ke *j* adalah 3. *f* adalah saudara kandung *e*, tetapi *g* bukan saudara kandung *e*, karena orangtua mereka berbeda. Derajat sebuah simpul adalah jumlah upapohon (atau jumlah anak) pada simpul tersebut. Derajat *a* adalah 3, derajat *b* adalah 2, derajat *d* adalah satu dan derajat *c* adalah 0. Jadi, derajat yang dimaksudkan di sini adalah derajat-keluar. Derajat maksimum dari semua simpul merupakan derajat pohon itu sendiri. Simpul yang berderajat nol (atau tidak mempunyai anak) disebut daun. Simpul *h, i, j, f, c, l,* dan *m* adalah daun. Simpul yang mempunyai anak disebut simpul dalam. Simpul *b, d, e, g,* dan *k* adalah simpul dalam. Pohon di atas memiliki 4 tingkat(*level*) {0,1,2,3,4}. Tingkat meksimum dari suatu pohon disebut tinggi atau kedalaman pohon tersebut. Pohon ini mempunyai tinggi 4.[2]

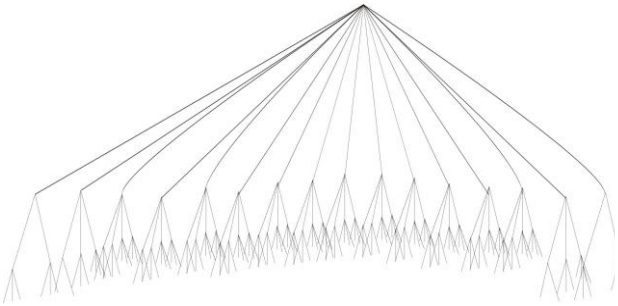
2.2.3 Permodelan Grid menjadi Pohon Ruang Solusi

Jika terdapat suatu grid berukuran 4x4 sebagai berikut :

A	B	C	D
E	F	G	H
I	J	K	L
M	N	O	P

Tabel 2-1 Grid berukuran 4x4

, maka akan dapat dibuat Pohon Ruang Solusi sebagai berikut :



Gambar 2-4 Pohon Solusi

2.3 DFS

Depth-First Search adalah sebuah cara untuk mentraverse sebuah graf. Prinsip dari algoritma DFS simpel : maju telusuri (secara mendalam) selama masih memungkinkan, jika tidak memungkinkan lakukan *backtrack*.[3]

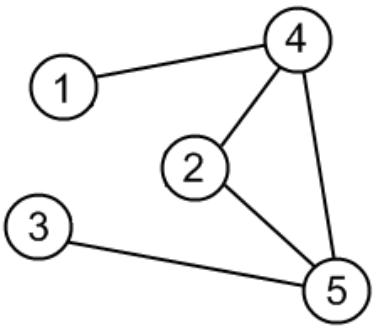
2.3.1 Algoritma DFS

Awalnya semua simpul belum kita kunjungi. DFS dimulai dari simpul root (dibangkitkan bebas/random atau ditentukan dari awal) dengan mengikuti langkah-langkah sebagai berikut :

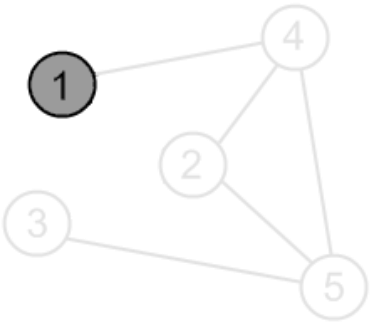
1. Tandai simpul u
2. Untuk setiap sisi (u,v), dimana v belum dikunjungi, jalankan DFS untuk u secara rekursif
3. Tandai simpul u sebagai simpul yang sudah diproses dan lakukan backtrack pada simpul parent[3]

Contoh pengaplikasian Algoritma DFS :

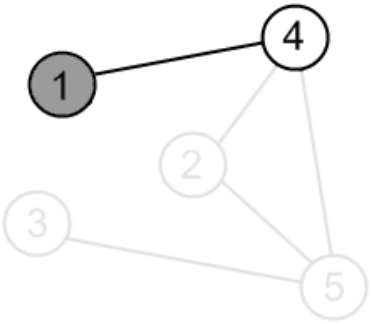
1. Diberikan graf sebagai berikut



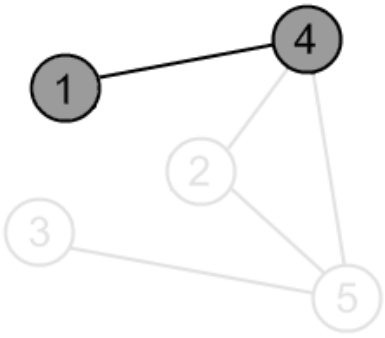
2. Tandai simpul 1 sebagai yang sedang diproses



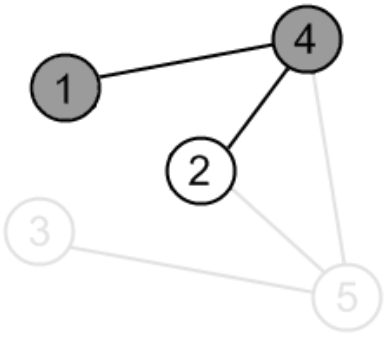
3. Karena ada sisi (1,4) dan simpul 4 belum dikunjungi maka kita kunjungi kesana



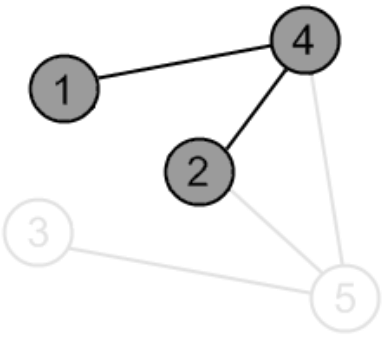
4. Tandai simpul 4 sebagai yang sedang diproses



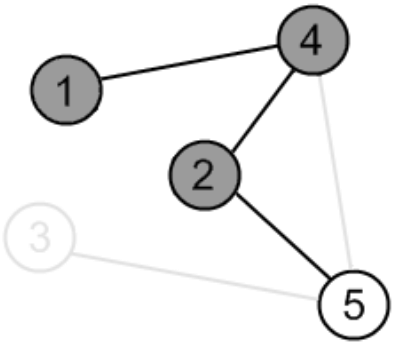
5. Karena ada sisi (4,2) dan simpul 2 belum dikunjungi maka kita kunjungi kesana



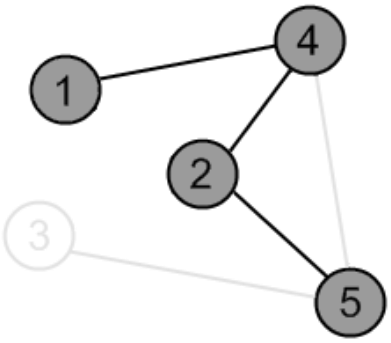
6. Tandai simpul 2 sebagai yang sedang diproses



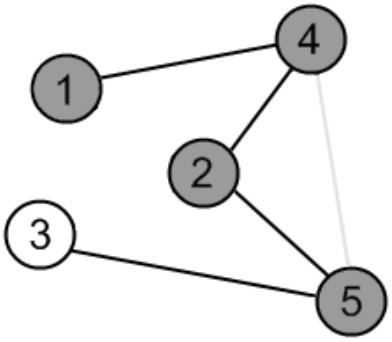
7. Karena ada sisi (2,5) dan simpul 5 belum dikunjungi maka kita kunjungi kesana



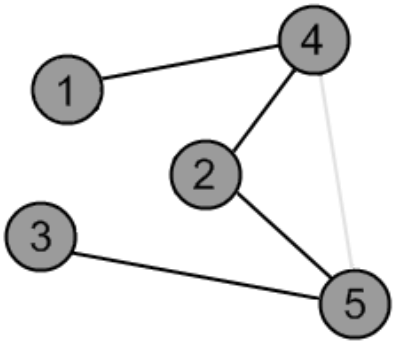
8. Tandai simpul 5 sebagai yang sedang diproses



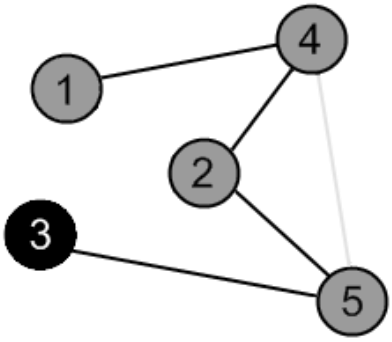
9. Karena ada sisi (5,3) dan simpul 3 belum dikunjungi maka kita kunjungi kesana



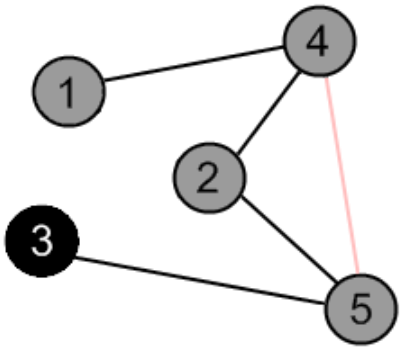
10. Tandai simpul 3 sebagai yang sedang diproses



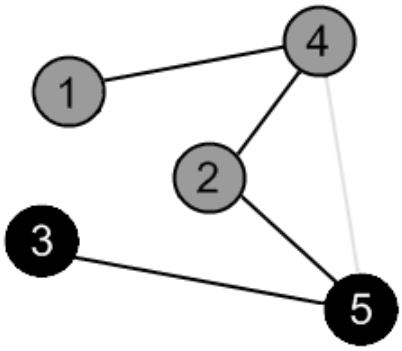
11. Karena tidak ada lagi sisi yang terhubung dari simpul 3, kita tandai simpul 3 dengan yang sudah diproses dan backtrack ke simpul 5



12. Periksa sisi (5,4). Karena simpul 4 sedang diproses maka tidak dapat memungkinkan

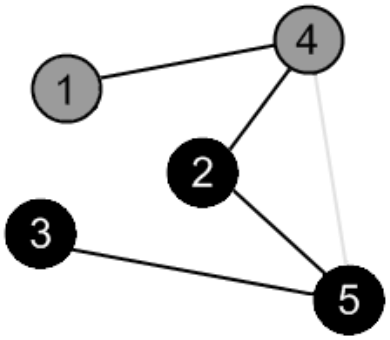


13. Karena tidak ada lagi sisi yang terhubung dari simpul 5, kita tandai simpul 5 dengan yang sudah diproses dan backtrack ke simpul 2

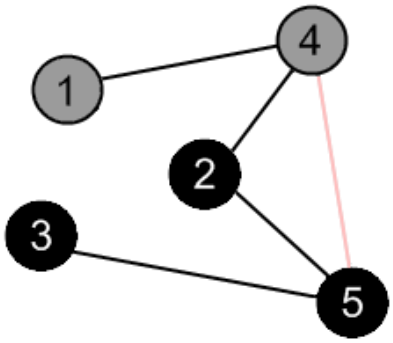


14. Karena tidak ada lagi sisi yang terhubung dari

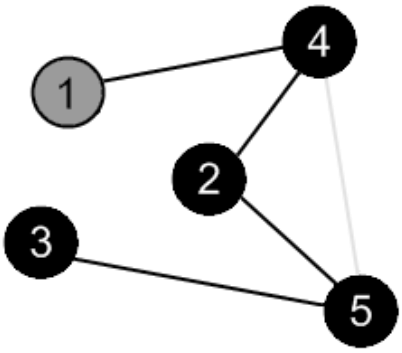
simpul 2, kita tandai simpul 2 dengan yang sudah diproses dan backtrack ke simpul 4



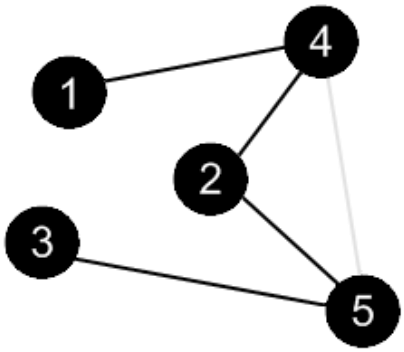
15. Periksa sisi (4,5). Karena simpul 5 sudah diproses maka tidak dapat memungkinkan



16. Karena tidak ada lagi sisi yang terhubung dari simpul 4, kita tandai simpul 4 dengan yang sudah diproses dan backtrack ke simpul 1



17. Karena tidak ada lagi sisi, kita selesai pada simpul 1. Tandai simpul 1 dengan yang sudah diproses. DFS selesai



2.4 Backtracking

Runut-balik (*backtracking*) adalah algoritma yang berbasis pada DFS untuk mencari solusi persoalan secara lebih mangkus. Runut-balik merupakan perbaikan dari algoritma brute-force, secara sistematis mencari solusi persoalan di antara semua kemungkinan solusi yang ada. Istilah Runut-balik pertama kali diperkenalkan oleh D. H. Lehmer pada tahun 1950.[4]

2.4.1 Prinsip Pencarian Solusi dengan Metode Runut-balik

- Solusi dicari dengan membentuk lintasan dari akar ke daun. Aturan pembentukan yang dipakai adalah mengikuti aturan pencarian mendalam (DFS)
- Simpul-simpul yang sudah dilahirkan dinamakan simpul hidup (live node)
- 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" sehingga menjadi simpul mati (dead node)
- Fungsi yang digunakan untuk membunuh simpul-E adalah dengan menerapkan fungsi pembatas (bounding function)
- Simpul yang sudah mati tidak akan pernah diperluas lagi
- Pencarian dihentikan bila kita telah menemukan solusi atau tidak ada lagi simpul hidup untuk runut-balik[4]

III. PEMBAHASAN

Kita analisis soal grid dengan soal sebagai berikut :

Gambar 3-1 Contoh persoalan Wordament[5]

Kita akan lakukan DFS pertama dengan memulai pencarian dari tile 'E' pada sudut kiri atas dan didapatkan list karakter seperti ini :

E-H-C-I-O-E-A-M-R-D-I-O-T-E-G

List diatas didapat menggunakan penelusuran mendalam dengan urutan prioritas ATAS-KANAN-BAWAH-KIRI. Saat mendapatkan list karakter, akan dicari apakah terdapat kata yang dikandung pada list tersebut. Bahasa yang digunakan untuk mencari kata adalah menggunakan bahasa inggris. Pencarian akan dibacktrack jika tidak dapat dilanjutkan. Lalu setelah dilakukan backtrack, akan diciptakan list yang baru. Berikut ini adalah sebagian list-list yang berhasil di generate :

E-H-C-I-O-E-A-M-R-D-I-O-T-E-G
E-H-C-I-O-E-A-M-R-D-I-E-G
E-H-C-I-O-E-A-M-R-D-I-E-T-O
E-H-C-I-O-E-A-M-R-D-G-E-I-O-T
E-H-C-I-O-E-A-M-R-D-G-E-T-O-I

Pada list-list di atas terdapat kata yang valid pada bahasa inggris. Kata-kata tersebut adalah :

E-H-C-I-O-E-A-M-R-**D-I-E-T**-O
E-H-C-I-O-E-A-M-R-D-**G-E-T**-O-I

Jika terdapat kata dalam bahasa inggris, masukan kedalam list kata.

IV. KESIMPULAN

Dengan algoritma backtracking, kita dapat menyelesaikan permainan Wordament. Penggunaan algoritma backtracking pada Wordament hanyalah salah satu pemanfaatan ilmu Strategi Algoritma dalam kehidupan kita sehari-hari karena sesungguhnya masih banyak lagi algoritma yang cocok digunakan pada permainan Wordament dan tidaklah mustahil jika algoritma yang lain memiliki kompleksitas waktu dan memori yang lebih mangkus.

REFERENCES

- http://wordament.com/how-to-play-wordament/ Terakhir diakses pada 14 Desember 2013 pukul 09.00 WIB
- Munir. Rinaldi. "Presentasi IF2091 Struktur Diskrit" Kurikulum 2008-2013
- http://www.algolist.net/Algorithms/Graph/Undirected/Depth-first_search Terakhir diakses pada 20 Desember 2013 pukul 12.41 WIB
- Munir. Rinaldi. "Algoritma Runut-balik (*Backtracking*)". Bahan Kuliah IF2251 Strategi Algoritmik
- www.bestwp7games.com Terakhir diakses pada 20 Desember 2013 pada pukul 14.30 WIB

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, 20 Desember 2013

Krisna Fathurahman/13511006