Penerapan Algoritma BFS dan DFS pada Permainan Capitals

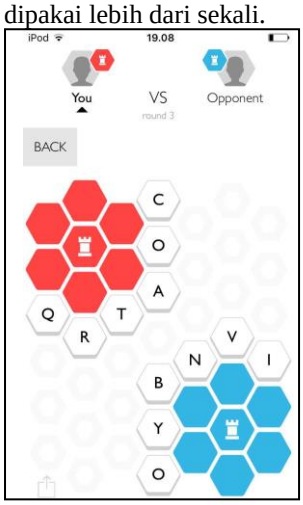
Friska 13514042
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung, 40132
13514042@std.stei.itb.ac.id

***Abstrak*—Makalah ini ditujukan untuk melihat implementasi dari algoritma BFS dan DFS dalam pembentukan kata pada permainan Capitals. Pembuatan makalah ini didasari keinginan penulis untuk membahas lebih dalam mengenai algoritma BFS dan DFS yang menjamin pemberian solusi selama berada dalam lingkup berhingga. Makalah ini diharapkan menjadi salah satu bentuk dokumentasi pengetahuan di bidan informatika.**

Kata Kunci — pembentukan, penyusunan, kata, algoritma, BFS, DFS, Capitals.

I. PENGANTAR

Capitals adalah sebuah permainan pembentukan kata yang disediakan oleh Apple Store. Permainan ini menguji kemampuan pemain untuk menyusun kata dalam Bahasa Inggris dari huruf - huruf acak yang disediakan permainan. Dalam satu kali bertanding, permainan dimainkan oleh minimal dua orang. Permainan memiliki sel – sel yang berbentuk heksagon yang dapat diperluas dengan menggunakan huruf – huruf di sekitar sel tersebut, meski demikian tidak ada batasan untuk menggunakan huruf yang mana saja di manapun lokasinya dengan catatan satu huruf tersedia tidak bisa dipakai lebih dari sekali.



Gambar 1: Tampilan permainan Capitals

Kedua pihak perlu mempertahankan pion mereka (satu buah) yang sudah diberikan pada awal permainan dan berada dalam sel sendiri. Kedua pihak dapat saling menyerang dengan cara mengembangkan daerah teritori (memperluas sel) dan memakan sel lawan dengan cara menggunakan huruf – huruf yang terletak bersebelahan dengan sel lawan dalam pembentukan kata. Kata yang disusun harus sah (*valid*) dan tersedia dalam Bahasa Inggris. Kata yang diajukan oleh pemain akan dicek oleh sistem kebenarannya.

Setiap huruf yang digunakan dalam kata akan dibalas dengan pemberian sel oleh permainan. Jadi, semakin panjang kata maka semakin besar sel yang bertambah sehingga kemungkinan memenangkan permainan semakin besar. Permainan juga akan memberikan tambahan huruf – huruf baru di tempat tertentu setiap kali perluasan sel terjadi. Pertandingan dinyatakan berakhir bila sel yang berisi pion milik salah satu pemain termakan akibat penggunaan huruf di sekitar sel tersebut. Satu kali permainan bisa terdiri dari sekian kali pertandingan (lebih dari satu). Setiap kali pertandingan berakhir maka sel milik pemain yang kalah akan berkurang. Permainan berakhir bila salah satu pemain tidak memiliki sel tersisa dari pertandingan.

Penyusunan kata yang digunakan dalam permainan Capitals perlu dipikirkan dengan baik agar mendapat solusi paling menguntungkan untuk pemain. Penyusunan kata dapat dibuat dengan menerapkan algoritma BFS dan DFS. Dengan kedua algoritma tersebut, didata semua kata yang sah dan mungkin terbentuk dari semua huruf yang mungkin. Daftar semua solusi tersebut kemudian dipilih kata yang paling menguntungkan, dalam makalah ini akan dibahas berdasarkan panjang kata.

II. TEORI YANG BERKAITAN

A. *Algoritma DFS*

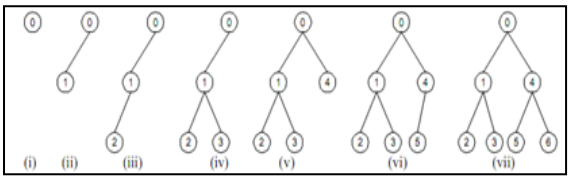
Depth-First Search (yang selanjutnya akan disingkat DFS) digunakan sebagai algoritma yang memberikan solusi dengan mrmproses simpul – simpul pada sebuah graf. DFS melakukan traversal atau kunjungan simpul yang dapat dicapai dari akar (suatu sumber pencarian). DFS sendiri dapat diterapkan

menjadi graf statis dan graf dinamis. Pada graf statis, graf sudah tersedia beserta seluruh simpul yang akan dikenakan algoritma DFS. Pencarian dilakukan dengan memilih sebuah simpul menjadi simpul akar kemudian dilakukan kunjungan ke simpul yang bertetangga.

DFS diterjemahkan sebagai pencarian mendalam karena dipilih simpul tertentu untuk dikunjungi dan dibahas sesuai ketentuan yang dibuat pemrogram. Lebih jelasnya, bila ada lebih dari satu simpul tetangga, program akan memilih simpul spesifik sesuai algoritma yang dibuat pemrogram. Pada setiap simpul diperiksa ketercapaian solusi. Proses ini terus dilakukan sampai tidak ada lagi simpul yang belum dikunjungi. Setelah itu, dilakuan proses runut balik (*backtrack*) ke simpul yang dikunjungi sebelumnya (orang tua atau *parent*) dari simpul terakhir dengan syarat simpul tersebut mempunyai simpul tetangga yang belum dikunjungi. Proses kunjungan dilakukan kembali dilanjutkan dengan runut balik, demikian seterusnya sampai tidak ada lagi simpul yang belum dikunjungi.

Pada graf dinamis, hal yang berbeda adalah simpul – simpul pada graf tersebut. Simpul pada graf dinamis dibangkitkan pada saat keberjalanan, bukan sudah tersedia dari smula sebagaimana pada graf statis. Graf dinamis membangkitkan seluruh simpul tetangga dari sebuah simpul namun tetap mengunjungi simpul spesifik sesuai algoritma pemrogram. Hal lainnya tidak berbeda dengan graf statis, tetap dilakukan kunjungan dan juga proses runut balik.

Berikut adalah gambaran pemrosesan simpul pada graf dengan algoritma DFS:



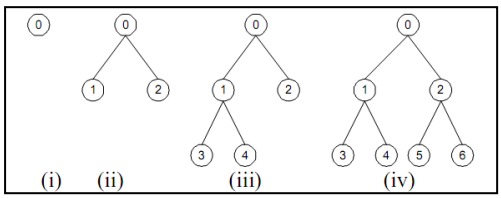
Gambar 2 : Pembentukan Pohon Ruang Status DFS

Penggunaan graf statis dan graf dinamis bergantung pada keadaan data. *Completeness* atau jaminan keterdapatan solusi pada DFS dapat disebut ya dengan catatan ada batasan berhingga pada data. Kompleksitas waktu untuk algoritma DFS adalah $O(b^m)$ dan kompleksitas ruang adalah O(bm) dengan b adalah *branching factor* (maksimum pencabangan yang mungkin dari suatu simpul), d adalah *depth* (kedalaman dari solusi terbaik), dan m adalah maksimum kedalaman dari ruang status. Keuntungan dari DFS adalah kebutuhan memorinya lanjar seiring perunutan simpul. Hal ini berbeda dengan algoritma BFS. Sayangnya, tidak ada jaminan bahwa pencarian mendalam akan memberikan solusi yang minimal.

B. Algoritma BFS

Breadth-First Search (yang selanjutnya akan disingkat BFS) adalah pengembangan lain untuk algoritma traversal graf (seperti algoritma DFS). BFS diterjemahkan menjadi pencarian melebar sesuai dengan metode dan langkah kunjungan simpul yang dilakukan. Berbeda dengan algoritma DFS, algoritma BFS tidak hanya mengambil satu simpul untuk ditelusuri dan dirunut terus menerus sampai akhirnya perlu dilakukan runut balik. Algoritma BFS membuka jalan untuk semua cabang / daun yang ada pada graf untuk dirunut dan ditelusuri. Dapat dikatakan bahwa algoritma BFS merunut sistem pada level tertentu. Seperti algoritma DFS yang bisa diterapkan pada graf statis dan graf dinamis, algoritma BFS juga demikian. Pada graf statis, seluruh simpul yang akan dikunjungi telah tersedia. Pada graf dinamis, BFS membangkitkan simpul seiring berjalannya proses pengecekan dan kunjungan. Algoritma BFS memilih sebuah simpul yang dijadikan akar atau sumber pencarian. Dari akar, algoritma membangkitkan seluruh simpul tetangga dari akar tersebut kemudian dilakukan kunjungan kepada setiap simpul yang berada pada level tersebut. Pada setiap simpul diperiksa ketercapaian solusi. Untuk seterusnya, pada setiap simpul akan dibangkitkan simpul tetangganya dan proses pengecekan dimulai kembali. Proses BFS tidak menerapkan runut balik.

Berikut adalah pembentukan pohon ruang status algoritma BFS:



Gambar 3 : Pembentukan Pohon Ruang Status BFS

Algoritma BFS akan menghasilkan solusi (*completeness* bernilai baik). Kompleksitas waktu untuk algoritma BFS adalah $O(b^d)$ dan kompleksitas ruang adalah $O(b^d)$ dengan b adalah *branching factor* (maksimum pencabangan yang mungkin dari suatu simpul), d adalah *depth* (kedalaman dari solusi terbaik).

III. PEMBAHASAN

Before you begin to format your paper, first write and save the content as a separate text file. Keep your text and graphic files separate until after the text has been formatted and styled. Do not use hard tabs, and limit use of hard returns to only one return at the end of a paragraph. Do not add any kind of pagination anywhere in the paper. Do not number text heads-the template will do that for you.

Finally, complete content and organizational editing before formatting. Please take note of the following items when proofreading spelling and grammar:

A. Pencarian Kata pada Permainan Capitals

Pada Capitals, pemain harus mengerahkan kemampuannya untuk menyusun huruf – huruf yang ada sehingga membentuk sebuah kata bermakna dalam bahasa inggris. Peserta bisa memilih semua huruf yang tersedia, meski memilih huruf di lokasi tertentu dapat membuat peserta menang atau melangkah menuju kemenangan. Kombinasi huruf yang ada pada permainan ini akan ditentukan lewat algoritma BFS dan DFS. Dengan algoritma BFS dan DFS, ditentukan gabungan dari huruf – huruf yang diharap membentuk kata. Huruf yang tersedia tidak boleh dipakai dua kali dan tidak semua huruf harus digunakan. Pencarian akan terus dilakukan sampai

menemukan semua kemungkinan kata yang layak dan sah (*valid*) tersedia dalam Bahasa Inggris kemudian dimasukkan sebuah fungsi khusus untuk menentukan kata yang akan dibentuk. Algoritma BFS dan DFS memiliki kompleksitas waktu dan ruang yang berbeda.Untuk kedua algoritma solusi pasti ditemukan karena jumlah huruf yang tersedia berhingga.

Penulis memutuskan memberikan contoh penerapan algoritma BFS dan DFS pada kasus yang terjadi pada permainan di Gambar 1.

B. Algoritma BFS pada Permainan Capitals

Huruf – huruf yang disediakan oleh permainan, misal berjumlah n buah harus didaftarkan terlebih dahulu agar pemain dapat mengetahui dan memperkirakan kombinasi yang dapat dibentuk dari huruf – huruf tersebut.Daftar huruf tersebut dapat disimpan dalam struktur apa saja karena tidak memengaruhi permainan. Permainan membebaskan pemain untuk memilih huruf yang mana saja selama tidak berulang. Dalam makalah ini, penulis menyarankan untuk menyimpan keterangan lokasi huruf untuk dijadikan strategi dalam permainan.

Pertama – tama, diperlukan sebuah huruf awal, misal huruf X yang akan digunakan untuk menentukan kombinasi huruf lainnya. Tidak seperti persoalan jarak yang diketahui posisi awal (kota tempat seseorang bermula untuk pergi), pada permainan tidak ada huruf yang harus dipilih menjadi huruf pertama. Pada makalah ini, akan dibahas bahwa huruf pertama yang dipilih adalah secara acak (*random*). Hal ini dilakukan karena lebih mungkin sama dengan keadaan di kehidupan nyata yang mana pemain akan mempertimbangkan satu huruf sebagai permulaan kata dan mencoba membentuk kata dengan kombinasi huruf lainnya.

Huruf tersebut kemudian dijadikan akar dari sebuah graf. Dibangkitkan daun yang menyimpan huruf lain yang berjumlah $n-1$ buah (tidak termasuk huruf X mengingat huruf tidak boleh berulang). Pada setiap simpul diperiksa apakah solusi telah tersedia, dalam hal ini solusi adalah urutan jalur yakni gabungan huruf – huruf pada pembangkitan simpul dicek apakah telah membentuk kata yang sah. Bila iya, kata akan disimpan dalam sebuah struktur, misalkan list. Pencarian dilanjutkan untuk menemukan semua solusi kata yang mungkin. Masing – masing daun akan membangkitkan simpul lagi (daun baru) yang berisi huruf lain yang belum dituliskan pada jalur tersebut. Pengecekan terhadap solusi terus dilakukan sampai akhirnya semua kombinasi huruf telah teruji. Setelah itu, dilakukan pembentukan kata dengan cara yang sama namun huruf pertama yang dipilih adalah huruf berbeda.

Semua kemungkinan kata ini kemudian diurutkan berdasarkan keuntungan yang akan diberikan bila pemain menggunakan kata tersebut. Pemain dapat memilih kata yang paling panjang atau kata yang semua atau sebagian hurufnya terletak menyentuh sel lawan.

Untuk lebih menjelaskan pembentukan kata dapat dilihat contoh huruf yang tersedia:

Dari Gambar 1, didapatkan daftar huruf yang disediakan permainan saat itu adalah {C,O,A,T,R,Q,I,V,N,B,Y,O}. Akan dibahas dua alternatif pembentukan kata dengan huruf awal berbeda menggunakan algoritma BFS untuk menunjukkan perbedaan yang didapat.

TABLE I. TABEL PENELUSURAN PERMAINAN DENGAN BFS ALTERNATIF 1

No.	Simpul Ekspansi	Simpul Hidup - Dibangkitkan
1	C	$O_C,A_C,T_C,R_C,Q_C,I_C,V_C,N_C,B_C,Y_C,O_C$
2	Oc	$A_{CO},T_{CO},R_{CO},Q_{CO},I_{CO},V_{CO},N_{CO},B_{CO},Y_{CO},O_{CO}$
3	A_C	$O_{CA},T_{CA},R_{CA},Q_{CA},I_{CA},V_{CA},N_{CA},B_{CA},Y_{CA},O_{CA}$
4	T_C	$O_{CT},A_{CT},R_{CT},Q_{CT},I_{CT},V_{CT},N_{CT},B_{CT},Y_{CT},O_{CT}$
5	R_C	$O_{CR},A_{CR},T_{CR},Q_{CR},I_{CR},V_{CR},N_{CR},B_{CR},Y_{CR},O_{CR}$
6	Q_C	$O_{CQ},A_{CQ},T_{CQ},R_{CQ},I_{CQ},V_{CQ},N_{CQ},B_{CQ},Y_{CQ},O_{CQ}$
7	I_C	$O_{CI},A_{CI},T_{CI},R_{CI},Q_{CI},V_{CI},N_{CI},B_{CI},Y_{CI},O_{CI}$
8	V_C	$O_{CV},A_{CV},T_{CV},R_{CV},Q_{CV},I_{CV},N_{CV},B_{CV},Y_{CV},O_{CV}$
9	N_C	$O_{CN},A_{CN},T_{CN},R_{CN},Q_{CN},I_{CN},V_{CN},B_{CN},Y_{CN},O_{CN}$
10	B_C	$O_{CB},A_{CB},T_{CB},R_{CB},Q_{CB},I_{CB},V_{CB},N_{CB},Y_{CB},O_{CB}$
11	Y_C	$O_{CY},A_{CY},T_{CY},R_{CY},Q_{CY},I_{CY},V_{CY},N_{CY},B_{CY},O_{CY}$
12	O_C	$O_{CO},A_{CO},T_{CO},R_{CO},Q_{CO},I_{CO},V_{CO},N_{CO},B_{CO},Y_{CO}$
13	A_{CO}	$T_{COA},R_{COA},Q_{COA},I_{COA},V_{COA},N_{COA},B_{COA},Y_{COA},O_{COA}$
14	…	Proses dilanjutkan seturut data elemen di baris 2
15	Y_{CO}	$T_{COY},R_{COY},Q_{COY},I_{COY},V_{COY},N_{COY},B_{COY}$
16	O_{CA} … Y_{CA}	Proses dilanjutkan seturut data elemen di baris 3
17	O_{CT} … Y_{CT}	Proses dilanjutkanseturut data elemen di baris 4
18	…	Proses dilanjutkan untuk semua data yang diperoleh dari baris 5 sampai 10
19	O_{CY} … B_{CY}	Proses dilanjutkanseturut data elemen di baris 11
20	T_{COA}	$R_{COAT},Q_{COAT},I_{COAT},V_{COAT},N_{COAT},B_{COAT},Y_{COAT},O_{COAT}$
21	…	Proses dilanjutkan seturut data elemen di baris 13
22	Y_{COA}	T_{COAY}, $R_{COAY},Q_{COAY},I_{COAY},V_{COAY},N_{COAY},B_{COAY},Y_{COAY},O_{COAY}$
23	..	Proses yang sudah dilangsungkan yakni membangkitkan kombinasi huruf yang terbentuk dan mengurus semua elemen di dalamnya dahulu kemudian beralih ke simpul expansi lainnya dilanjutkan terus sampai semua huruf telah terpakai

Tabel 1 adalah salah satu alternatif pembentukan kata dengan algoritma BFS yang menggunakan huruf awal “C”. Dari penelusuran didapat kata yang sah dalam Bahasa Inggris yakni “COAT” yang berarti mantel (ditandai dengan warna merah, berada pada baris ke-13).

TABLE II. TABEL PENELUSURAN PERMAINAN DENGAN BFS ALTERNATIF 2

No.	Simpul Ekspansi	Simpul Hidup - Dibangkitkan
1	T	C_T ,O_T,A_T, R_T,Q_T,I_T,V_T,N_T,B_T,Y_T,O_T
2	C_T	O_{TC}, A_{TC},R_{TC},Q_{TC},I_{TC},V_{TC},N_{TC},B_{TC},Y_{TC},O_{TC}
3	O_T	C_{TO}, A_{TO}, R_{TO},Q_{TO},I_{TO},V_{TO},N_{TO},B_{TO},Y_{TO},O_{TO}
4	A_T	C_{TA},. O_{TA}, ,R_{TA},Q_{TA},I_{TA},V_{TA},N_{TA},B_{TA},Y_{TA},O_{TA}
5	R_T	C_{TR}. O_{TR},A_{TR} ,Q_{TR},I_{TR},V_{TR},N_{TR},B_{TR},Y_{TR},O_{TR}
6	Q_T	C_{TQ} ,O_{TQ},A_{TQ}, R_{TQ},I_{TQ},V_{TQ},N_{TQ},B_{TQ},Y_{TQ},O_{TQ}
7	I_T	C_{TI} ,O_{TI},A_{TI}, R_{TI},Q_{TI},V_{TI},N_{TI},B_{TI},Y_{TI},O_{TI}
8	V_T	C_{TV}, O_{TV},A_{TV}, R_{TV},Q_{TV},I_{TV},N_{TV},B_{TV},Y_{TV},O_{TV}
9	N_T	C_{TN}, O_{TN},A_{TN}, R_{TN},Q_{TN},I_{TN},V_{TN},B_{TN},Y_{TN},O_{TN}

No.	Simpul Ekspansi	Simpul Hidup - Dibangkitkan
10	B_T	$C_{TB,}$ O_{TB},$A_{TB,}$ R_{TB},Q_{TB},I_{TB},V_{TB},N_{TB},Y_{TB},O_{TB}
11	Y_T	$C_{TY,}$ O_{TY},$A_{TY,}$ R_{TY},Q_{TY},I_{TY},V_{TY},N_{TY},B_{TY},O_{TY}
12	O_T	$C_{TO,}$ O_{TO},$A_{TO,}$ R_{TO},Q_{TO},I_{TO},V_{TO},N_{TO},B_{TO},Y_{TO}
13	O_{TC}	A_{TCO} ,R_{TCO},Q_{TCO},I_{TCO},V_{TCO},N_{TCO},B_{TCO},Y_{TCO},O_{TCO}
14	…	Proses dilanjutkan seturut data elemen di baris 2
15	Y_{TC}	A_{TCY} ,R_{TCY},Q_{TCY},I_{TCY},V_{TCY},N_{TCY},B_{TCY} ,O_{TCY}
16	…	Proses dilanjutkan untuk semua data yang diperoleh dari baris 3 sampai baris 5 elemen C_{TR}
17	A_{TR}	C_{TRA} , O_{TRA} ,Q_{TRA},I_{TRA},V_{TAR},N_{TAR},B_{TRA} , $Y_{TRA,}$ O_{TRA}
18	…	Proses dilanjutkan untuk semua data yang diperoleh dari baris 5 elemen Q_{TR} sampai baris 12
19	A_{TCO}	$R_{TCOA,}$ Q_{TCOA},I_{TCOA},V_{TCOA},N_{TCOA},B_{TCOA} , Y_{TCOA} , O_{TCOA}
20	…	Proses dilanutkanseturut semua data elemen di baris 13
21	Y_{TCO}	A_{TCOY} , $R_{TCOY,}$ Q_{TCOY},I_{TCOY},V_{TCOY},N_{TCOY},B_{TCOY} , O_{TCOY}
22	…	Proses dilanjutkan untuk semua data yang diperoleh dari baris 14 sampai baris 17 elemen Q_{TRA}
23	I_{TRA}	C_{TRAI} , $O_{TRAI,}$ Q_{TRAI}, V_{TRAI},N_{TRAI},B_{TRAI} , Y_{TRAI} , O_{TRAI}
23	..	Proses yang sudah dilangsungkan yakni membangkitkan kombinasi huruf yang terbentuk dan mengurus semua elemen di dalamnya dahulu kemudian beralih ke simpul expansi lainnya dilanjutkan terus sampai semua huruf telah terpakai

Tabel 2 adalah salah satu alternatif pembentukan kata dengan algoritma BFS yang menggunakan huruf awal "T". Dari penelusuran didapat kata yang sah dalam Bahasa Inggris yakni "TRAIN" yang bisa berarti kereta atau latihan. (ditandai dengan warna merah, berada pada baris ke-23).

Kedua tabel adalah perwujudan dari algoritma BFS yang lebih sering dijumpai dalam bentuk graf. Bentuk tabel ini dijadikan alternatif mengingat jumlah huruf yang cukup banyak sehingga graf akan sangat luas. Pada pengerjaan tabel, dipilih satu simpul yang diekspansi kemudian dibangkitkan semua simpul yang bertetangga yakni huruf yang belum dijalankan sebelumnya pada jalur yang dipilih. Dilakukan pengecekan apakah terbentuk kata yang sah, bila iya maka kata dicatat dan pengerjaan terus dilanjutkan. Masing – masing simpul dibangkitkan lagi kombinasi yang mungkin dengan huruf lain kemudian dilakukan pengecekan kembali.

Dari daftar huruf yang ada dapat dilihat adanya huruf yang sama yakni huruf "O". Hal ini tidak berarti bahwa huruf "O" hanya boleh dipakai sekali. Semua huruf "O" yang tersedia dapat dipakai, untuk itu tetap dikembangkan (kemunginan kata). Untuk perunutan dengan algoritma, salah satu simpul "O" ditandai sebagai simpul ekstra (dalam pengerjaan dimisalkan simpul O yang kedua kali ditemukan). Simpul O ekstra tidak akan diekspansi lagi karena akan menghasilkan hasil yang sama dengan simpul "O" yang pertama.

Dari tabel yang dihasilkan, diberikan contoh ditemukan (minimal) dua kata yang dihasilkan dari kombinasi huruf yang tersedia, yakni "COAT" dan "TRAIN". Kedua kata ini dinyatakan sah karena ada dalam Bahasa Inggris (*coat* berarti mantel dan *train* bisa berarti kereta atau latihan). Kata – kata yang ditemukan ditampung untuk kemudian ditentukan kata yang akan dipilih. Kata yang akan dipilih tentunya yang memberi keuntungan terbesar, dalam konteks ini keuntungan terbesar didapat dari panjang kata. Kata "COAT" terdiri dari empat huruf sedangkan "TRAIN" terdiri dari 5 huruf maka kata "TRAIN" yang terpilih. BFS diterapkan untuk mencari semua solusi yang mungkin karena mungkin saja kata yang ditemukan pertama bukanlah kata yang optimal mengingat tidak ada kepastian huruf yang harus diambil (misalkan ditentukan hruuf pertama yang harus diambil). Dapat dilihat bahwa dengan mengambil huruf "C" sebagai permulaan kata, ditemukan kata "COAT" yang sebenarnya sudah merupakan kata yang sah, namun bila pencarian diteruskan (sampai akhirnya dilakukan penggantian huruf), ditemukan kata lain "TRAIN" yang lebih menguntungkan. Bila BFS langsung berhenti saat menemukan solusi, tentu kata "TRAIN" tidak akan ditemukan.

Berbeda dengan penerapan algoritma BFS pada pencarian jalur terpendek (*shortest path*) dengan memperhitungkan jarak, bila ditemukan solusi yang lebih buruk dari jalur terpendek sebelumnya akan dibuang. Sayangnya, pada pencarian kata yang dicari adalah maksimal sehinggatidak ada batasan untuk pembuangan karena niai maksimum lokal (kata menguntungkan yang ditemukan dari sebuah jalur) belum tentu nilai maksimum global (kata menguntungkan yang ditemukan dari semua jalur)

C. Algoritma DFS pada Permainan Capitals

Sama seperti BFS, huruf – huruf yang ada pada permainan harus didaftarkan dahulu misal sejumlah n buah huruf. Dari huruf – huruf yang ada, dipilih satu huruf misal huruf X yang akan menjadi huruf pertama dari kata yang hendak dibuat. Setelah itu dibangkitkan simpul – simpul yang bertetangga yakni huruf – huruf lain selain huruf pertama yang terpilih. Salah satu daun kemudian diproses terlebih dahulu secara khusus. Pada setiap simpul diperiksa apakah solusi telah tersedia, dalam hal ini solusi adalah urutan jalur yakni gabungan huruf – huruf pada pembangkitan simpul dicek apakah telah membentuk kata yang sah. Bila iya, kata akan disimpan dalam sebuah struktur, misalkan list. Pencarian dilanjutkan untuk menemukan semua solusi kata yang mungkin. Daun yang secara khusus dipilih itu akan membangkitkan daun – daun lain yakni huruf selain yang sudah berada pada jalur yang sudah diproses. Proses dilanjutkan dengan memilih satu simpul khusus pada pembangkitan dan seterusnya sampai tidak ada lagi huruf yang belum terpakai. Setelah itu, dilakukan proses runut balik (*backtrack*) kepada orang tua dari simpul terakhir. Orang tua dari simpul terakhir itu menjadi simpul yang diproses untuk dilihat lagi kombinasi yang mungkin dari huruf yang sedang dikerjakan. Proses runut balik terus dilanjutkan sampai sudah kembali lagi ke akar. Setelah proses DFS untuk satu huruf awal tertentu dinyatakan selesai, proses DFS dilanjutkan lagi dengan memilih huruf awal baru.

Untuk lebih menjelaskan pembentukan kata dapat dilihat contoh huruf yang tersedia:

Dari Gambar 1, didapatkan daftar huruf yang disediakan permainan saat itu adalah {C,O,A,T,R,Q,I,V,N,B,Y,O}. Akan

dibahas dua alternatif pembentukan kata dengan huruf awal berbeda menggunakan algoritma DFS untuk menunjukkan perbedaan yang didapat.

TABLE III. TABEL PENELUSURAN PERMAINAN DENGAN DFS ALTERNATIF 1

No.	Simpul Ekspansi	Simpul Hidup - Dibangkitkan
1	C	$O_C,A_C,T_C,R_C,Q_C,I_C,V_C,N_C,B_C,Y_C,O_C$
2	Oc	$A_{CO},T_{CO},R_{CO},Q_{CO},I_{CO},V_{CO},N_{CO},B_{CO},Y_{CO},O_{CO}$
3	A_{CO}	$T_{COA},R_{COA},Q_{COA},I_{COA},V_{COA},N_{COA},B_{COA},Y_{COA},O_{COA},$ $O_{CA},T_{CA},R_{CA},Q_{CA},I_{CA},V_{CA},N_{CA},B_{CA},Y_{CA},O_{CA},$ $A_{CO},T_{CO},R_{CO},Q_{CO},I_{CO},V_{CO},N_{CO},B_{CO},Y_{CO},O_{CO},$ $O_C,A_C,T_C,R_C,Q_C,I_C,V_C,N_C,B_C,Y_C,O_C$
4	T_{COA}	$R_{COAT},Q_{COAT},I_{COAT},V_{COAT},N_{COAT},B_{COAT},Y_{COAT},O_{COAT},$ $R_{COA},Q_{COA},I_{COA},V_{COA},N_{COA},B_{COA},Y_{COA},O_{COA},$ $O_{CA},T_{CA},R_{CA},Q_{CA},I_{CA},V_{CA},N_{CA},B_{CA},Y_{CA},O_{CA},$ $A_{CO},T_{CO},R_{CO},Q_{CO},I_{CO},V_{CO},N_{CO},B_{CO},Y_{CO},O_{CO},$ $O_C,A_C,T_C,R_C,Q_C,I_C,V_C,N_C,B_C,Y_C,O_C$
5	R_{COAT}	Q_{RCOAT}, I_{RCOAT}, $V_{RCOAT},N_{RCOAT},B_{RCOAT},Y_{RCOAT},O_{RCOAT},$ $Q_{COAT},I_{COAT},V_{COAT},N_{COAT},B_{COAT},Y_{COAT},O_{COAT},$ $R_{COA},Q_{COA},I_{COA},V_{COA},N_{COA},B_{COA},Y_{COA},O_{COA},$ $O_{CA},T_{CA},R_{CA},Q_{CA},I_{CA},V_{CA},N_{CA},B_{CA},Y_{CA},O_{CA},$ $A_{CO},T_{CO},R_{CO},Q_{CO},I_{CO},V_{CO},N_{CO},B_{CO},Y_{CO},O_{CO},$ $O_C,A_C,T_C,R_C,Q_C,I_C,V_C,N_C,B_C,Y_C,O_C$
6	Q_{RCOAT}	I_{QRCOAT}, $V_{QRCOAT},N_{QRCOAT},B_{QRCOAT},Y_{QRCOAT},O_{QRCOAT},$ Q_{RCOAT}, I_{RCOAT}, $V_{RCOAT},N_{RCOAT},B_{RCOAT},Y_{RCOAT},O_{RCOAT},$ $Q_{COAT},I_{COAT},V_{COAT},N_{COAT},B_{COAT},Y_{COAT},O_{COAT},$ $R_{COA},Q_{COA},I_{COA},V_{COA},N_{COA},B_{COA},Y_{COA},O_{COA},$ $O_{CA},T_{CA},R_{CA},Q_{CA},I_{CA},V_{CA},N_{CA},B_{CA},Y_{CA},O_{CA},$ $A_{CO},T_{CO},R_{CO},Q_{CO},I_{CO},V_{CO},N_{CO},B_{CO},Y_{CO},O_{CO},$ $O_C,A_C,T_C,R_C,Q_C,I_C,V_C,N_C,B_C,Y_C,O_C$
7	I_{QRCOAT}	$V_{IQRCOAT},N_{IQRCOAT},B_{IQRCOAT},Y_{IQRCOAT},O_{IQRCOAT},V_{QRCOAT},$ $N_{QRCOAT},B_{QRCOAT},Y_{QRCOAT},O_{QRCOAT}$, Q_{RCOAT}, I_{RCOAT}, $V_{RCOAT},N_{RCOAT},B_{RCOAT},Y_{RCOAT},O_{RCOAT},$ $Q_{COAT},I_{COAT},V_{COAT},N_{COAT},B_{COAT},Y_{COAT},O_{COAT},$ $R_{COA},Q_{COA},I_{COA},V_{COA},N_{COA},B_{COA},Y_{COA},O_{COA},$ $O_{CA},T_{CA},R_{CA},Q_{CA},I_{CA},V_{CA},N_{CA},B_{CA},Y_{CA},O_{CA},$ $A_{CO},T_{CO},R_{CO},Q_{CO},I_{CO},V_{CO},N_{CO},B_{CO},Y_{CO},O_{CO},$ $O_C,A_C,T_C,R_C,Q_C,I_C,V_C,N_C,B_C,Y_C,O_C$
8	$V_{IQRCOAT}$	$N_{VIQRCOAT}$, $B_{VIQRCOAT}$, $Y_{VIQRCOAT}$, $O_{VIQRCOAT}$, $N_{IQRCOAT}$, $B_{IQRCOAT}$, $Y_{IQRCOAT}$, $O_{IQRCOAT}$, V_{QRCOAT}, N_{QRCOAT}, B_{QRCOAT},Y_{QRCOAT}, O_{QRCOAT}, Q_{RCOAT}, I_{RCOAT}, $V_{RCOAT},N_{RCOAT},B_{RCOAT},Y_{RCOAT},O_{RCOAT},$ $Q_{COAT},I_{COAT},V_{COAT},N_{COAT},B_{COAT},Y_{COAT},O_{COAT},$ $R_{COA},Q_{COA},I_{COA},V_{COA},N_{COA},B_{COA},Y_{COA},O_{COA},$ $O_{CA},T_{CA},R_{CA},Q_{CA},I_{CA},V_{CA},N_{CA},B_{CA},Y_{CA},O_{CA},$ $A_{CO},T_{CO},R_{CO},Q_{CO},I_{CO},V_{CO},N_{CO},B_{CO},Y_{CO},O_{CO},$ $O_C,A_C,T_C,R_C,Q_C,I_C,V_C,N_C,B_C,Y_C,O_C$
9	$N_{VIQRCOAT}$	$B_{NVIQRCOAT}$, $Y_{NVIQRCOAT}$, $O_{NVIQRCOAT}$, $B_{VIQRCOAT}$, $Y_{VIQRCOAT}$, $O_{VIQRCOAT}$, $N_{IQRCOAT}$, $B_{IQRCOAT}$, $Y_{IQRCOAT}$, $O_{IQRCOAT}$, V_{QRCOAT}, N_{QRCOAT}, B_{QRCOAT},Y_{QRCOAT}, O_{QRCOAT}, Q_{RCOAT}, I_{RCOAT}, $V_{RCOAT},N_{RCOAT},B_{RCOAT},Y_{RCOAT},O_{RCOAT},$ $Q_{COAT},I_{COAT},V_{COAT},N_{COAT},B_{COAT},Y_{COAT},O_{COAT},$ $R_{COA},Q_{COA},I_{COA},V_{COA},N_{COA},B_{COA},Y_{COA},O_{COA},$ $O_{CA},T_{CA},R_{CA},Q_{CA},I_{CA},V_{CA},N_{CA},B_{CA},Y_{CA},O_{CA},$ $A_{CO},T_{CO},R_{CO},Q_{CO},I_{CO},V_{CO},N_{CO},B_{CO},Y_{CO},O_{CO},$ $O_C,A_C,T_C,R_C,Q_C,I_C,V_C,N_C,B_C,Y_C,O_C$
10	$B_{NVIQRCOAT}$	$Y_{BNVIQRCOAT}$, $O_{BNVIQRCOAT}$, $Y_{BNVIQRCOAT}$, $O_{NVIQRCOAT}$, $B_{VIQRCOAT}$, $Y_{VIQRCOAT}$, $O_{VIQRCOAT}$, $N_{IQRCOAT}$, $B_{IQRCOAT}$, $Y_{IQRCOAT}$, $O_{IQRCOAT}$, V_{QRCOAT}, N_{QRCOAT}, B_{QRCOAT},Y_{QRCOAT}, O_{QRCOAT}, Q_{RCOAT}, I_{RCOAT}, $V_{RCOAT},N_{RCOAT},B_{RCOAT},Y_{RCOAT},O_{RCOAT},$ $Q_{COAT},I_{COAT},V_{COAT},N_{COAT},B_{COAT},Y_{COAT},O_{COAT},$ $R_{COA},Q_{COA},I_{COA},V_{COA},N_{COA},B_{COA},Y_{COA},O_{COA},$ $O_{CA},T_{CA},R_{CA},Q_{CA},I_{CA},V_{CA},N_{CA},B_{CA},Y_{CA},O_{CA},$ $A_{CO},T_{CO},R_{CO},Q_{CO},I_{CO},V_{CO},N_{CO},B_{CO},Y_{CO},O_{CO},$ $O_C,A_C,T_C,R_C,Q_C,I_C,V_C,N_C,B_C,Y_C,O_C$
11	$Y_{BNVIQRCOAT}$	$O_{YBNVIQRCOAT}$, $O_{BNVIQRCOAT}$, $Y_{BNVIQRCOAT}$, $O_{NVIQRCOAT}$, $B_{VIQRCOAT}$, $Y_{VIQRCOAT}$, $O_{VIQRCOAT}$, $N_{IQRCOAT}$, $B_{IQRCOAT}$, $Y_{IQRCOAT}$, $O_{IQRCOAT}$, V_{QRCOAT}, N_{QRCOAT}, B_{QRCOAT},Y_{QRCOAT}, O_{QRCOAT}, Q_{RCOAT}, I_{RCOAT}, $V_{RCOAT},N_{RCOAT},B_{RCOAT},Y_{RCOAT},O_{RCOAT},$ $Q_{COAT},I_{COAT},V_{COAT},N_{COAT},B_{COAT},Y_{COAT},O_{COAT},$ $R_{COA},Q_{COA},I_{COA},V_{COA},N_{COA},B_{COA},Y_{COA},O_{COA},$ $O_{CA},T_{CA},R_{CA},Q_{CA},I_{CA},V_{CA},N_{CA},B_{CA},Y_{CA},O_{CA},$ $A_{CO},T_{CO},R_{CO},Q_{CO},I_{CO},V_{CO},N_{CO},B_{CO},Y_{CO},O_{CO},$ $O_C,A_C,T_C,R_C,Q_C,I_C,V_C,N_C,B_C,Y_C,O_C$
12	$O_{YBNVIQRCOAT}$	$O_{BNVIQRCOAT}$, $Y_{BNVIQRCOAT}$, $O_{NVIQRCOAT}$, $B_{VIQRCOAT}$, $Y_{VIQRCOAT}$, $O_{VIQRCOAT}$, $N_{IQRCOAT}$, $B_{IQRCOAT}$, $Y_{IQRCOAT}$, $O_{IQRCOAT}$, V_{QRCOAT}, N_{QRCOAT}, B_{QRCOAT},Y_{QRCOAT}, O_{QRCOAT}, Q_{RCOAT}, I_{RCOAT}, $V_{RCOAT},N_{RCOAT},B_{RCOAT},Y_{RCOAT},O_{RCOAT},$ $Q_{COAT},I_{COAT},V_{COAT},N_{COAT},B_{COAT},Y_{COAT},O_{COAT},$ $R_{COA},Q_{COA},I_{COA},V_{COA},N_{COA},B_{COA},Y_{COA},O_{COA},$ $O_{CA},T_{CA},R_{CA},Q_{CA},I_{CA},V_{CA},N_{CA},B_{CA},Y_{CA},O_{CA},$ $A_{CO},T_{CO},R_{CO},Q_{CO},I_{CO},V_{CO},N_{CO},B_{CO},Y_{CO},O_{CO},$ $O_C,A_C,T_C,R_C,Q_C,I_C,V_C,N_C,B_C,Y_C,O_C$
13	…	Dilakukan proses runut balik dan diterapkan proses pengecekan yang sama sampai akhirnya semua kombinasi kata dikunjungi

TABLE IV. TABEL PENELUSURAN PERMAINAN DENGAN DFS ALTERNATIF 2

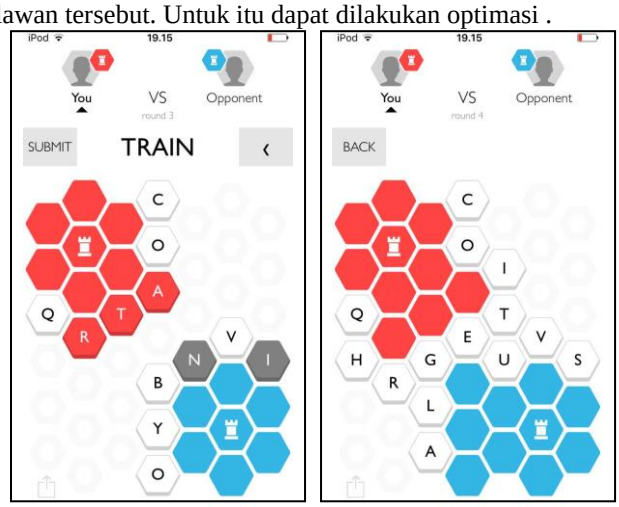
No.	Simpul Ekspansi	Simpul Hidup - Dibangkitkan
1	T	C_T, $O_T,A_T,R_T,Q_T,I_T,V_T,N_T,B_T,Y_T,O_T$
2	…	Proses pembangkitan, pengunjungan, dan runut balik dilakukan sesuai algoritma DFS untuk elemen C_T, O_T,A_T
3	R_T	C_{TR}, O_{TR},A_{TR} ,$Q_{TR},I_{TR},V_{TR},N_{TR},B_{TR},Y_{TR},O_{TR}$
4	…	Proses pembangkitan, pengunjungan, dan runut balik dilakukan sesuai algoritma DFS untuk elemen C_{TR}, O_{TR}
5	A_{TR}	C_{TRA}, $O_{TRA},Q_{TRA},I_{TRA},V_{TRA},N_{TRA},B_{TRA},Y_{TRA}$, O_{TRA}
6	..	Proses pembangkitan, pengunjungan, dan runut balik dilakukan sesuai algoritma DFS untuk elemen C_{TRA}, O_{TRA},Q_{TRA}
7	I_{TRA}	C_{TRAI}, $O_{TRAI},Q_{TRAI},V_{TRAI},N_{TRAI},B_{TRAI},Y_{TRAI}$, O_{TRAI}
8	…	Proses pembangkitan, pengunjungan, dan runut balik dilakukan sesuai algoritma DFS untuk simpul yang belum dikerjakan

Dari daftar huruf yang ada dapat dilihat adanya huruf yang sama yakni huruf "O". Hal ini tidak berarti bahwa huruf "O" hanya boleh dipakai sekali. Semua huruf "O" yang tersedia dapat dipakai, untuk itu tetap dikembangkan (kemunginan kata). Untuk perunutan dengan algoritma, salah satu simpul "O" ditandai sebagai simpul ekstra (dalam pengerjaan dimisalkan simpul O yang kedua kali ditemukan). Simpul O ekstra tidak akan diekspansi lagi karena akan menghasilkan hasil yang sama dengan simpul "O" yang pertama.

Proses pengunjungan simpul dilakukan seperti gambar 2. Algoritma DFS yang diterapkan memilih satu huruf tertentu untuk terus dikunjungi dan dibangkitkan simpul bertetangga lain yang diperoleh dari sana. Setelah dari satu jalur khusus tidak ada lagi huruf yang tersedia, dilakukan proses runut balik (seperti nampak pada proses 12).

D. *Analisis Penggunaan BFS dan DFS pada Permainan Capitals*

Tujuan pemain dalam bermain tentunya untuk mencari kemenangan. Dengan algoritma DFS dan BFS, pemain dapat menentukan kata terbaik yang bisa disusun dari kombinasi huruf yang ada. Kriteria kata terbaik yang dimaksud adalah kata terpanjang karena poin dihitung dari panjang kata. Dibandingkan dengan mengandalkan perkiraan dan ilmu menebak, pemain tentu dapat memiliki kata yang lebih panjang dengan melakukan pendataan dengan algoritma DFS dan BFS. Dari segi kompleksitas, algortima BFS lebih memakan ruang (kompleksitas ruang = $O(b^d)$) dibanding algoritma DFS (kompleksitas ruang = O(bm)). Algoritma BFS memiliki kompleksitas waktu pencarian $O(b^d)$ sedangkan algoritma BFS memiliki kompleksitas waktu $O(b^m)$ dengan b adalah *branching factor*, d adalah kedalaman dari solusi terbaik dan m adalah maksimum kedalaman dari ruang status. Kedua algoritma menjamin ditemukannya solusi. Kata terbaik yang dapat dipilih pemain sebenarnya tidak hanya bergantung pada panjang kata. Bila pemain memilih huruf yang lokasinya bersebelahan dengan sel lawan, pemain dapat mengambil alih sel lawan tersebut. Untuk itu dapat dilakukan optimasi .



Gambar 4: Eksekusi Kata Terpilih pada Permainan (kiri)
Gambar 5: Hasil Eksekusi yang Telah Diinisiasi Kata Baru (kanan)

E. *Optimasi*

Optimasi yang dapat dilakukan adalah dengan mencatat huruf yang berada pada lokasi bersebelahan dengan sel lawan.Pembentukan huruf dapat dilakukan dengan prioritas utama huruf – huruf yang berada pada lokasi tersebut. Algoritma BFS dan DFS yang dipakai tidaklah berubah hanya saja daftar huruf yang dikenakan berubah. Bila sebelumnya semua huruf yang diberikan permainan langsung dikenakan algoritma, kali ini algoritma diterapkan terlebih dahulu pada daftar huruf – huruf yang berada pada lokasi khusus tersebut. Bila ada, dipilih kombinasi terbaik dari kombinasi huruf yang dihasilkan huruf – huruf tersebut. Bila tidak ada, barulah seluruh daftar huruf dikenakan algoritma BFS dan DFS. Dari kombinasi – kombinasi huruf yang dihasilkan, diutamakan kata yang mengandung banyak huruf berlokasi khusus (bersebelahan dengan sel lawan) dan kata yang panjang.

IV. KESIMPULAN

Pembentukan kata dari sejumlah huruf yang disediakan pada permainan "Capitals" dapat dilakukan dengan menggunakan algoritma BFS dan DFS. Kedua algoritma menghasilkan semua solusi yang mungkin (kata yang tersedia dalam Bahasa Inggris) untuk dipilih kata yang paling menguntungkan (kata dengan jumlah huruf terbanyak)

UCAPAN TERIMA KASIH

Penulis mengucpakan terima kasih kepada Tuhan YME atas penyertaan selama pembuatan makalah. Terima kasih juga disampaikan kepada Bu Ulfa dan Pak Rinaldi atas bimbingan dan ilmunya selama perkuliahan IF2211 Strategi Algoritma.

REFERENSI

Presentasi "Algoritma DFS dan BFS" dari http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2014-2015/BFS%20dan%20DFS%20(2015).pptx, diakses 3 Mei 2016.

Munir, Rinaldi. 2009. Diktat Kuliah IF2211 Strategi Algoritma. Bandung: Penerbit ITB

http://intelligence.worldofcomputing.net/ai-search/breadth-first-search.html#.Vy727IR9602, diakses 3 Mei 2016

http://intelligence.worldofcomputing.net/ai-search/depth-first-search.html#.Vy7214R9602, diakses 3 Mei 2016

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis adalah tulisan saya sendiri, bukan saduran, atau terjamahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 8 Mei 2016

Friska
13514042