PENERAPAN ALGORITMA PENCARIAN PADA PERMAINAN BOGGLE

Risa Astari Dewi NIM 13506064

Teknik Informatika, Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10, Bandung
e-mail: if16064@students.if.itb.ac.id

ABSTRAK

Permainan Boggle termasuk dalam permainan olah kata. Kunci dari permainan ini terletak pada pencarian susunan kata yang dapat dibuat dari kumpulan huruf acak yang ditampilkan dalam matriks blok *k x k*. Pemain akan diberi waktu tertentu untuk menebak sebanyak-banyaknya kemungkinan kata. Penilaian terhadap setiap kata ditentukan oleh banyak huruf dan tingkat kesulitan. Kata yang diterima paling tidak mengandung 3 huruf dan tiap blok huruf hanya dipakai satu kali. Permainan ini tergolong sukar bagi orang yang memiliki sedikit perbendaharaan kata. Permainan dapat dibagi menjadi 3 bagian, kamus, mesin pencari dan antarmuka. Kamus menjadi acuan penilaian permainan, apakah kata yang dimasukkan terdaftar resmi. Mesin pencari akan membandingkan masukan dengan isian pada kamus atau memberi petunjuk kemungkinan kata yang dapat dibuat pada pemain. Antarmuka menjadi perantara pemain dan sistem. Tulisan ini membahas tentang algoritma yang dapat diterapkan dalam mesin pencari. Algoritma yang dibandingkan antara DFS dan *brute force* untuk mencari kemungkinan susunan huruf dari huruf acak yang tersedia. Sebagai pelengkap pencarian, pencocokan string akan ditambahkan pada kedua algoritma dan masing-masing menggunakan metoda yang sama.

Kata kunci: Boggle, DFS, hasbro, Scrabbel, weboggle

1. PERMAINAN BOGGLE

Boggle merupakan permainan kata yang didesain oleh Allan Turoff. Permainan ditampilkan dalam bentuk grid blok sebanyak *k x k*. Inti dari permainan ini, pemain dapat menebak sebanyak-banyaknya kata yang dapat disusun dari huruf acak pada blok.

Permainan dimulai dengan mengacak papan blok (misalkan ukuran papan 4 x 4 blok). Tiap blok memiliki huruf berbeda untuk ditampilkan. Waktu standar permainan adalah 3 menit.

Tiap pemain mencari kata yang dapat disusun dari huruf yang ditampilkan. Penggabungan huruf dapat dilakukan mendatar, menurun, diagonal atau kombinasi dari ketiganya. Kata yang disusun minimal memiliki 3 huruf, jamak atau tunggal dan bisa berupa derifatif (permainan ini hanya mendukung bahasa Inggris). Tiap pemain memilih blok huruf dan mencatat tiap kata yang telah disusunnya, dalam persoalan ini pencatatan akan otomatis dilakukan oleh program. Pengecekan kebenaran kata dilakukan sebelum program mencatat kata. Jika kata benar tercantum dalam kamus, program secara otomatis menampilkannya dalam daftar kata dan langsung memberi nilai. Jika tidak program akan me-*reset* blok yang dipilih[3].

Gambar 1. Contoh pemilihan kata

Proses penilaian dimulai dari membaca daftar kata dari para pemain. Jika dua atau lebih pemain menuliskan kata yang sama, kata tersebut tidak masuk dalam penilaian. Selanjutnya tiap kata dihitung jumlah hurufnya, berdasarkan aturan dari *Official Scrabbel Players Dictionary (*OSPD*)*yang juga diterapkan pada Boggle, nilai kata :

3 bernilai 1
4 → 1
5 → 2

6 → 3
7 → 5
8+ → 11

Nilai diatas akan ditambah jika kata yang disusun tergolong unik dilihat dari data kemungkinan kata tertebak dalam kamus[1].

2. PERMASALAHAN PENCARIAN

Permainan ini melakukan 2 jenis pencarian. Pertama saat pemain memilih huruf dan mendaftarkannya. Program akan menjalankan pencarian, apakah susunan huruf yang dimasukkan tercantum dalam kamus.

Untuk permasalahan pencariaan yang pertama, awalnya pemain memilih blok huruf hingga terbentuk susunan tertentu. Ketika pemain menekan tombol *submit* program memulai proses pencocokan. Pada proses ini program membaca huruf pertama yang dimasukkan misalkan EAT, huruf pertama "E". Program mendata kata berawalan "E". Pembacaan diteruskan ke huruf "A", program mengecek semua kata yang memiliki prefiks "EA". Dan ketika program membaca huruf terakhir menjadi "EAT", program mengecek kata yang memiliki prefiks "EAT". Dalam kasus ini ada beberapa kata yang memiliki prefiks yang sama dan sekaligus "EAT" merupakan kata utuh. Program menampilkan kata terpilih dalam daftar dan memberi nilai. Untuk pencarian ini pemakaian algoritma bisa merupakan algoritma pencocokan string biasa.

Pencarian kedua, saat pemain memilih tombol "HINT", program akan secara otomatis menampilkan susunan huruf yang memungkinkan dari deretan huruf yang ada. Jawaban permasalahan disampaikan pada bab selanjutnya.

3. SOLUSI

Analisa menggunakan DFS dengan *brute force* sebagai pembanding.

3.1 *Brute force*

Dasar dari algoritma ini adalah pendekatan yang lempang untuk memecahkan suatu masalah, biasanya didasarkan langsung pada pernyataan masalah. Algoritma *brute force* memecahkan masalah dengan sangat sederhana, langsung dan dengan cara yang jelas [2].

Pada metode ini program akan melakukan kombinasi dari huruf yang ada pada papan blok. Jika diambil contoh papan berukuran 4 x 4

O	A	A	N
E	T	R	I
I	H	K	R
I	F	L	V

Kombinasi dilakukan untuk kemungkinan 3 huruf, 4 huruf, 5 huruf, 6 huruf , 7 huruf hingga 16 huruf. Dari perhitungan tersebut didapat angka 28,448 kombinasi.

```
procedure CariKata
{Mencari kombinasi susunan huruf}
Deklarasi
kata : array [1..16] of char
c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,c13,c14,c15,c16
: char
i,n : integer
algoritma :
i←k x k {ukuran papan}
for n ← 3 to i
        kombinasi(n,kata) {mengkombinasikan huruf
mulai kombinasi 3 hingga 16 huruf pada papan dan
menyimpan dalam kata }
        if kataValid(kata) then {mencari kata dalam
kamus }
            output cetak(kata) {mencetak susunan huruf
yang didapat}
```

Kelemahan algoritma ini , program tidak dapat mengecek apakah suku awal dari susunan 3 huruf ada dalam kamus, semisal tidak ada, pencarian seharusnya akan berhenti untuk susunan 4 huruf dengan suku awal yang sama. Metoda ini tentu menghabiskan waktu yang lama. Terlebih lagi jika ukuran papan lebih besar. Berikut data kasus terburuk untuk ukuran papan k x k

Tabel 1 Tabel kemungkinan kasus terburuk

Ukuran papan	Jumlah kombinasi
3 x 3	620
4 x 4	28,448
5 x 5	3,060,312
6 x 6	873,239,616

Selanjutnya, masih dengan algoritma yang sama tapi program akan membatasi sample huruf yang akan dikombinasikan. Tidak semua huruf akan diperiksa karena program akan melakukan acak untuk memilihnya.

Jumlah kombinasi memang berkurang banyak namun dari hasil eksperimen, kemungkinan kata yang diterima masih kecil. Dapat dilihat pada tabel berikut.

Tabel 2 Tabel kemungkinan dengan pemanggilan acak

Ukuran	Rata-rata jumlah kombinasi	Rata-rata kata yang ditemukan
3x3	118	10
4x4	321	16
5x5	653	37
6x6	1016	49
10x10	3296	158

Metoda ini masih tidak efektif melihat dari kecilnya persentase kata yang ditemukan dari kombinasi susunan huruf yang diperiksa.

3.2 *Depth-First-Search*

Dasar dari algoritma ini adalah penulusuran simpul dimulai dari simpul v. Simpul berikutnya yang dikunjungi adalah simpul w yang bertetangga dengan simpul t, lalu pencarian mendalam dimulai lagi secara rekursif dari w. Jika tiba di simpul u dan tidak memenuhi kendala permasalahan, posisi simpul kembali ke simpul sebelumnya (w) dan memulai pencarian dengan membangkitkan simpul baru. Penelusuran berhenti jika semua simpul telah dikunjungi atau kendala terpenuhi [2].

Untuk mengoptimalkan pencarian kata, algoritma mengecek apakah susunan huruf menjadi prefiks dari kata lain pada kamus sebelum program memproses posisi baru pada papan dan menambahkan huruf lainnya. Hal ini memungkinkan program menghentikan pencarian *path* jika tidak ada kata dalam kamus yang sesuai susunan huruf saat itu. Algoritma ini memungkinkan pencarian yang tidak memakan waktu lama dan susunan kata yang didapat tercantum dalam kamus[1].

Penyusunan dilakukan secara DFS dengan membangkitkan simpul pada kata. Misalkan diberikan papan blok sebagai berikut :

	1	**2**	**3**	**4**
1	W	U	O	X
2	Y	E	R	R
3	N	P	M	O
4	U	T	S	H

Langkah pencarian :

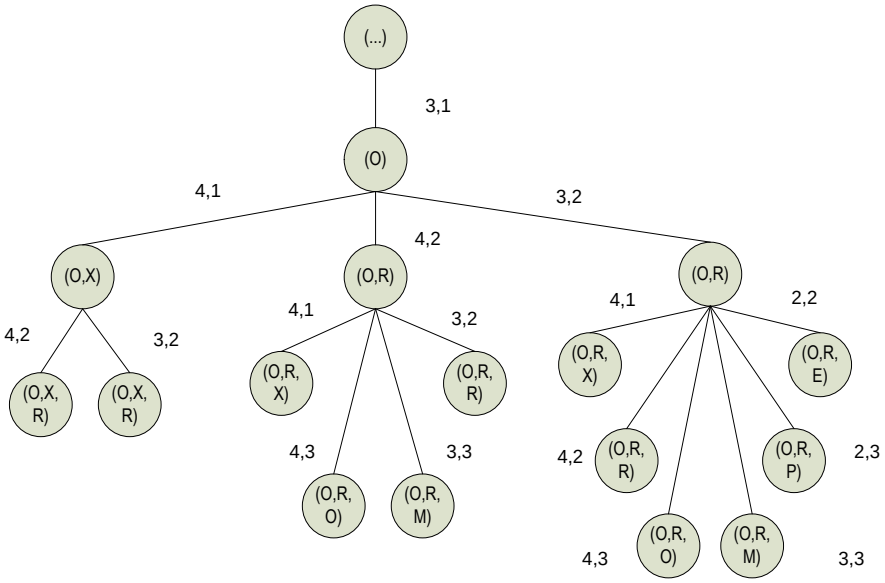
1 Program membangkitkan simpul 1,1 sebagai simpul awal, selanjutnya simpul sebelah kanan 1,2
2 Program memeriksa apakah “WU” merupakan prefiks dari salah satu kata dalam kamus. Jika tidak program tidak akan meneruskan simpul tersebut.
3 Program membangkitkan simpul 2,2, memeriksa apakah “WE” menjadi prefiks. Jika benar program akan membangkitkat simpul 1,2.
4 Untuk kasus ini baik “WEU”, “WEO”, “WER”, “WEM”,”WEP”,”WEN” dan “WY” tidak menjadi prefiks kata dalam kamus.
5 Program membangkitkan simpul 1,2 sebagai simpul awal. Dan seterusnya belum ditemukan prefiks yang tepat.
6 Selanjutnya program membangkitkan simpul 1,3 sebagai simpul awal. Hingga ditemukan prefiks “OR”.
7 Simpul selanjutnya yang akan dibangkitkan adalah 4,1 tapi “ORX” bukan prefiks. Program kembali ke simpul sebelumnya dan membangkitkan simpul 4,3 dan kembali mengecek prefiks. Jika tidak sesuai kembali kesimpul sebelumnya.
8 Hingga program membangkitkan simpul 2,2. “ORE” diperiksa apakah menjadi satu prefiks. Ternyata “ORE” menjadi prefiks sekaligus kata. Program mencatat susunan blok kemudian ditampilkan ke layar.

Gambar 2. Contoh pemilihan kata melalui DFS

Jika pemain kembali meminta petunjuk, program memulai pencarian dengan melanjutkan pencarian sebelumnya.

Gambar 3 Contoh pemilihan kata 2

Tinjau langkah pencarian dalam bentuk pohon ruang status berikut. Pohon yang ditampilkan hanya sebagian karena kenyataannya cukup banyak simpul yang dibangkitkan untuk kasus ini.



Gambar 4 Pohon Ruang Status DFS

4. KESIMPULAN

Proses pencarian dengan DFS mampu mereduksi kerja program saat mencari susunan huruf dari karakter acak pada Boggle. Proses tersebut juga dapat diaplikasikan pada komputer saat pemain manusia melawan komputer. Namun untuk menciptakan komputer yang mampu mengalahkan manusia dibutuhkan pengkajian lebih lanjut khususnya di bidang intelegensia buatan.

REFERENSI

[1] Elson, Jeremy , "The Boggle Solver", http:// www.circlemud.org.

[2] Munir, Rinaldi, "Diktat Kuliah IF2251-Strategi Algoritmik", Bandung, 2007.

[3] Hasbro Incorporated, Boggle, http :// www.boggle.com