PENGGUNAAN *DEPTH FIRST SEARCH* DALAM PENGAMBILAN KEPUTUSAN PADA KLIMAKS PERMAINAN SCRABBLE

Evan

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung
Jl. Ganesa 10, Bandung
e-mail: evangozali@yahoo.com

ABSTRAK

Permainan scrabble adalah permainan strategi yang cukup kompleks. Untuk membuat kecerdasan buatan yang dapat bermain scrabble, dibutuhkan banyak komputasi yang cukup rumit, seperti mencari kata dalam kamus, mencari tempat menaruh biji pada papan, dan mencari langkah yang menghasilkan nilai maksimum.
Makalah ini membahas salah satu aspek kecil dalam permainan scrabble, yaitu memaksimalkan nilai yang dapat diperoleh pada klimaks permainan dengan algoritma *depth first search*. Klimaks permainan adalah saat permainan akan berakhir dan pemain sudah tidak dapat mengambil biji scrabble yang baru.

Kata kunci: scrabble, klimaks, maksimasi, *depth first search*

1. PENDAHULUAN

Permainan scrabble merupakan salah satu permainan strategi yang cukup menarik. Permainan ini membutuhkan pengetahuan kosakata yang tinggi dan strategi yang matang. Pertama-tama penulis akan membahas sedikit tentang peraturan permainan scrabble.

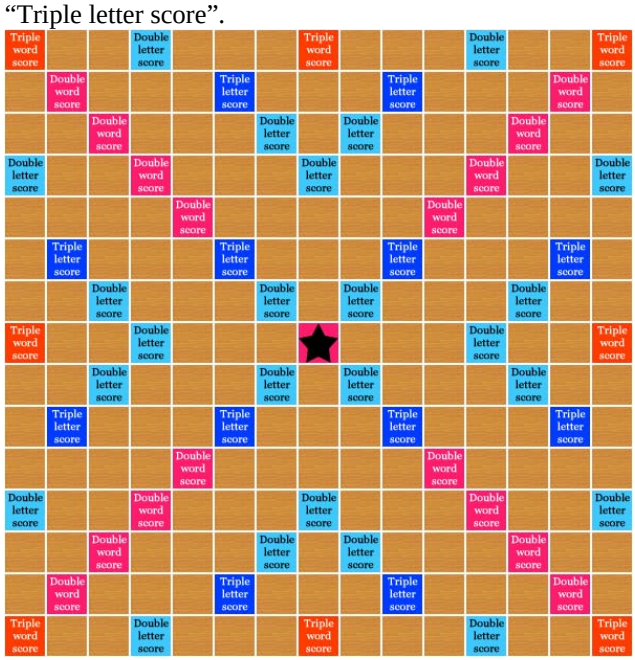
Permainan scrabble adalah permainan menyusun huruf-huruf alfabet pada sebuah papan sehingga membentuk kata-kata yang memiliki arti. Setiap huruf memiliki sebuah point yang menandakan nilai yang dapat diperoleh pemain jika menaruh huruf tersebut. Selain huruf alfabet, ada sebuah huruf khusus "blank" yang dapat menggantikan huruf apapun. Di bawah ini diberikan daftar point untuk setiap huruf pada scrabble berbahasa Inggris.

Tabel 1 Point untuk masing-masing huruf

Huruf	Nilai	Huruf	Nilai	Huruf	Nilai
Blank	0	I	1	R	1
A	1	J	8	S	1
B	3	K	5	T	1
C	3	L	1	U	1
D	2	M	3	V	4
E	1	N	1	W	4
F	4	O	1	X	8
G	2	P	3	Y	4
H	4	Q	10	Z	10

Permainan scrabble dimainkan oleh 2 sampai 4 orang. Di awal permainan, setiap pemain mengambil 7 buah biji huruf dari kantung biji. Lalu secara bergiliran setiap pemain menaruh huruf-huruf yang dimilikinya ke papan untuk memperoleh nilai. Setelah menaruh huruf, pemain kembali mengambil biji dari kantung biji sehingga jumlah biji yang dimilikinya ada 7 buah.

Nilai yang didapat pemain bergantung pada point huruf-huruf yang dikeluarkannya dan juga tempat menaruhnya. Gambar berikut menampilkan sebuah papan permainan scrabble. Dapat dilihat bahwa selain kotak kosong, ada juga kotak-kotak yang memiliki tulisan "Double letter score", "Triple letter score", "Double word score", dan "Triple letter score".



Gambar 1. Papan permainan scrabble

"Double letter score" berarti nilai huruf yang ditempatkan pada kotak tersebut akan menjadi 2 kali lipat, sedangkan "triple letter score" 3 kali lipat. "Double word score" berarti nilai seluruh kata menjadi 2 kali lipat, dan

"triple word score" berarti nilai seluruh kata menjadi 3 kali lipat. Jadi nilai yang dapat diperoleh pemain bergantung pada di mana dia meletakkan biji-bijinya.

Permainan akan berakhir jika kantung biji sudah kosong dan salah satu pemain telah memakai semua bijinya, atau semua pemain sudah tidak dapat mengeluarkan biji lagi. Setelah permainan berakhir, jumlah point biji-biji pemain yang tidak terpakai akan dikurangi dari nilai akhirnya. Pada permainan dengan selisih nilai yang ketat, pengurangan nilai tersebut dapat sangat menentukan pemenang permainan. Karena itu menjelang permainan berakhir, pemain perlu memikirkan strategi untuk memperoleh total nilai setinggi-tingginya dengan pengurangan nilai sesedikit mungkin. Klimaks permainan didefinisikan penulis sebagai saat kantung biji sudah kosong dan pemain sudah tidak dapat mengambil biji baru, jadi yang dapat dilakukan oleh pemain hanya menaruh biji ke papan.

2. MASALAH

2.1 Deskripsi Masalah

Misalkan biji-biji yang dimiliki pemain dinomori dengan nomor 1 sampai n. Point setiap biji dilambangkan dengan p[i]. Lalu ada sebuah tabel yang berisi nilai maksimal yang dapat diperoleh pemain jika menaruh biji-biji tertentu. Nilai tersebut bukan hanya dihitung dari jumlah point, tapi juga dari penempatan biji. Tabel tersebut diasumsikan telah digenerate melalui pencarian kata pada kamus dan juga pencarian posisi yang menghasilkan nilai maksimum.

Sebagai contoh, misalkan biji-biji yang dimiliki pemain adalah A, B, E, G, T, O, U. Tabel yang digenerate dapat berbentuk seperti berikut:

Tabel 2 Contoh tabel nilai

Himpunan huruf	Nilai maksimum
B, E, G	12
A, T	2
G, E, T	5
E, A, T	3
B, U, G	9
...	...

Karena nilai yang digenerate untuk setiap himpunan huruf adalah nilai maksimum yang mungkin diperoleh, himpunan huruf pada tabel unik sehingga tidak ada 2 entry tabel yang berisi himpunan huruf yang sama. Perhatikan juga bahwa entry tabel dapat berisi himpunan yang saling beririsan (misalnya {B, E, G} dan {G, E, T}) ataupun tidak beririsan (misalnya {E, A, T} dan {B, U, G}).

Untuk memaksimumkan nilai yang dapat diperoleh, pemain harus memilih himpunan-himpunan huruf yang memberikan nilai maksimum dan juga meminimumkan pengurangan nilai dari huruf yang tersisa. Secara formal, masalah dapat dideskripsikan sebagai memaksimumkan fungsi

$$f(H) = \sum_{\substack{A_i \subseteq H \\ A_i \cap A_j = \emptyset, i \neq j}} T(A_i) - \sum_{i \in H-(\cup A_i)} p[i]$$

Di mana H = himpunan biji-biji yang dimiliki pemain
f(H) = nilai maksimum yang dapat diperoleh pemain dengan menaruh biji-biji tertentu
A_i = himpunan bagian dari H yang saling lepas (tidak beririsan)
$T(A_i)$ = nilai maksimum yang dapat diperoleh jika menaruh himpunan A_i, didapat dari tabel nilai
p[i] = point biji-biji yang tidak ditaruh oleh pemain

2.2 Penyederhanaan Masalah

Adanya pengurangan nilai dari biji-biji yang tidak dikeluarkan pemain cukup mempersulit persoalan, karena pada persoalan maksimasi fungsi sebaiknya fungsi bersifat *non-decreasing*. Tapi fungsi di atas dapat dibuat menjadi fungsi *non-decreasing* sebagai berikut.

$$f(H) = \sum_{\substack{A_i \subseteq H \\ A_i \cap A_j = \emptyset, i \neq j}} T(A_i) - \sum_{i \in H-(\cup A_i)} p[i]$$

$$= \sum_{\substack{A_i \subseteq H \\ A_i \cap A_j = \emptyset, i \neq j}} T(A_i) + \left(\sum_{i \in (\cup A_i)} p[i] - \sum_{i \in (\cup A_i)} p[i] \right) - \sum_{i \in H-(\cup A_i)} p[i]$$

$$= \sum_{\substack{A_i \subseteq H \\ A_i \cap A_j = \emptyset, i \neq j}} T(A_i) + \sum_{i \in (\cup A_i)} p[i] - \left(\sum_{i \in (\cup A_i)} p[i] + \sum_{i \in H-(\cup A_i)} p[i] \right)$$

$$= \sum_{\substack{A_i \subseteq H \\ A_i \cap A_j = \emptyset, i \neq j}} \left(T(A_i) + \sum_{i \in A_i} p[i] \right) - \sum_{i \in H} p[i]$$

$$= \sum_{\substack{A_i \subseteq H \\ A_i \cap A_j = \emptyset, i \neq j}} \left(T'(A_i) \right) - P$$

di mana P = jumlah point seluruh biji yang dimiliki pemain, nilai ini selalu konstan
$T'(A_i)$ = nilai tabel yang telah diperbaharui dengan menambahkan point setiap biji pada nilai maksimum tabel.

Contoh tabel yang diperbaharui dari tabel 2:

Tabel 3 Tabel nilai yang diperbaharui

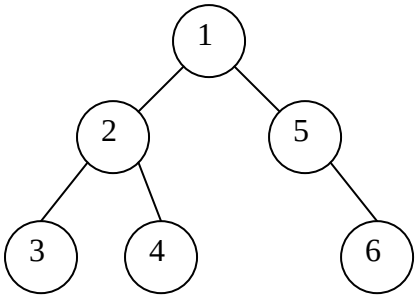
Himpunan huruf	Nilai maksimum	Jumlah point	Nilai maksimum baru
B, E, G	12	3+1+2 = 6	18
A, T	2	1+1 = 2	4
G, E, T	5	2+1+1 = 4	9
E, A, T	3	1+1+1 = 3	6
B, U, G	9	3+1+2 = 6	15
...	...	...	...

Dengan demikian fungsi f(H) telah diubah menjadi sebuah fungsi *non-decreasing* dengan cara memperbaharui tabel nilai.

3. METODE

3.1 Algoritma Depth First Search

Algoritma Depth First Search (DFS) adalah algoritma traversal di dalam graf dengan menelusuri simpul-simpul yang bertetangga terlebih dahulu. Khusus untuk persoalan ini, graf yang akan ditelusuri berbentuk pohon. Contoh penelusuran algoritma DFS diberikan pada gambar di bawah ini.



Gambar 2. Contoh penelusuran DFS

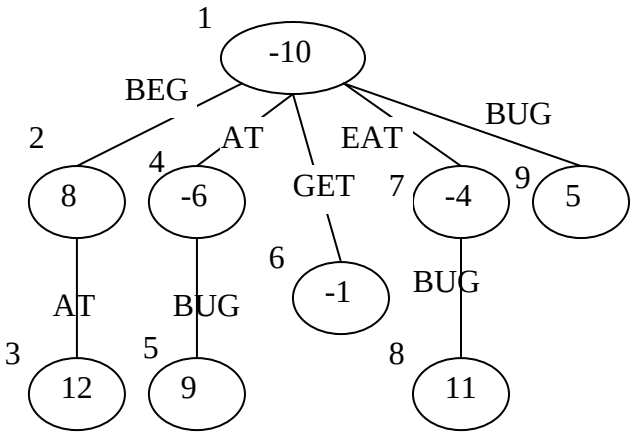
Pada pohon di atas, penelusuran dimulai dari simpul akar bernomor 1. Simpul berikutnya yang ditelusuri adalah simpul 2 yang bertetangga dengan simpul 1, lalu simpul 3 yang bertetangga dengan simpul 2. Karena simpul 3 sudah tidak memiliki tetangga, penelusuran akan berlanjut ke tetangga simpul 2 yaitu simpul 4. Setelah itu simpul 5 yang bertetangga dengan simpul 1, dan terakhir simpul 6 yang bertetangga dengan simpul 5.

3.2 Pencarian solusi

Untuk memecahkan persoalan memaksimalkan f(H), dilakukan penelusuran terhadap semua himpunan bagian A_i yang saling lepas. Setiap simpul berisi himpunan solusi biji-biji yang akan ditaruh dan jumlah nilai dari himpunan solusi. Penelusuran dimulai dari simpul akar yang merupakan himpunan kosong dengan nilai -P. Simpul-simpul berikutnya dibangkitkan dari himpunan biji pada tabel nilai yang saling lepas dengan himpunan solusi yang telah terbentuk sejauh ini. Jika penelusuran telah mencapai simpul daun dan tidak ada lagi simpul yang dapat dibangkitkan, nilai total dari himpunan solusi disimpan sebagai nilai maksimum sementara. Pencarian dilakukan sampai mendapatkan nilai maksimum yang paling besar.

Sebagai contoh digunakan huruf-huruf pada contoh sebelumnya yaitu A, B, E, G, T, O, U. Tabel nilai yang digunakan adalah tabel 3. Nilai simpul akar = -(1+3+1+2+1+1+1) = -10.

Mulai dari simpul akar, simpul pertama yang dibangkitkan berisi {B, E, G}. Lalu simpul berikutnya dibangkitkan dengan menambahkan {A, T} menjadi {B, E, G, A, T}. Setelah itu tidak ada lagi simpul yang dapat dibangkitkan, jadi nilai untuk solusi ini adalah 12.



Gambar 3. Contoh pencarian solusi

Dari simpul 2 juga sudah tidak ada lagi simpul yang dapat dibangkitkan karena {B, E, G} beririsan dengan {G, E, T}, {E, A, T}, maupun {B, U, G}. Jadi penelusuran akan kembali ke simpul akar. Dari simpul akar dibangkitkan simpul {A, T}, lalu {A, T, B, U, G} dengan nilai total 9. Penelusuran dilakukan terus sampai semua simpul dibangkitkan. Dari gambar di atas dapat dilihat bahwa nilai maksimum yang dapat diambil adalah 12 dengan menaruh biji {B, E, G} dan {A, T}.

Algoritma pencarian ini dapat dituliskan seperti berikut.

```
Function findmax(T: tabel, S: set of biji,
v: integer): integer
{T = tabel nilai yang telah diperbaharui
S = himpunan solusi sejauh ini
v = index tabel yang sedang dicari. v = 0
adalah awal pencarian
}
Deklarasi:
  point: integer
  m: integer
  i: integer
Algoritma:
  If v = 0 then
    {hitung nilai –P}
    point ← -totalpoint()
  else
    {ambil nilai tabel ke-v}
    point ← T[v].nilai
    {masukan himpunan di tabel ke S}
    S ← S ∪ T[v].himpunan
  Endif
  m ← 0
  For i ← v + 1 to T.length do
    If (S ∩ T[i].himpunan) = ∅ then
      m ← max(m, findmax(T,S,i))
    endif
  endfor
  → point + m
```

Algoritma di atas menggunakan skema rekursif di mana pada setiap langkah, point untuk simpul tersebut dihitung lalu ditambahkan dengan nilai maksimum dari simpul-simpul yang bertetangga dengan simpul tersebut.

4. KESIMPULAN

Algoritma DFS dapat digunakan untuk memecahkan masalah pengambilan keputusan pada klimaks permainan scrabble. Namun sebenarnya algoritma DFS kurang mangkus untuk pencarian nilai maksimum karena semua simpul yang mungkin harus dibangkitkan. Algoritma yang lebih baik untuk mencari nilai maksimum adalah algoritma branch and bound, namun untuk algoritma branch and bound perlu dicari fungsi pembatas yang dapat menjadi upper bound bagi nilai yang mungkin diperoleh. Tapi pada akhirnya, karena pada permainan scrabble jumlah biji maksimum hanya 7 sehingga jumlah simpul yang dapat dibangkitkan sedikit, penulis berpendapat bahwa algoritma DFS sudah cukup baik untuk menyelesaikan permasalahan ini.

REFERENSI

[1] Munir, Rinaldi. “Strategi Algoritmik”. Teknik Informatika ITB. 2005.