Penerapan Algoritma Backtracking dalam Pemecahan Boggle

Farhan Yusuf Akbar 13519202[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
[1]13519202@std.stei.itb.ac.id

***Abstract*—Algoritma Backtracking adalah algoritma yang muncul karna ketidakpuasan atas *exhaustive search* yang memiliki kompleksitas yang tinggi. Algoritma ini muncul seiring perkembangan *computer science*. Salah satu permasalahan yang dapat dipecahkan dengan Algoritma Backtracking adalah Boggle. Boggle adalah permainan menemukan kata-kata dalam urutan huruf yang berdekatan dalam sebuah kotak plastik berisi dadu-dadu berhuruf.**

***Keywords*—algoritma, backtracking, kompleksitas, Boggle**

I. PENDAHULUAN

Seiring perkembangan *computer science*, *engineer* berlomba-lomba untuk menemukan algoritma yang dapat memecahkan masalah dengan kompleksitas yang rendah. Algoritma adalah bentuk dari sebuah strategi. Strategi adalah langkah-langkah yang sistematis dalam menyelesaikan sebuah permasalahan. Algoritma dapat digunakan sebagai strategi pemecahan masalah di dunia nyata.

Perkembangan *computer science* membuat banyak algoritma baru ditemukan. Algoritma tersebut diantaranya adalah Algoritma Brute Force, Algoritma Greedy, Algoritma Divide and Conquer, Algoritma Decrease and Conquer, Algoritma BFS, Algoritma DFS, Algoritma IDS, Algoritma Backtracking, Algoritma Branch and Bound, Algoritma KMP, Algoritma Boyer Moore, Algoritma A* dan masih banyak lagi.

Setiap algoritma memiliki keunikannya masing-masing. Setiap algoritma dirancang untuk menyelesaikan permasalahan tertentu, seperti permasalahan optimasi atau pengambilan keputusan. Contoh permasalahan di dunia *computer science* yang dapat dipecahkan oleh algoritma-algoritma yang telah ditemukan adalah sebagai berikut: Travelling Salesperson Problem (TSP), Knapsack Problem, Integer Knapsack Problem, Graph Colouring, Routh Planning, String Matching, N-Queen Problem, Coin Exchange Problem, Huffman Code dan masih banyak lagi.

Algoritma dapat diklasifikasikan berdasarkan karakteristik tertentu. Berdasarkan kebutuhan waktu untuk menyelesaikan suatu permasalahan, algoritma dapat dibagi menjadi algoritma waktu-polinom dan algoritma waktu-non-polinom. Persoalan yang dapat diselesaikan dengan waktu-polinom disebut *tractable*, sedangkan persoalan yang dapat diselesaikan dengan waktu-non-polinom disebut *intractable*. Sebuah persoalan yang dapat diselesaikan dengan mesin Turing disebut permasalahan yang *solveable*, sedangkan yang tidak dapat diselesaikan dengan mesin Turing disebut permasalahan yang *unsolvable.* Permasalahan yang *solvable* dibagi menjadi *tractable* dan *intractable problem.*

Salah satu algoritma yang ditemukan adalah algoritma backtracking. Algoritma Backtracking adalah algoritma untuk memecahkan masalah secara rekursif dengan mencoba membangun solusi secara bertahap, satu per satu, dengan menghilangkan solusi yang gagal memenuhi suatu batasan di setiap tahap. Algoritma ini dirancang sebagai sebuah metode pemecahan masalah yang mangkus, terstruktur, dan sistematis, baik untuk persoalan optimasi maupun non-optimasi.

Boggle adalah salah satu permainan yang dapat diselesaikan dengan Algoritma Backtracking. Boggle adalah permainan menemukan kata-kata dalam urutan huruf yang berdekatan dalam sebuah kotak plastik berisi dadu-dadu berhuruf. Pada makalah ini akan dibahas cara penyelesaian Boggle dengan Algoritma Backtracking

II. KAJIAN TEORI

A. *Algoritma Backtracking*

Algoritma Backtracking adalah algoritma untuk memecahkan masalah secara rekursif dengan mencoba membangun solusi secara bertahap, satu per satu, dengan menghilangkan solusi yang gagal memenuhi suatu batasan di setiap tahap. Algoritma ini ditemukan pertama kali oleh matematikawan Amerika D. H. Lehmer pada 1950-an.

Alih-alih mencari semua kemungkinan solusi yang ada, Algoritma Backtracking memangkas kemungkinan yang tidak akan mengarah ke solusi. Oleh karena itu, Algoritma Backtracking disebut sebagai perbaikan dari *exhaustive search* pada Algoritma Brute Force. Pada Algoritma Backtracking hanya pilihan yang mengarah ke solusi yang dieksplorasi atau diekspan, pilihan yang tidak mengarah ke solusi tidak dipertimbangkan lagi.

Ada tiga tipe permasalahan yang dapat diselesaikan dengan Algoritma Backtracking:

1. Permasalahan pengambilan keputusan, pada permasalahan ini dicari solusi yang *feasible*.
2. Permasalahan optimasi, pada permasalahan ini dicari solusi paling optimal.

3. Permasalahan enumerasi, pada permasalahan ini dicari semua kemungkinan solusi yang *feasible*.

Algoritma Backtracking memiliki properti umum sebagai berikut:

1. Solusi persoalan.

   Solusi persoalan dinyatakan dengan vektor yang memiliki n-tuple:

   $$X = (x_1 , x_2 , \ldots, x_n), x_i \in S_i$$

2. Fungsi pembangkit.

   Fungsi pembangkit dinyatakan sebagai,

   $$T(x_{[1]}, x_{[2]}, \ldots, x_{[k-1]})$$

   yang akan membangkitkan nilai untuk xk, komponen dari vektor solusi.

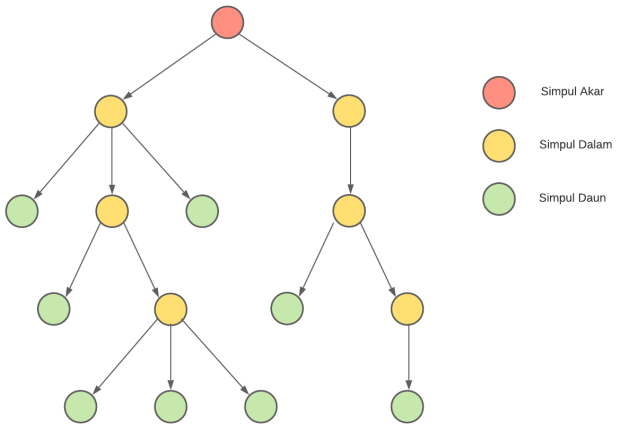
3. Fungsi pembatas.

   Fungsi pembatas atau *bounding function* dinyatakan sebagai,

   $$B(x_1, x_2, \ldots, x_k)$$

   Jika $(x_1, x_2, \ldots, x_k)$ pada $B(x_1, x_2, \ldots, x_k)$ tidak melanggar batasan (*constraint*) maka B(x1, x2, …,xk) mengembalikan nilai true dan $(x_1, x_2, \ldots,x_k)$ mengarah ke solusi. Sebaliknya jika $(x_1, x_2, \ldots, x_k)$ pada $B(x_1, x_2, \ldots, x_k)$ melanggar batasan (*constraint*) maka $B(x_1, x_2, \ldots, x_k)$ mengembalikan nilai false dan $(x_1, x_2, \ldots, x_k)$ tidak diekspan atau dipertimbangkan lagi.

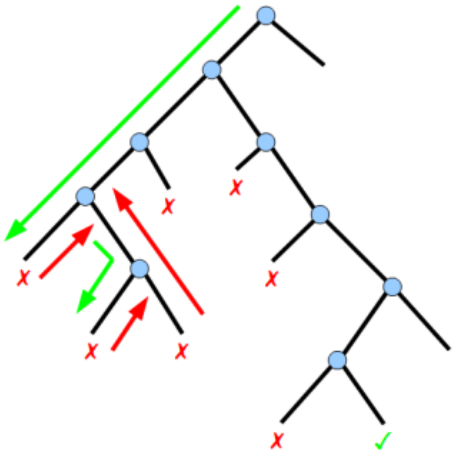
Semua kemungkinan solusi yang didapat dari sebuah persoalan dikumpulkan menjadi sebuah ruang solusi. Ruang solusi dapat direpresentasikan dalam struktur pohon. Setiap simpul pohon menyatakan status (*state*) persoalan. Lintasan dari akar ke daun menyatakan solusi yang mungkin. Seluruh lintasan dari akar ke daun membentuk ruang solusi. Pengorganisasian pohon ruang solusi diacu sebagai pohon ruang status.



Gambar 1. Pohon Ruang Status (Sumber: Dokumen Pribadi)

Langkah-langkah pencarian solusi dengan Algoritma Backtracking adalah sebagai berikut:

1. Solusi dicari dengan membangkitkan simpul-simpul status sehingga menghasilkan lintasan dari akar ke daun. Aturan pembangkitan simpul sama dengan Algoritma DFS (Depth First Search).
2. Simpul-simpul yang sudah dibangkitkan disebut simpul hidup. Simpul yang sedang diperluas disebut simpul ekspan atau simpul-E.
3. Jika simpul-E tidak mengarah ke solusi maka simpul-E dimatikan. Simpul-E dinyatakan tidak mengarah ke solusi dengan menggunakan fungsi pembatas
4. Jika pembentukan lintasan berakhir dengan simpul mati, maka proses pencarian *backtrack* ke simpul sebelumnya.
5. Pencarian diteruskan dengan simpul anak lain dari simpul hasil *backtrack*.
6. Pencarian selesai jika telah sampai di simpul daun atau *goal node.*



Gambar 2. Langkah-Langkah Pencarian Solusi dengan Algoritma Backtracking pada Pohon Ruang Status (Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-backtracking-2021-Bagian1.pdf)

B. Boggle

Gambar 3. Boggle (Sumber: https://en.wikipedia.org/wiki/Boggle)

Boggle adalah permainan kata yang ditemukan oleh Allan Turrof dan didistribusikan oleh Parker Brothers pada tahun 1972. Permainan ini dimainkan menggunakan kotak plastik yang terdiri dari dadu-dadu berhuruf. Pemain harus menemukan kata-kata dalam susunan huruf yang berdekatan, baik secara horizontal, vertikal maupun diagonal.

Permainan dimulai dengan mengguncang kotak Boggle yang berisi 16 dadu, masing-masing dengan huruf berbeda tercetak di setiap sisinya. Dadu tersusun pada kotak Boggle dengan sususan 4 × 4. Pada setiap dadu, hanya huruf bagian atas yang terlihat. Setelah diguncangkan, *timer* akan diset selama tiga menit dan permainan dimulai.

Pemain mencari kata-kata yang dapat dibuat dari huruf-huruf pada dadu yang berdekatan secara berurut, di mana dadu yang berdekatan adalah dadu yang bertetangga secara horizontal, vertikal, atau diagonal. Kata yang dicari setidaknya terdiri dari tiga huruf dan tidak boleh menggunakan dadu huruf yang sama lebih dari satu kali per kata. Setiap pemain mencatat semua kata yang mereka temukan dengan menulis di selembar kertas. Setelah tiga menit berlalu, semua pemain harus segera berhenti menulis dan permainan memasuki fase penilaian.

F	R	O	O
Y	I	E	S
L	D	N	T
A	E	R	E

Gambar 4. Contoh Susunan Huruf pada Boggle (Sumber: Dokumen Pribadi)

Dalam fase penilaian, setiap pemain membacakan daftar kata-kata yang ditemukan. Jika dua atau lebih pemain menulis kata yang sama, kata itu akan dihapus dari semua daftar pemain. Setiap pemain dapat menantang validitas sebuah kata, dalam hal ini kamus yang sebelumnya ditunjuk digunakan untuk memverifikasi atau membantahnya. Untuk semua kata yang tersisa setelah duplikat dihilangkan, poin diberikan berdasarkan panjang kata. Pemenangnya adalah pemain yang total poinnya tertinggi. Poin yang didapat dihitung dari tabel poin berikut.

TABLE 1. POIN YANG DIPEROLEH BERDASARKAN PANJANG KATA

Panjang Kata	Poin
3-4	1
5	2
6	3
7	5
8	11

Pada Boggle satu dadu memiliki huruf "Qu". Hal ini karena huruf Q hampir selalu diikuti oleh huruf U dalam kata-kata berbahasa Inggris. Jika ada huruf Q di Boggle, akan sulit untuk digunakan jika huruf U, secara kebetulan, tidak muncul di sebelahnya. Oleh karena itu Qu pada boggle dihitung sebagai dua huruf: Qu pada SQUID akan dihitung dua huruf.

Contoh kata-kata yang dapat ditemukan pada susunan huruf seperti Gambar 4 adalah sebagai berikut.

F_1	R_2	O	O
Y	I_3	E_4	S
L	D_6	N_5	T
A	E	R	E

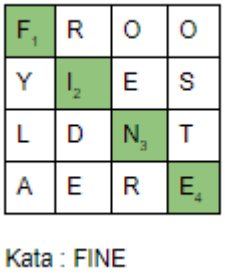
Kata : FRIEND

Gambar 5. Contoh Kata yang Ditemukan pada Boggle 1 (Sumber : Dokumen Pribadi)

F	R_1	O_2	O
Y	I	E	S_3
L	D	N	T_4
A	E	R_6	E_5

Kata : ROSTER

Gambar 6. Contoh Kata yang Ditemukan pada Boggle 2 (Sumber : Dokumen Pribadi)



Gambar 7. Contoh Kata yang Ditemukan pada Boggle 3 (Sumber : Dokumen Pribadi)

Gambar 8. Contoh Kata yang Ditemukan pada Boggle 4 (Sumber : Dokumen Pribadi)

Gambar 9. Contoh Kata yang Ditemukan pada Boggle 5 (Sumber : Dokumen Pribadi)

III. Penyelesaian Boggle dengan Algoritma Backtracking

Sebelum melakukan langkah penyelesaian Boggle dengan Algoritma Backtracking, terlebih dahulu dipetakan komponen pada Boggle ke dalam properti Algoritma Backtracking. Pemetaan tersebut dapat dilihat sebagai berikut.

1. Solusi Persoalan

   Solusi persoalan pada Boggle adalah urutan huruf-huruf yang didaftarkan ($X=(x_1,x_2,..x_k)$, dengan xi adalah huruf yang didaftarkan pada tahap ke-i).

2. Fungsi Pembangkit

   Fungsi pembangkit $T(x_{[1]}, x_{[2]}, \ldots, x_{[k-1]})$ pada Boggle adalah x_k dimana x_k adalah huruf yang didaftarkan pada tahap ke k.

3. Fungsi Pembatas

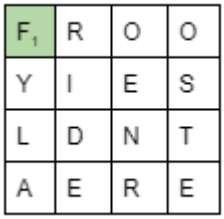
   Fungsi pembatas $B(x_1, x_2, \ldots, x_k)$ pada Boggle adalah apakah huruf-huruf pada $(x_1, x_2, \ldots, x_k)$ merupakan sebuah kata atau prefix yang terdaftar dalam sebuah kamus resmi. Pada makalah ini kamus yang digunakan adalah kamus Merriam-Webster yang dapat diakses pada https://www.merriam-webster.com.

Pada setiap tahap pemecahan Boggle dengan Algoritma Backtracking memiliki 4 kemungkinan berikut.

1. Huruf-huruf yang telah didaftarkan tidak membentuk kata dan tidak membentuk prefix. Langkah yang dilakukan adalah *backtrack* ke huruf sebelumnya.
2. Huruf-huruf yang telah didaftarkan tidak membentuk kata tetapi membentuk prefix. Langkah yang dilakukan adalah meng-*explore* huruf selanjutnya.
3. Huruf-huruf yang telah didaftarkan tidak membentuk prefix tetapi membentuk kata. Langkah yang dilakukan adalah menyimpan kata tersebut dan *backtrack* ke huruf sebelumnya untuk mencari kata yang baru.
4. Huruf-huruf yang telah didaftarkan membentuk kata dan membentuk prefix. Langkah yang dilakukan adalah menyimpan kata tersebut dan terus meng-*explore* huruf selanjutnya untuk mencari kemungkinan kata yang baru.

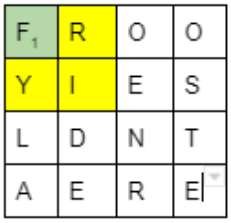
Sebagai contoh kasus, akan digunakan susunan huruf yang sama seperti gambar 4. Langkah-langkah penyelesaian Boggle dengan kasus tersebut adalah sebagai berikut.

1. Pilih satu huruf untuk dijadikan simpul akar. Penentuan simpul akar dibebaskan, namun untuk keteraturan penyelesaian, huruf di pojok kiri atas akan dipilih sebagai simpul akar pertama.



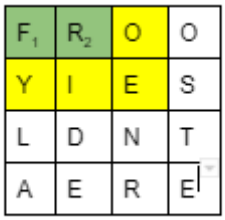
Gambar 10. Langkah 1 Pilih F Sebagai Simpul Akar (Sumber : Dokumen Pribadi)

2. Simpul akar (F) diekspan, sehingga simpul huruf R, I, dan Y aktif.



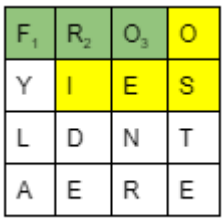
Gambar 11. Langkah 2 Simpul R,I,Y Aktif (Sumber : Dokumen Pribadi)

3. Pilih simpul R untuk diekspan. B(F,R) bernilai *true* sehingga simpul R dapat diekspan. Simpul O, E, I, Y aktif.



Gambar 12. Langkah 3 Simpul R Diekspan, Simpul O,E,I,Y Aktif (Sumber : Dokumen Pribadi)

4. Pilih simpul O untuk dieskpan. B(F,R,O) bernilai *true* sehingga simpul O dapat diekspan. Simpul O,S,E,I aktif

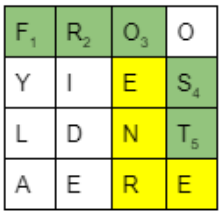


Gambar 13. Langkah 4 Simpul O Diekspan, Simpul O,S,E,I Aktif (Sumber : Dokumen Pribadi)

5. Pilih simpul O untuk disimpan. B(F,R,O,O) bernilai *false* sehingga simpul O tidak dapat diekspan dan dan dimatikan. Pilih simpul S untuk disimpan. B(F,R,O,S) bernilai *true* sehingga dapat diekspan. Simpul O,E,N,T aktif.

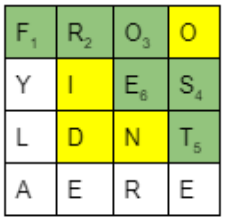
Gambar 14. Langkah 5 Simpul S Diekspan, Simpul O,E,N,T Aktif (Sumber : Dokumen Pribadi)

6. Pilih simpul O untuk diekspan. B(F,R,O,S,O) bernilai *false* sehingga simpul O tidak dapat diekspan. Pilih simpul E untuk diekspan. B(F,R,O,S,E) bernilai *false* sehingga simpul E tidak dapat diekspan. Pilih simpul N untuk diekspan. B(F,R,O,S,N) bernilai *false* sehingga simpul N tidak dapat diekspan. Pilih simpul T untuk diekspan. B(F,R,O,S,T) berniali *true* sehingga simpul T dapat diekspan. FROST merupakan sebuah kata sekaligus prefix sehingga kata FROST disimpan dan terus dieksplor karena sekaligus merupakan prefix.



Gambar 15. Langkah 6 Simpul T Diekspan, Simpul E,N,R,E Aktif, Kata FROST Ditemukan (Sumber : Dokumen Pribadi)

7. Pilih simpul E untuk diekspan. B(F,R,O,S,T,E) bernilai *true* sehingga simpul E dapat diekspan. Simpul O,I,D,N aktif.



Gambar 16. Langkah 7 Simpul E Diekspan, Simpul O,I,D,N Aktif. (Sumber : Dokumen Pribadi)

8. Pilih simpul O untuk diekspan. B(F,R,O,S,T,E,O) bernilai *false* sehingga simpul O tidak dapat diekspan. Pilih simpul I untuk diekspan. B(F,R,O,S,T,E,I) bernilai *false* sehingga simpul I tidak dapat diekspan.

Pilih simpul D untuk diekspan. B(F,R,O,S,T,E,D) bernilai *true* sehingga simpul D dapat diekspan. FROSTED merupakan sebuah kata dan bukan prefix sehingga kata FROSTED disimpan dan dilakukan *backtrack* ke langkah 7 dengan simpul yang diproses adalah simpul N.

Gambar 17. Langkah 8 Simpul D Diekspan, Kata FROSTED Ditemukan (Sumber : Dokumen Pribadi)

9. Setelah *backtrack* ke langkah 7, simpul N diperiksa. Ternyata B(F,R,O,S,T,E,N) bernilai *false* sehingga tidak dapat diekspan. Karena tidak ada simpul yang tersisa yang bisa diproses pada langkah 7, maka dilakukan *backtrack* lagi ke langkah 6 seperti pada Gambar 15. Simpul yang akan diproses adalah simpul yang sebelumnya belum dievaluasi dengan fungsi pembatas. Langkah ini diulang-ulang sampai hasil *bactrack* kembali ke simpul akar F. Untuk saat ini, kata yang telah ditemukan ada dua, yaitu FROST dan FROSTED. Namun masih banyak lagi kemungkinan kata lainnya yang mungkin ditemukan.
10. Setelah hasil *backtrack* kembali ke simpul akar F maka simpul akar diganti. Simpul akar selanjutnya yang menjadi simpul akar adalah simpul R. Maka simpul R diekspan seperti langkah-langkah sebelumnya sebagaimana langkah yang sama dilakukan pada simpul F. Begitu simpul akar R selesai diproses, simpul akar diganti lagi menjadi simpul lain sampai semua huruf pada Boggle telah diproses. Akhirnya, didapat semua kata yang mungkin ditemukan dari Boggle.

Untuk penyelesaian Boggle 4x4 memang dibutuhkan waktu yang cukup banyak untuk memeriksa kemungkinan-kemungkinan yang tidak dilanggar oleh fungsi pembatas. Tujuan penggunaan Algoritma Backtracking pada permainan Boggle ini bukan untuk mencari kata dengan cepat, melainkan untuk mencari semua kemungkinan kata yang ada. Penggunaan Algoritma Backtracking dalam pemecahan Boggle ini jauh lebih baik dibandingkan dengan mencari semua kemungkinan kombinasi huruf yang bisa dibuat. Kompleksitasnya akan sangat jauh lebih tinggi dibanding Algoritma Backtracking.

IV. KESIMPULAN

Algoritma Backtracking dapat dengan baik memecahkan permasalahan permainan Boggle. Algoritma ini dapat memangkas kemungkinan-kemungkinan yang melanggar fungsi pembatas sehingga lebih baik dibandingkan dengan memeriksa semua kemungkinan kombinasi huruf yang ada. Dapat disimpulkan Algoritma Backtracking dapat memecahkan permainan Boggle.

REFERENCES

[1] Munir, Rinaldi. Algoritma Runut-Balik (Backtracking). https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-backtracking-2021-Bagian1.pdf. Diakses pada tanggal 10 Mei 2021

[2] Geeksforgeeks.org. Backtracking Introduction. https://www.geeksforgeeks.org/backtracking-introduction/. Diakses pada tanggal 11 Mei 2021

[3] Murphy, Mary Beth.. 1978. Toy Designer's Creations, Bill Cooke, 1974 Boggle the Mind.

[4] Merriam-Webster. https://www.merriam-webster.com/dictionary. Diakses pada 11 Mei 2021

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.

Padang, 11 Mei 2021

Farhan Yusuf Akbar
13519202