

Penerapan Algoritma DFS dan BFS untuk Permainan Wordsearch Puzzle

Stefan Lauren / 13510034¹

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia

¹13510034@std.stei.itb.ac.id

Abstract—Banyak strategi algoritma yang diciptakan untuk mempermudah penyelesaian masalah. Pada makalah ini strategi algoritma yang dipilih adalah *DFS*(*Depth First Search*) dan *BFS*(*Breadth First Search*). Kedua strategi algoritma ini akan diimplementasikan terhadap sebuah permainan berjenis puzzle yang bernama wordsearch. Tujuan dari makalah ini adalah untuk membandingkan langkah kerja yang dilakukan masing-masing antara *DFS* dan *BFS* untuk menyelesaikan permainan wordsearch tersebut.

Index Terms—*BFS*, *DFS*, puzzle, Wordsearch.

I. PENDAHULUAN

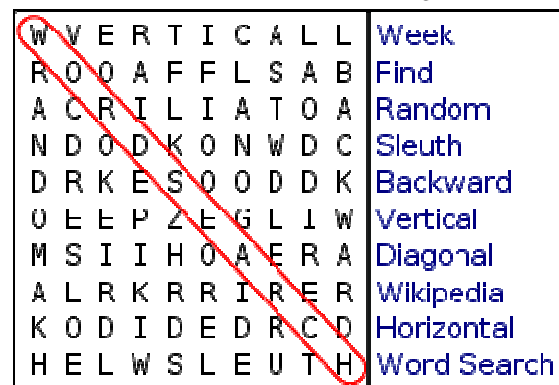
Perkembangan industri permainan meningkat dengan pesat setiap tahunnya. Berbagai macam jenis permainan bermunculan dari jenis permainan klasik sampai gabungan dari jenis-jenis permainan lainnya. Salah satu jenis permainan yang cukup digemari oleh banyak orang di segala usia adalah puzzle. Setiap permainan berjenis puzzle ini membutuhkan waktu dan pikiran untuk menyelesaikannya. Hal inilah yang menjadi tantangan dan daya tarik dari jenis permainan ini.

Daya tarik lain dari permainan ini adalah terdapat wujud nyata dari permainan. Salah satu contoh permainan puzzle adalah teka-teki silang(TTS) yang dengan mudah diwujudkan secara nyata biasanya melalui media cetak. Banyak permainan berjenis puzzle yang dapat dibuat bentuk nyatanya seperti wordsearch puzzle, tic-tac-toe, sudoku, kakuro, dan lain sebagainya.

Seiring perkembangan permainan tersebut, banyak orang yang tertarik untuk dapat menyelesaikan permainan tersebut dengan cepat. Pola pikir untuk menyelesaikan setiap permainan berjenis puzzle ini biasanya unik untuk setiap permainan. Oleh sebab itu pola penyelesaian dapat direpresentasikan menggunakan banyak strategi algoritma. Pada permainan wordsearch puzzle akan digunakan dua buah strategi algoritma yang terkenal yaitu *DFS* dan *BFS*. Kedua algoritma ini diharapkan dapat memberikan solusi yang sama meskipun dengan pola pikir algoritma yang cukup berbeda.

II. PERMAINAN WORDSEARCH

Wordsearch puzzle adalah sebuah permainan kata dengan banyak huruf diletakkan pada sebuah papan yang biasanya memiliki bentuk persegi atau persegi panjang. Tujuan dari permainan puzzle ini adalah untuk menemukan dan menandai semua kata-kata yang tersembunyi di dalam papan. Kata-kata tersebut dapat tersusun secara horizontal, vertikal, atau diagonal.



(Sumber: en.wikipedia.org)

Biasanya terdapat daftar kata tersembunyi yang harus dicari. Tetapi terdapat variasi permainan wordsearch yang tidak memberikan daftar kata sehingga lebih menantang bagi pemain untuk menemukan semua kata-kata yang ada. Variasi lainnya dari permainan ini adalah adanya tema untuk kata-kata maupun tampilan yang sesuai dengan temanya. Permainan ini biasanya ditemukan pada koran harian dan buku puzzle. Bahkan wordsearch dapat dijadikan sebagai salah satu metoda pengajaran untuk anak-anak sebagai sarana untuk mengingat arti kata, pengejaan kata, dan rekreasi.

Sejarahnya adalah permainan wordsearch ini dirancang oleh Norman E. Gibat dan dimunculkan pada sebuah majalah di tahun 1968 pada kota Norman, Oklahoma. Permainan puzzle ini langsung digemari oleh banyak orang dan menjadi terkenal. Penyebarannya sangat pesat terutama didukung oleh para guru yang memberikan salinannya ke sekolah-sekolah terdekatnya. Perkembangannya setelah itu adalah banyak orang yang berusaha menirukan permainan wordsearch yang asli

dengan sedikit modifikasi.



See how many of these newspaper related words you can find in the puzzle.
The words can be forward, backward or diagonal.

- | | | | |
|------------|---------------|------------|-------------|
| 1. SPRING | 5. SEASON | 9. EQUATOR | 13. FLOWERS |
| 2. VERNAL | 6. EQUAL | 10. SUN | 14. PLANTS |
| 3. EQUINOX | 7. NORTHERN | 11. EARTH | 15. GARDEN |
| 4. MARCH | 8. HEMISPHERE | 12. AXIS | 16. TULIPS |

(Sumber: nierocks.areavoices.com)

Strategi penyelesaian permainan ini yang umum adalah dengan membaca huruf satu per satu dari kiri ke kanan dan mencari huruf pertama dari kata yang dicari. Setelah mendapatkan huruf yang dimaksud, pemain dapat mencari huruf kedua dengan melihat ke delapan arah yang mengelilingi huruf pertama tersebut. Metoda ini terus dilakukan sampai seluruh huruf pada kata ditemukan.

Strategi lainnya adalah dengan mencari huruf-huruf unik seperti Q, O, U, X, Z jika ada pada kata yang dicari. Biasanya kelima huruf ini akan langsung terlihat pada papan permainan. Strategi lainnya adalah dengan langsung mencari dua huruf pertama pada kata yang dicari. Strategi ini cukup efektif karena lebih mudah mencari dua huruf yang berdampingan.

Jika daftar kata tidak diberikan maka salah satu cara penyelesaiannya adalah dengan mencari kata per baris dengan variasi horizontal, vertikal, dan diagonal secara bergantian. Biasanya puzzle memberikan kemudahan kepada pemain dengan adanya pola penempatan kata yang mirip.

Wordsearch juga dapat digunakan untuk menyampaikan pesan tersembunyi. Cara pertama adalah dengan menggabungkan kata-kata yang ditulis secara terbalik (kanan ke kiri). Cara lainnya adalah dengan menggabungkan seluruh huruf yang tidak digunakan oleh kata apapun dari papan permainan tersebut. Pesan juga dapat disampaikan dengan tidak adanya kata-kata pada pesan yang ingin disampaikan pada daftar kata yang harus dicari. Ada juga variasi permainan ini yang memperbolehkan kata yang dicari dapat disusun dengan adanya belokan sebesar 90° pada huruf apapun.

III. ALGORITMA DFS

Misalkan kita mempunyai graf G yang mempunyai n buah simpul. Kita akan melakukan traversal di dalam graf, dan misalkan traversal dimulai dari simpul v . Algoritma *DFS* adalah sebagai berikut: Misalkan penelusuran dimulai dari simpul v . Simpul berikutnya yang dikunjungi adalah simpul w yang bertetangga dengan simpul v , lalu pencarian mendalam dimulai lagi secara rekursif dari simpul w . Ketika mencapai simpul u sedemikian sehingga semua simpul yang bertetangga dengannya telah dikunjungi, pencarian dirunut-balik ke simpul terakhir yang dikunjungi sebelumnya dan mempunyai simpul w yang belum dikunjungi. Pencarian mendalam dimulai lagi dari w . Pencarian berakhir bila tidak ada lagi simpul yang belum dikunjungi yang dapat dicapai dari simpul yang telah dikunjungi.

Algoritma *DFS* yang lengkap adalah sebagai berikut:

procedure *DFS* (*input* v : *integer*)

{*Mengunjungi seluruh simpul graf dengan algoritma pencarian DFS.*

Masukan: v adalah simpul awal kunjungan.

Keluaran: semua simpul yang dikunjungi ditulis ke layar. }

Deklarasi

w : *integer*

Algoritma:

write(v)

dikunjungi[v] $\leftarrow$ *true*

for $w \leftarrow 1$ **to** n **do**

if $A[v, w] = 1$ **then** {*simpul v dan w bertetangga*}

if not *dikunjungi*[w] **then**
DFS(w)

endif

endif

endfor

IV. ALGORITMA BFS

Misalkan kita mempunyai graf G yang mempunyai n buah simpul. Kita akan melakukan traversal di dalam graf, dan misalkan traversal dimulai dari simpul v . Algoritma *BFS* adalah sebagai berikut: Kunjungi simpul v , kemudian semua simpul yang bertetangga dengan simpul v dikunjungi terlebih dahulu. Selanjutnya, simpul yang belum dikunjungi dan bertetangga dengan simpul-simpul tadi dikunjungi, demikian seterusnya. Jika graf berbentuk pohon berakar, maka semua simpul pada aras d dikunjungi lebih dahulu sebelum simpul-simpul pada aras $d + 1$.

Algoritma *BFS* yang lengkap adalah sebagai berikut:

procedure *BFS* (*input* v : *integer*)

{*Traversal graf dengan algoritma pencarian BFS.*

Masukkan: v adalah simpul awal kunjungan.

Keluaran: semua simpul yang dikunjungi dicetak ke layar.}

Deklarasi:

w: integer

q : antrian;

procedure BuatAntrian (input/output q : antrian)
{membuat antrian kosong, kepala(q) diisi 0}

procedure MasukAntrian (input/output q : antrian,
input v : integer)
{memasukkan v ke dalam antrian q pada posisi belakang}

procedure HapusAntrian (input/output q : antrian,
output v : integer)
{menghapus v dari kepala antrian q}

function AntrianKosong (input q : antrian) → boolean
{true jika antrian q kosong, false jika sebaliknya}

Algoritma

BuatAntrian(q); {buat antrian kosong}

write(v) {cetak simpul awal yang dikunjungi}
dikunjungi[v] ← true {simpul v telah dikunjungi,
tandai dengan true}
MasukAntrian(q,v) {masukkan simpul awal
kunjungan ke dalam antrian}

{kunjungi semua simpul graf selama antrian belum
kosong}

while not AntrianKosong(q) do

HapusAntrian(q,v) {simpul v telah dikunjungi,
hapus dari antrian}

for w ← 1 to n do

if A[v,w] = 1 then {v dan w bertetangga}

if not dikunjungi[w] then

write(w) {cetak simpul yang
dikunjungi}

MasukAntrian(q,w)

dikunjungi[w] ← true

endif

endif

endfor

endwhile

{AntrianKosong(q)}

V. PENYELESAIAN WORDSEARCH DENGAN DFS

Penyelesaian dengan DFS akan menggunakan contoh kasus. Pada penyelesaian ini, prioritas arah secara berurutan adalah kiri, kiri-atas, atas, kanan-atas, kanan, kanan-bawah, bawah, dan kiri-bawah. Asumsi lainnya adalah pencarian pada arah yang bersangkutan akan dihentikan ketika huruf sudah tidak sama lagi. Kata yang

akan dicari adalah “SIP”. Pencarian dimulai dari kotak paling kiri dan paling atas.

Papan awal:

A	S	N	C	W
M	T	I	O	R
H	U	C	P	F
K	A	S	B	E
Q	D	J	G	O

Langkah 1:

Kotak pertama berisi “A” tidak sesuai dengan huruf pertama pada kata yang dicari “S”. Pemeriksaan selesai, dilanjutkan ke kotak dengan prioritas tertinggi yaitu kotak di sebelah kanan.

A	S	N	C	W
M	T	I	O	R
H	U	C	P	F
K	A	S	B	E
Q	D	J	G	O

Langkah 2:

Kotak kedua berisi “S” yang sesuai dengan huruf pada pencarian kata pertama. Kotak ini akan disimpan untuk sementara dan pencarian berlanjut

A	S	N	C	W
M	T	I	O	R
H	U	C	P	F
K	A	S	B	E
Q	D	J	G	O

Langkah 3:

“A” tidak sama dengan “T”, pencarian dilanjutkan.

A	S	N	C	W
M	T	I	O	R
H	U	C	P	F
K	A	S	B	E
Q	D	J	G	O

Langkah 4:

“N” tidak sama dengan “T”, pencarian dilanjutkan.

A	S	N	C	W
M	T	I	O	R
H	U	C	P	F
K	A	S	B	E
Q	D	J	G	O

Langkah 5:

“T” sama dengan “T” untuk kata yang dicari. Kotak ini akan disimpan dan pencarian dilanjutkan.

A	S	N	C	W
M	T	I	O	R
H	U	C	P	F
K	A	S	B	E
Q	D	J	G	O

Langkah 6:

“T” tidak sama dengan “P”, pencarian dilanjutkan.

A	S	N	C	W
M	T	I	O	R
H	U	C	P	F
K	A	S	B	E
Q	D	J	G	O

Langkah 7:

“N” tidak sama dengan “P”, pencarian dilanjutkan.

A	S	N	C	W
M	T	I	O	R
H	U	C	P	F
K	A	S	B	E
Q	D	J	G	O

Langkah 8:

“C” tidak sama dengan “P”, pencarian dilanjutkan.

A	S	N	C	W
M	T	I	O	R
H	U	C	P	F
K	A	S	B	E
Q	D	J	G	O

Langkah 9:

“O” tidak sama dengan “P”, pencarian dilanjutkan.

A	S	N	C	W
M	T	I	O	R
H	U	C	P	F
K	A	S	B	E
Q	D	J	G	O

Langkah 10:

“P” sama dengan “P” pada kata yang dicari. Seluruh huruf pada kata sudah ditemukan, pencarian dihentikan.

A	S	N	C	W
M	T	I	O	R
H	U	C	P	F
K	A	S	B	E
Q	D	J	G	O

Pada proses pencarian, DFS membutuhkan 10 langkah perbandingan.

VI. PENYELESAIAN WORDSEARCH DENGAN BFS

Penyelesaian dengan BFS akan menggunakan contoh kasus yang sama dengan DFS.

Papan awal:

A	S	N	C	W
M	T	I	O	R
H	U	C	P	F
K	A	S	B	E
Q	D	J	G	O

Langkah 1:

“A” tidak sama dengan “S”, pencarian dilanjutkan.

A	S	N	C	W
M	T	I	O	R
H	U	C	P	F
K	A	S	B	E
Q	D	J	G	O

Langkah 2:

“S” sama dengan “S” yang dicari, kotak disimpan dan pencarian dilanjutkan.

A	S	N	C	W
M	T	I	O	R
H	U	C	P	F
K	A	S	B	E
Q	D	J	G	O

Langkah 3:

“A” tidak sama dengan “T”, pencarian dilanjutkan.

A	S	N	C	W
M	T	I	O	R
H	U	C	P	F
K	A	S	B	E
Q	D	J	G	O

A	S	N	C	W
M	T	I	O	R
H	U	C	P	F
K	A	S	B	E
Q	D	J	G	O

Langkah 4:
 “N” tidak sama dengan “I”, pencarian dilanjutkan.

A	S	N	C	W
M	T	I	O	R
H	U	C	P	F
K	A	S	B	E
Q	D	J	G	O

Langkah 7:
 “M” tidak sama dengan “I”, pencarian dilanjutkan.

A	S	N	C	W
M	T	I	O	R
H	U	C	P	F
K	A	S	B	E
Q	D	J	G	O

Langkah 5:
 “I” sama dengan “I” pada kata yang dicari, kotak disimpan dan pencarian dilanjutkan.

A	S	N	C	W
M	T	I	O	R
H	U	C	P	F
K	A	S	B	E
Q	D	J	G	O

Langkah 8:
 “T” tidak sama dengan “P”, pencarian dilanjutkan.

A	S	N	C	W
M	T	I	O	R
H	U	C	P	F
K	A	S	B	E
Q	D	J	G	O

Langkah 6:
 “T” tidak sama dengan “I”, pencarian dilanjutkan.

Langkah 9:
 “N” tidak sama dengan “P”, pencarian dilanjutkan.

A	S	N	C	W
M	T	I	O	R
H	U	C	P	F
K	A	S	B	E
Q	D	J	G	O

“P” sama dengan “P” pada kata yang dicari. Kata yang dicari sudah ditemukan namun pencarian menyebar masih berlanjut.

A	S	N	C	W
M	T	I	O	R
H	U	C	P	F
K	A	S	B	E
Q	D	J	G	O

Langkah 10:

“C” tidak sama dengan “P”, pencarian dilanjutkan.

A	S	N	C	W
M	T	I	O	R
H	U	C	P	F
K	A	S	B	E
Q	D	J	G	O

Langkah 11:

“O” tidak sama dengan “P”, pencarian dilanjutkan.

A	S	N	C	W
M	T	I	O	R
H	U	C	P	F
K	A	S	B	E
Q	D	J	G	O

Langkah 12:

Langkah 13:

“C” tidak sama dengan “P”, pencarian dilanjutkan.

A	S	N	C	W
M	T	I	O	R
H	U	C	P	F
K	A	S	B	E
Q	D	J	G	O

Langkah 14:

“U” tidak sama dengan “P”, pencarian berhenti.

A	S	N	C	W
M	T	I	O	R
H	U	C	P	F
K	A	S	B	E
Q	D	J	G	O

Pada proses pencarian, BFS membutuhkan 14 langkah perbandingan.

VII. KESIMPULAN

Pada permainan *Wordsearch* algoritma yang lebih mangkus dalam pencarian kata adalah algoritma DFS. Hal ini dikarenakan pada algoritma DFS pencarian dapat berhenti langsung jika sudah menemukan huruf yang diinginkan sedangkan pada BFS diharuskan mencari huruf-huruf selanjutnya yang masih berada pada tingkat yang sama. Walaupun pada kasus terburuk, DFS dapat menghasilkan jumlah perbandingan yang sama dengan BFS.

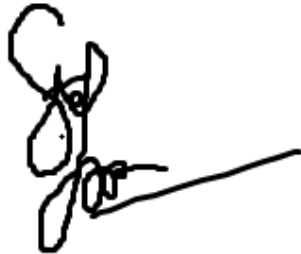
REFERENCES

- [1] Munir, Rinaldi, "Diktat Kuliah IF3051 Strategi Algoritma", Institut Teknologi Bandung, 2009.
- [2] http://www.dmoz.org/Games/Puzzles/Word_Search/ (diakses pada 17/12/2012 13:24)
- [3] <http://asia.gamespot.com/word-puzzle/platform/xbox360/> (diakses pada 17/12/2012 13:26)

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 2012



Stefan Lauren
13510034