Scientific Journal of Informatics
Vol. 1, No. 2, November 2014
p-ISSN 2407-7658 http://journal.unnes.ac.id/nju/index.php/sji e-ISSN 2460-0040

Penggunaan Metode *Depth First Search* (DFS) dan *Breadth First Search* (BFS) pada Strategi Game Kamen Rider Decade Versi 0.3

Budi Prasetiyo[1] , Maulidia Rahmah Hidayah[2]

[1,2]Program Studi Teknik Informatika, FMIPA Unnes,
Email: [1]prasemath@gmail.com, [2]maulidiarh@yahoo.com

Abstrak

Pada permainan *Game Kamen Rider Decade* ini sangat membutuhkan strategi yang tepat jika ingin memenangkan dengan mudah permainan ini. Penelitian ini bertujuan untuk mengimplementasikan metode *Dept First Search* (DFS) dan *Breadth First Search* (BFS) pada *Game Kamen Rider Decade*, yang merupakan permainan dengan strategi penyelesaiannya menggunakan metode pencarian buta (*blind search*). Pengumpulan data dilakukan dengan pendekatan kualitatif dengan metode deskriptif, dimana pengujian dilakukan dengan memainkan 3 kali masing-masing dengan metode selalu BFS dan selalu DFS. Hasil menunjukan peluang lebih besar memenangkan permainan ini adalah dengan strategi selalu BFS. Dimana kemampuan BFS pada permainan ini dapat berguna untuk pertahanan terhadap musuh.

Kata Kunci: *Game Kamen Rider Decade*, BFS, DFS, Strategi

1. PENDAHULUAN

Pada era teknologi dan informasi sekarang ini semua penggunaan teknologi menjadi semakin meningkat dan semakin intensif. Banyak dari anak kecil hingga orang dewasa menghabiskan waktunya untuk bermain permainan (*games*). Berbagai bentuk permainan berbasis komputer banyak bermunculan, baik yang sederhana maupun yang bersifat kompleks dari segi aturan permainan [1].

Pada suatu permainan perlu adanya metode untuk menyelesaikan permasalahan yang terjadi pada permainan itu. Salah satu metode yang ada adalah dengan metode pencarian. Dan salah satu permainan yang menggunakan metode pencarian lebih khususnya pencarian buta (*blind search*) dalam menyelesaikan permainannya adalah *Kamen Rider Decade*.

Kamen Rider Decade merupakan *game* yang berasal dari Jepang. Pada permainan ini, para *Kamen Rider* melawan para musuhnya yang memiliki senjata. Kesulitan dari *game* ini yaitu ketika salah memilih satu langkah atau strategi maka bisa saja cepat kalah dalam permianan ini. Untuk itu dibutuhkan adanya strategi yang baik untuk memenangkan permainan ini. Untuk menyelesaikan persoalan permainan ini dibutuhkan suatu metode *algoritma* yang efektif untuk dapat diterapkan. Dilihat dari kesulitan permainan ini, *Kamen Rider Decade* ini membentuk ruang solusi yang diorganisasikan ke dalam struktur pohon dinamis.

Struktur pohon dinamis sendiri dibangun dengan 2 metode transversal yaitu *Breadth First Search* (BFS) dan *Depth First Search* (DFS). Penelitian ini untuk mengetahui strategi yang efektif untuk diterapkan pada *Game Kamen Rider* melalui metode pencarian buta (*blind search*).

Permasalahan yang muncul dari penelitian ini adalah bagaimana perbedaan fungsi dari metode BFS dan DFS pada strategi *Kamen Rider* melawan musuhnya.

2. METODOLOGI

Penelitian ini menggunakan pendekatan kualitatif dengan metode deskriptif. Dimana pengumpulan data dilakukan dengan tiga cara. Pertama, melakukan observasi terhadap *Game Kamen Rider Decade* tersebut. Dengan cara mengetahui cara permainannya dan mengetahui peraturan yang ada dalam permainan tersebut dengan mengamatinya secara cermat kemudian mencatat temuan-temuan yang relevan. Yang kedua, dengan menguji strategi permainan sebanyak masing-masing 3 kali permainan dengan membandingkan ketika menggunakan strategi selalu BFS dan selalu DFS serta mencatat hasil pengujian ini. Ketiga, adalah dengan melakukan studi literatur melalui penelitian-penelitian yang sebelumnya dan buku-buku yang masih terkait.

2.1 Metode Strategi *Game Kamen Rider Decade*

Game Kamen Rider Decade merupakan game yang berasal dari Jepang. Diadatapsi dari serial *Kamen Rider*, terdiri dari 5 tokoh *Kamen Rider* yang akan menyerang musuhnya secara bergantian sebanyak 11 pada mission 1. Informasi mengenai *game* masih sangat terbatas dikarenakan tidak banyak buku atau website yang membahas tentang *game* ini. Untuk tampilan awal *game* dapat dilihat pada Gambar 1.

Gambar 1. Tampilan awal *game*

Cara bermain dari *game* ini adalah dengan cara: 1) secara acak salah satu *Kamen Rider* terpilih untuk bergerak (*move*), kemudian akan ada beberapa petak disekeliling *Kamen Rider* yang berwarna terang, 2) pemain memilih pergerakan akan bergeser ke arah mana. Jadi penggunaan BFS dan DFS disini untuk menentukan arah mana yang harus dipilih agar menang dalam permainan ini, 3) *Kamen Rider* jika berhadapan dengan musuh maka bisa memilih akan melawannya atau tetap bergeser.

Pemilihan langkah atau pergeseran *Kamen Rider* sangat mempengaruhi jalannya permainan, maka dari itu sangat dibutuhkan strategi yang tepat untuk memainkan permainan ini.

2.2 *Depth Search First* (DFS)

Dalam metode pencarian terbagi menjadi dua jenis yaitu pencarian buta (*blind search*) dan *heuristic search*. Pada pencarian buta terbagi menjadi dua macam yaitu pencarian BFS dan DFS.

Algoritma Depth First Search (DFS) adalah suatu metode pencarian pada sebuah pohon dengan menelusuri satu cabang sebuah pohon sampai menemukan solusi. Pencarian dilakukan pada satu *node* dalam setiap level dari yang paling kiri dan dilanjutkan pada *node* sebelah kanan. Jika solusi ditemukan maka tidak diperlukan proses *backtracking* yaitu penelusuran balik untuk mendapatkan jalur yang diinginkan. Pada metode DFS pemakaian memori tidak banyak karena hanya *node-node* pada lintasan yang akktif saja yang disimpan. Selain itu, jika solusi yang dicari berada pada level yang dalam dan paling kiri, maka DFS akan menemukannya secara cepat [2]. Contoh pergerakan DFS pada *Game Kamen Rider* dapat dilihat pada Gambar 2.

Gambar 2. Pergerakan DFS pada *Game Kamen Rider*

2.3 *Breadth Search First* (BFS)

Breadth First Search adalah suatu metode yang melakukan pencarian secara melebar yang mengunjungi simpul secara *preorder* yaitu mengunjungi suatu simpul kemudian mengunjungi semua simpul yang bertetangga dengan simpul tersebut dahulu. Selanjutnya, simpul yang belum dikunjungi dan bertetangga dengan simpul-simpul yang tadi dikunjungi, demikian seterusnya. Jika graf berbentuk pohon graf berakar, maka semua simpul pada aras *d* dikunjungi lebih dahulu sebelum simpul-simpul pada aras *d+1* [3]. Pada Gambar 3 contoh pergerakan BFS pada *Game Kamen Rider.*

Gambar 3. Pergerakan BFS pada *Game Kamen Rider*

2.4 Algoritma BFS dan DFS

Dalam metode pencarian baik yang BFS maupun DFS memiliki algoritma yang berbeda. Pada algoritma DFS adalah algoritma yang melakukan penelusuran dengan mengunjungi secara rekursif. Prosedur dari algoritma DFS dapat digambarkan sebagai berikut. a) Transversal dimulai dari simpul *v*, b) Kunjungi simpul *v*, c) Kunjungi simpul w yang bertetangga dengan *v*, d) Ulangi DFS mulai dari simpul *w*, e) Ketika mencapai simpul *u* sedemikian hingga semua simpul yang bertetangga dengannya telah dikunjungi, pencarian dirunut balik (*backtrack*) ke simpul terakhir yang dikunjungi sebelumnya dan mempunyai simpul *w* yang belum dikunjungi, f) Pencarian berakhir bila tidak ada lagi simpul yang belum dikunjungi yang dapat dicapai dari simpul yang telah dikunjungi [3].
Pseucode dari algoritma DFS dapat dilihat pada Gambar 4.

```
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] <- true
  for w <- 1 to n do
    if A[v,w]=1 then
      {simpul v dan simpul w bertetangga }
      if not dikunjungi[w] then
        DFS(w)
      endif
    endif
  endfor
```

Gambar 4. *Pseucode* dari algoritma DFS

Algoritma BFS adalah teknik umum yang digunakan untuk melakukan transversal pada graf. Prosedur algoritma BFS adalah sebagai berikut. a) Traversal dimulai dari simpul *v*, b) Kunjungi semua simpul *v*, c) Kunjungi semua simpuul yang bertetangga dengan simpul *v* terlebih dahulu, d) Kunjungi simpul yang belum dikunjungi dan bertetangga dengan simpul-simpul yang tadi dikunjungi, demikian seterusnya [3].

Pseudocode algoritma BFS dapat dilihat pada Gambar 5.

```
procedure BFS(input v:integer)
{
  Traversal graf dengan algoritma pencarian BFS.
  Masukan: v adalah simpul awal kunjungan
  Keluaran: semua simpul yang dikunjungi dicetak ke
layar
}

Deklarasi
  w : integer
  q : antrean

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

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

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

  function AntreanKosong(input q:antrean) -> boolean
  { true jika antrean q kosong, false jika sebaliknya }

Algoritma
  BuatAntrean(q) { buat antrean kosong }

  write(v) { cetak simpul awal yang dikunjungi }
  dikunjungi[v] <- true { simpul v telah dikunjungi,
        tandai dengan true }

  MasukAntrean(q,v) { masukkan simpul awal kunjungan ke
        dalam antrean }
        { kunjungi semua simpul graf selama antrean
        belum kosong }

  while not AntreanKosong(q) do
    HapusAntrean(q,v) { simpul v telah dikunjungi,
        hapus dari antrean }
    for tiap simpul w yang
        bertetangga dengan simpul v do
      if not dikunjungi[w] then
        write(w) {cetak simpul yang dikunjungi}
        MasukAntrean(q,w)
```

Gambar 5. *Pseudocode* algoritma BFS

Perbedaan antara DFS dan BFS hanya pada pemasukan daftar kunjungan ke simpul tetangga. Jika dalam BFS akan mengunjungi semua simpul yang bertetangga untuk memperlebar dengan tidak akan melewatinya lagi jika sudah melewati sampai semuanya dikunjungi. Jika pada DFS, ingin memperluas simpul dengan masuk ke dalam grafik, jadi memasukan simpul yang baru ke awal kunjungan [4].

3. HASIL DAN PEMBAHASAN

Berdasarkan data yang diperoleh melalui observasi langsung pada *game* ini dan memainkan *game* ini dengan selalu BFS dan selalu DFS sebanyak 3 kali menemukan perbedaan fungsi serta membantu dalam mengatur strategi jika mengetahui fungsi dari BFS dan DFS pada *game* ini.

Pada percobaan tersebut, ketika menggunakan selalu DFS maka akan kalah dalam waktu yang singkat. Namun pada penggunaan selalu BFS maka pada permainan 3 kali maka 1 kali permainan bisa menang dan dalam waktu yang masih bertahan lama. Kemungkinan penggunaan selalu BFS menang atau tidaknya juga dipengaruhi dari pemilihan pemain yang memang secara acak untuk dimainkan.

Secara pengetahuan umum mengenai strategi menekankan pada pengetahuan yang melebar (*breadth*) pada pengembangan produk kelompok, sedangkan secara khusus fokus untuk pengetahuan yang dalam (*depth*) [5]. Sehingga dapat dianalogikan bahwa metode *Breadth First Search* pada dasarnya memiliki fungsi sebagai pengatur strategi pengembangan pengetahuan kelompok yang baik, sedangkan pada DFS digunakan secara khusus seperti dalam hal tertentu.

Pada penggunaan metode DFS, para *Kamen Rider* berjalan maju yang membuat posisinya semakin dekat dengan musuhnya serta jika pada DFS maka para *Kamen Rider* akan melawan musuh lebih individu, berbeda dengan metode BFS dimana para *Kamen Rider* bergerak menyamping kiri maka membuat tempo musuh mendekat para *Kamen Rider* menjadi semakin lama serta para *Kamen Rider* akan berkumpul pada salah satu yang membuat kerjasama antar *Kamen Rider* menjadi semakin terlihat. Untuk hasil perbandingan dari algoritma BFS dan DFS dapat dilihat pada Tabel 1.

Tabel 1. Perbandingan BFS dan DFS

Aspek	BFS	DFS
Moving	25	18
Waktu rata-rata	06:29	05:58
keunggulan	Pertahanan	Kecepatan

4. SIMPULAN

Berdasarkan hasil pengujian bahwa dalam permainan *Kamen Rider* dibutuhkan strategi yang tepat baik melalui metode BFS maupun DFS. BFS pada *game* ini lebih bermanfaat untuk pertahanan dan kemungkinan lebih besar menang dibanding menggunakan DFS. Metode DFS bermanfaat kepada kecepatan permainan, sehingga dibutuhkan kolaborasi dari dua metode ini pada strategi permainan *Kamen Rider* sehingga dapat mudah memenangkan permainan ini.

5. REFERENSI

[1] Dana Creamer. 2007. *The Application of Artificial Intelligent to Solve a Physical Puzzle*. Departement of Computer and Information Sciences, Indiana University South Bend.
[2] Rinaldi Munir. 2005. *BFS dan DFS*. Bandung: Institut Teknologi Bandung.
[3] Cormen, T.H., Leiserson, C.E., Rivest, R.L. dan Stein, C. 2009. *Introduction to Algorithm.* Third Edition. Massachusetts: MIT Press.
[4] --, 6. 006 2011. *Intro to Algorithms: Recitation*. --:--.
[5] Scott F. Turner, *et al*. 2002. *Exploring Depth Versus Breadth in Knowledge Management Strategies. Computational & Mathematical Organization Theory*. Kluwer Academic Publishers 8, 49-73.