Penerapan Algoritma Dijkstra pada Permainan Counter-Strike

Muhamad Visat Sutarno 13513037
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
visat@std.itb.ac.id

***Abstract*— Counter-Strike adalah *video game* yang populer dimainkan sejak tahun 2000 hingga sekarang. Makalah ini akan membahas tentang pengaplikasian algoritma Dijkstra untuk mendapatkan rute terpendek dari tempat seorang pemain Counter-Strike berada ke tempat yang dia inginkan. Rute yang dipilih adalah rute yang memiliki waktu tempuh paling minimal sehingga pemain dapat dengan cepat sampai ke tujuan. Hal ini bertujuan untuk meningkatkan peluang menangnya seorang pemain Counter-Strike saat waktu yang tersisa sedikit.**

***Keywords*—Counter-Strike, Dijkstra, Graf, Rute Terpendek, Simpul.**

I. PENDAHULUAN

Valve Corporation adalah sebuah perusahaan yang bergerak di bidang pengembangan dan penerbit *video game* yang berasal dari Amerika Serikat. Pada tahun 2000, Valve merilis Counter-Strike pada platform Microsoft Windows. Counter-Strike sendiri sebenarnya adalah modifikasi dari permainan Half-Life oleh Minh Le and Jess Cliffe pada tahun 1999 yang sebelum akhirnya dibeli oleh Valve dan mereka dijadikan karyawannya.

Counter-Strike adalah permainan *first-person shooter* yang mempunyai pilihan untuk menjadikan pemain sebagai tim *terrorist*, tim *counter-terrorist* (CT), maupun hanya menjadi seorang penonton pertandingan saja. Setiap pertandingan dapat berbeda-beda tujuan misi tiap timnya tergantung dari *map* yang dipilih. Pertandingan terdiri dari beberapa ronde. Pada tiap rondenya, kedua tim akan saling mengalahkan tim lawan untuk mencapai tujuan dari misi mereka, tergantung pada mode yang dipilih. Setiap pemain dapat membeli senjata dan peralatan pada awal ronde dengan uang yang dimilikinya. Uang tersebut dapat didapatkan dari membunuh pemain tim lawan ataupun memenangi sebuah ronde.

Ada tiga jenis *map* yang paling sering dimainkan. Yang pertama adalah *Bomb Maps*, yaitu tempat di mana salah satu anggota tim teroris membawa satu bom dan harus memasangnya di suatu tempat yang ditentukan dan memastikan bomnya tidak dijinakkan oleh tim CT. Tim CT memenangi ronde jika tidak ada bom yang meledak. Yang kedua adalah *Hostage Maps*, yaitu tempat di mana tim CT harus membebaskan sandera yang ditahan oleh tim teroris. Tim teroris menang jika masih ada sandera yang tertahan dan waktu habis. Yang terakhir adalah *VIP Maps*, yaitu tempat di mana salah satu anggota tim CT adalah seorang VIP dan tim CT harus memastikan VIP tersebut sampai ke tujuan yang telah ditentukan *map*. Tim teroris memenangi ronde tersebut jika VIP terbunuh ataupun waktu telah habis.[1]

Gambar 1. Pemain tim teroris yang membawa bom dalam pertandingan *Bomb Maps*[5]

Untuk memenangi pertandingan Counter-Strike, tentu saja dibutuhkan keahlian dan kerjasama dalam tim. Selain itu, kecepatan dalam menanggapi suatu kejadian juga menentukan jalannya pertandingan. Sebagai contoh, jika anggota dalam satu tim terbunuh, anggota tim yang lain harus mengetahui di mana letak terakhir rekannya yang terbunuh. Hal ini dapat menentukan strategi yang akan digunakan selanjutnya, yaitu berusaha menjauhi tempat tersebut ataupun mendatangi tempat tersebut dan berusaha untuk membunuh anggota tim lawan. Contoh lainnya adalah ketika bom telah dipasang, tim CT harus secepatnya bergerak ke tempat bom terpasang.

Mendatangi tempat anggota tim lawan berada atau ke tempat bom terpasang harus dilakukan dengan cepat. Hal itu berarti pemain harus mencari rute yang terpendek dengan waktu tempuh yang paling minimal. Dengan algoritma Dijkstra, rute terpendek tersebut dapat dicari.

II. LANDASAN TEORI

A. Graf

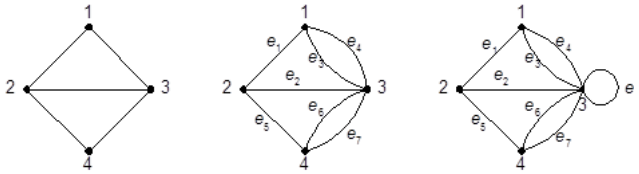
Graf ditulis dalam notasi G = (V, E), dengan:

$$V = \{v_1, v_2, \dots, v_n\}$$
$$E = \{e_1, e_2, \dots, e_n\}$$

V adalah himpunan tidak-kosong dari simpul dan E adalah himpunan sisi yang menghubungkan sepasang simpul.

Berdasarkan ada tidaknya gelang atau sisi ganda pada suatu graf, maka graf digolongkan menjadi dua jenis:

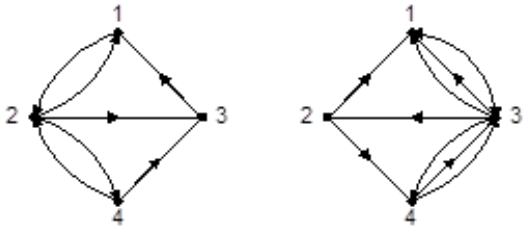
1. Graf Sederhana
   Graf yang tidak mengandung gelang maupun sisi-ganda dinamakan graf sederhana.
2. Graf Tak-sederhana
   Graf yang mengandung sisi ganda atau gelang dinamakan graf tak-sederhana.



Gambar 2. Graf sederhana dan graf tak-sederhana[2]

Berdasarkan orientasi arah pada sisi, maka secara umum graf dibedakan atas dua jenis:

1. Graf Tak-berarah
   Graf yang sisinya tidak mempunyai orientasi arah disebut graf tak-berarah. Urutan pasangan simpul yang dihubungkan oleh sisi pada graf ini tidak diperhatikan.
2. Graf Berarah
   Graf yang setiap sisinya diberikan orientasi arah disebut sebagai graf berarah. Urutan pasangan simpul yang dihubungkan oleh sisi pada graf ini diperhatikan, karena bisa saja bobot keduanya berbeda.



Gambar 3. Graf berarah[2]

B. Algoritma Greedy

Algoritma greedy merupakan metode yang paling populer untuk memecahkan persoalan optimasi. Persoalan optimasi adalah persoalan mencari solusi yang paling optimum. Hanya ada dua macam persoalan optimasi, yaitu maksimasi dan minimasi.

Algoritma greedy adalah algoritma yang memecahkan masalah langkah per langkah. Pada setiap langkah:

1. Mengambil pilihan yang terbaik yang dapat diperoleh pada saat itu tanpa memperhatikan konsekuensi ke depan.
2. Berharap bahwa dengan memilih optimum lokal pada setiap langkah akan berakhir dengan optimum global.

Elemen-elemen algoritma greedy:

1. Himpunan kandidat, C
   Berisi elemen-elemen pembentuk solusi.
2. Himpunan solusi, S
   Berisi kandidat-kandidat yang terpilih sebagai solusi persoalan.
3. Fungsi seleksi
   Memilih kandidat yang paling memungkinkan mencapai solusi optimal. Kandidat yang sudah dipilih pada suatu langkah tidak pernah dipertimbangkan lagi pada langkah selanjutnya.
4. Fungsi kelayakan
   Memeriksa apakah suatu kandidat yang telah dipilih dapat memberikan solusi yang layak, yakni kandidat tersebut bersama-sama dengan himpunan solusi yang sudah terbentuk tidak melanggar kendala (constraints) yang ada. Kandidat yang layak dimasukkan ke dalam himpunan solusi, sedangkan kandidat yang tidak layak dibuang dan tidak pernah dipertimbangkan lagi.
5. Fungsi obyektif
   Fungsi yang memaksimumkan atau meminimumkan nilai solusi (misalnya panjang lintasan, keuntungan, dan lain-lain).

Algoritma greedy melibatkan pencarian sebuah himpunan bagian, S, dari himpunan kandidat, C, yang dalam hal ini, S harus memenuhi beberapa kriteria yang ditentukan, yaitu menyatakan suatu solusi dan S dioptimisasi oleh fungsi obyektif.

Skema algoritma Greedy secara umum adalah sebagai berikut:

```
function Greedy(
input C: himpunan_kandidat)
→ himpunan_kandidat
{ Mengembalikan solusi dari persoalan
optimasi dengan algoritma greedy
Masukan: himpunan kandidat C
Keluaran: himpunan solusi yang
bertipe himpunan_kandidat }

Deklarasi
  X: kandidat
  S: himpunan_kandidat

Algoritma
  S ← {} {inisialisasi S dengan
kosong}
```

```
  while (not SOLUSI(S) and C <> {})
do
    { pilih sebuah kandidat dari
himpunan kandidat C }
    X ← SELEKSI(C)
    { elemen himpunan kandidat
berkurang satu }
    C ← C - {X}
    if LAYAK(S ∪ {X}) then
      { tambahkan elemen ke dalam
himpunan solusi }
      S ← S ∪ {X}
    end if
  end while
  { SOLUSI(S) telah diperoleh atau
C={}}

  if SOLUSI(S) then
    → S
  else
    → {}
```

C. Algoritma Dijkstra

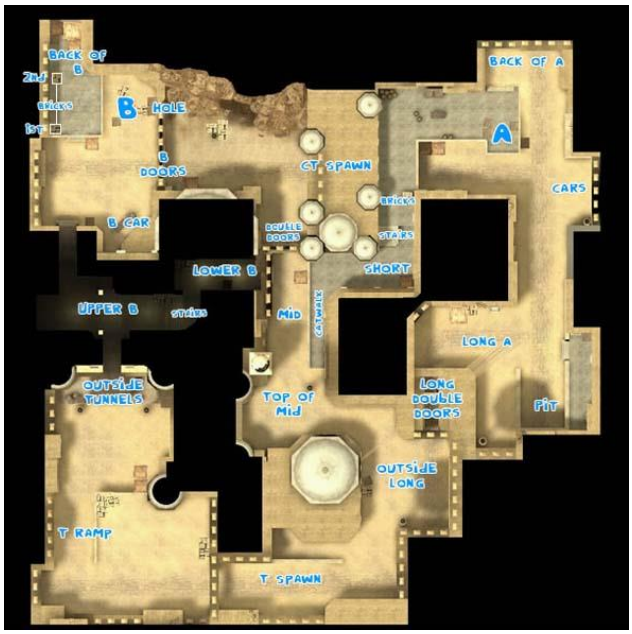
Algoritma Dijkstra pada dasarnya menggunakan strategi *greedy*. Pada setiap langkah, ambil sisi yang berbobot minimum yang menghubungkan sebuah simpul yang sudah terpilih dengan sebuah simpul lain yang belum terpilih. Lintasan dari simpul asal ke simpul yang baru haruslah merupakan lintasan yang terpendek diantara semua lintasannya ke simpul-simpul yang belum terpilih.

Pseudocode algoritma Dijkstra sebagai berikut:

```
procedure Dijkstra (
input G: weighted_graph,
input a: initial_vertex)

Deklarasi
     S: himpunan  simpul solusi
     L: array[1..n] of real
     { L(z) berisi panjang lintasan
terpendek dari a ke z}

Algoritma
  for i←1 to n
    L(vi) ← ∞
  end for

  L(a) ← 0
  S ← { }

  while z ∉ S do
    u ← simpul yang bukan di dalam S
dan memiliki L(u) minimum
    S ← S ∪ {u}
    for semua simpul v yang tidak
terdapat di dalam S
      if L(u) + G(u,v) < L(v) then
        L(v) ← L(u) + G(u,v)
    end for
  end while
```

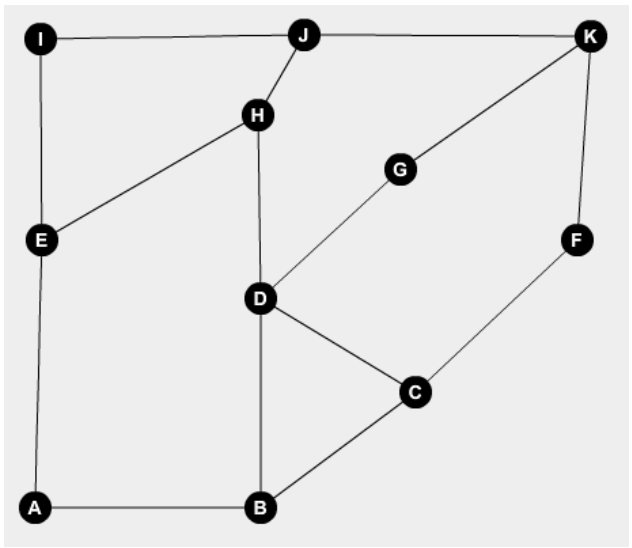
III. Analisis dan Pembahasan

Pada pembahasan kali ini, penulis akan menginstansiasi permasalahan yaitu pemain yang sedang memainkan salah satu mode permainan yaitu *Bomb Maps* dengan *map* bernama de_dust2.



Gambar 4. Denah *map* de_dust2[6]

Kemudian, penulis menyederhanakan *map* tersebut menjadi graf seperti gambar berikut:



Gambar 5. Graf dari *map* de_dust2

Lalu, dengan memperkirakan bobot antara graf dengan memainkan *map* tersebut secara langsung, penulis merepresentasikan matriks ketetanggaan yang berisi bobot setiap sisi seperti berikut:

	A	B	C	D	E	F	G	H	I	J	K
A	0	9	∞	∞	14	∞	∞	∞	∞	∞	∞
B	9	0	10	9	∞	∞	∞	∞	∞	∞	∞
C	∞	10	0	10	∞	12	∞	∞	∞	∞	∞
D	∞	13	10	0	∞	∞	13	9	∞	∞	∞
E	14	∞	∞	∞	0	∞	∞	10	15	∞	∞
F	∞	∞	12	∞	∞	0	∞	∞	∞	∞	21
G	∞	∞	∞	13	∞	∞	0	∞	∞	∞	13
H	∞	∞	∞	9	10	∞	∞	0	∞	7	∞
I	∞	∞	∞	∞	15	∞	∞	∞	0	14	∞
J	∞	∞	∞	∞	∞	∞	∞	7	14	0	19
K	∞	∞	∞	∞	∞	16	13	∞	∞	14	0

Tabel 1. Representasi matriks ketetanggaan

Instansiasi permasalahan kali ini pemain merupakan anggota tim CT. Pemain saat ini sedang berada di simpul A. Kemudian tiba-tiba dia mendapat pemberitahuan jika bom telah terpasang di simpul K. Untuk itu, pemain harus mencari rute terpendek agar dapat menjinakkan bom dalam waktu yang ditentukan.

Dengan menggunakan algoritma Dijkstra yang telah dibahas pada bab sebelumnya, kita dapat menentukan rute yang harus dilalui pemain. Langkah-langkahnya adalah:

1. Berikan nilai 0 pada simpul A dan nilai tak berhingga pada simpul lainnya.
2. Untuk simpul yang sedang diproses, cari semua simpul yang bertetangga dengannya. Simpan nilai pada simpul tetangga tersebut yang paling minimum, antara nilai sebelumnya dengan nilai simpul yang sedang diproses ditambah bobot dari sisi yang menghubungkan keduanya.
3. Jika semua simpul telah diproses, algoritma selesai. Jika belum, pilih simpul yang belum diproses dan memiliki nilai yang terkecil dan lakukan kembali langkah 2.

Langkah-langkah tersebut akan menghasilkan pemrosesan simpul-simpul seperti berikut:

- Simpul awal

	A	B	C	D	E	F	G	H	I	J	K
Nilai	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
Prev	-	-	-	-	-	-	-	-	-	-	-

- Proses simpul A

	A	B	C	D	E	F	G	H	I	J	K
Nilai	0	9	∞	∞	14	∞	∞	∞	∞	∞	∞
Prev	-	A	-	-	A	-	-	-	-	-	-

- Proses simpul B

	A	**B**	C	D	E	F	G	H	I	J	K
Nilai	0	9	19	18	14	∞	∞	∞	∞	∞	∞
Prev	-	A	B	B	A	-	-	-	-	-	-

- Proses simpul E

	A	**B**	C	D	**E**	F	G	H	I	J	K
Nilai	0	9	19	18	14	∞	∞	24	29	∞	∞
Prev	-	A	B	B	A	-	-	E	E	-	-

- Proses simpul D

	A	**B**	C	**D**	**E**	F	G	H	I	J	K
Nilai	0	9	19	18	14	∞	31	24	29	∞	∞
Prev	-	A	B	B	A	-	D	E	E	-	-

- Proses simpul C

	A	**B**	**C**	**D**	**E**	F	G	H	I	J	K
Nilai	0	9	19	18	14	31	31	24	29	∞	∞
Prev	-	A	B	B	A	C	D	E	E	-	-

- Proses simpul H

	A	**B**	**C**	**D**	**E**	F	G	**H**	I	J	K
Nilai	0	9	19	18	14	31	31	24	29	31	∞
Prev	-	A	B	B	A	C	D	E	E	H	-

- Proses simpul I

	A	**B**	**C**	**D**	**E**	F	G	**H**	**I**	J	K
Nilai	0	9	19	18	14	31	31	24	29	31	∞
Prev	-	A	B	B	A	C	D	E	E	H	-

- Proses simpul F

	A	**B**	**C**	**D**	**E**	**F**	G	**H**	**I**	J	K
Nilai	0	9	19	18	14	31	31	24	29	31	52
Prev	-	A	B	B	A	C	D	E	E	H	F

- Proses simpul G

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	J	K
Nilai	0	9	19	18	14	31	31	24	29	31	44
Prev	-	A	B	B	A	C	D	E	E	H	G

- Proses simpul J

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	K
Nilai	0	9	19	18	14	31	31	24	29	31	44
Prev	-	A	B	B	A	C	D	E	E	H	G

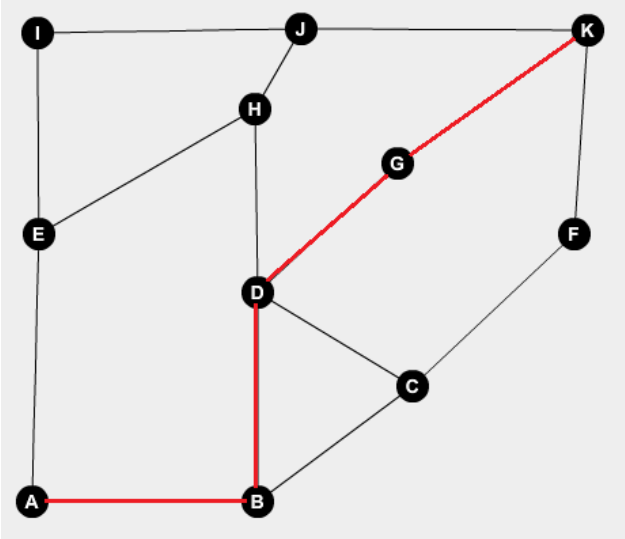
- Proses simpul K

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**
Nilai	0	9	19	18	14	31	31	24	29	31	44
Prev	-	A	B	B	A	C	D	E	E	H	G

- Algoritma selesai

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**
Nilai	0	9	19	18	14	31	31	24	29	31	44
Prev	-	A	B	B	A	C	D	E	E	H	G

Dari algoritma Dijkstra tersebut, dapat diketahui jarak terdekat dari simpul A ke simpul K adalah 44 satuan jarak. Dengan menelusuri balik dari "Prev", dapat diketahui rute terpendek yang dapat dilalui adalah K-G-D-B-A, atau jika memulainya dari simpul A maka rute terpendeknya adalah A-B-D-G-K.



Gambar 6. Graf dengan rute terpendek dari simpul A ke simpul K

Penulis lalu melakukan pengujian langsung pada permainan Counter-Strike untuk membuktikan hasil dari algoritma Dijkstra tersebut.

Gambar 7. Penulis sedang berada pada simpul A

Setiap pengujian dilakukan, penulis berusaha untuk memulai perjalanan rute dari tempat dan sudut pandang yang persis sama. Hal ini diharapkan untuk meminimalisasi terjadinya *error* pada hasil pengujian.

Gambar 8. Penulis telah sampai di simpul K

Pengujian dilakukan sebanyak lima kali dan dilakukan dengan melewati rute yang berbeda-beda. Hasil yang diperoleh adalah:

Rute	Waktu Tempuh (detik)
A-B-D-G-K	25,45
A-E-H-J-K	28,64
A-B-C-F-K	28,87
A-E-I-J-K	30,05
A-B-D-H-J-K	32,34

Tabel 2. Hasil pengujian

Dari tabel tersebut dapat disimpulkan bahwa algoritma Dijkstra mampu memberikan informasi rute terpendek dengan tepat, yaitu dengan hasil waktu tempuh yang paling minimum di antara yang lain sesuai pengujian yang dilakukan oleh penulis.

IV. KESIMPULAN

Permainan strategi seperti Counter-Strike, suatu sumberdaya seperti waktu adalah hal yang sangat berharga walaupun jumlahnya sedikit. Saat mengejar lawan, berusaha kabur dari lawan, ataupun berusaha untuk menjinakkan bom, waktu satu detik pun dapat mengubah alur jalannya sebuah pertandingan. Strategi yang digunakan haruslah tepat, salah satunya adalah strategi dalam mencari rute yang terpendek saat mencapai tujuan dengan waktu terbatas. Dengan algoritma Dijkstra, rute yang dihasilkan merupakan rute terpendek, sehingga pemain tidak akan membuang-buang waktu yang berharga. Jadi, algoritma Dijkstra dapat membantu pemain memenangkan pertandingan dalam permainan Counter-Strike.

VII. UCAPAN TERIMA KASIH

Penulis mengucapkan puji dan syukur kepada Allah SWT untuk segala rahmat-Nya sehingga penulis bisa menyelesaikan makalah ini. Penulis juga mengucapkan terima kasih kepada dosen pengajar Strategi Algoritma, Bapak Rinaldi Munir dan Ibu Nur Ulfa Maulidevi, untuk segala pengajaran yang telah diberikan mengenai Strategi Algoritma, khususnya mengenai algoritma Dijkstra yang menjadi salah satu dasar penting dalam penulisan makalah ini.

REFERENSI

[1] Blaze, Jill. 2015. *Counter-Strike* hlm. 4–5. CreateSpace.
[2] Munir, Rinaldi. 2009. *Matematika Diskrit*. Bandung: Informatika.
[3] Munir, Rinaldi. 2009. *Diktat Kuliah IF2211 Strategi Algoritma*. Bandung: Informatika.
[4] Liem, Inggriani. 2008. *Diktat Struktur Data*. Bandung: Program Studi Informatika Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung.
[5] http://vignette1.wikia.nocookie.net/egamia/images/b/bf/Csmapdust2.jpg/revision/latest?cb=20070414235247 Diakses 4 Mei 2015 pukul 20.26 WIB.
[6] http://www.gamasutra.com/db_area/images/blog/211005/de_dust2.jpg Diakses 4 Mei 2015 pukul 20.28 WIB.

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, 5 Mei 2015

Muhamad Visat Sutarno
13513037