Penerapan Pohon Merentang pada Permainan FTL:Faster Than Light

Sandy Gunawan Tanuwijaya, 13510025
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
sandy.gunawan@mail.students.itb.ac.id

***Abstrak*— FTL: Faster Than Light adalah permainan *roguelike*/simulasi pesawat luar angkasa yang dirilis untuk Windows, Linux dan Mac OS. Game ini dibuat oleh grup independen/*indie* Subset Games dengan pengadaan dana secara *crowdfunding* lewat *Kickstarter* dan dirilis secara *online* di berbagai situs seperti Steam dan GOG (*Good Old Games*) pada tanggal 14 September 2012. Bisa disebut sebagai *Firefly* (judul film)/*Star Trek* ala *Rogue-like*, dalam permainan ini, pemain bertugas untuk memandu kru sebuah pesawat luar angkasa mengarungi luar angkasa yang penuh bahaya dan kembali ke markas dengan selamat dan mengalahkan pemberontak. Makalah ini membahas tentang aplikasi pohon pada permainan beserta fungsi-fungsinya.**

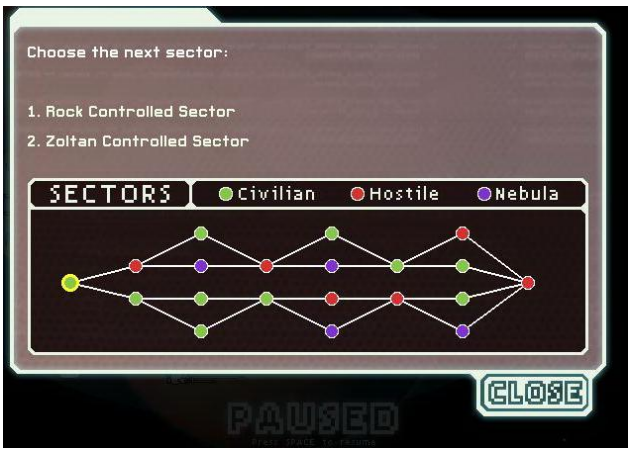
***Kata Kunci*—Roguelike, Kickstarter, FTL:Faster Than Light, Pohon, Pohon Merentang**

I. PENDAHULUAN

Gambar 1: Cover FTL

Dalam permainan FTL: Faster Than Light, pemain diberikan satu dari berbagai model pesawat luar angkasa dengan empat kru dari beberapa ras yang ada, letak ruangan, dan konfigurasi senjata yang berbeda-beda.

Setelah memilih pesawat luar angkasa, pemain akan diterjunkan pada sektor galaksi pertama (yang pasti selalu netral/*civilian*, sektor kedua dan seterusnya selalu diatur secara *random*), dan pemain harus tiba pada sektor kedelapan (sektor Federasi) untuk melawan *final boss* dan memenangkan permainan.



Gambar 2: Contoh Deretan Sektor pada Game (urutan selalu di-*random* setiap permainan)

Sebuah sektor terdiri dari lusinan *waypoint* yang berfungsi sebagai simpul yang menyambung titik start dengan titik exit sektor, juga terdapat beberapa *waypoint* khusus yang berisi *event* yang biasanya di-*generate* secara random seperti sinyal SOS (yang mungkin juga merupakan jebakan), toko untuk memperbaiki pesawat maupun membeli perlengkapan atau senjata, atau bahkan jebakan yang dipasang oleh pasukan pemberontak. Di samping itu, beberapa *waypoint* biasa juga dapat berisi pesawat musuh biasa, awan nebula tebal yang dapat mengurangi daya pandang pesawat, sabuk asteroid dengan ancaman asteroid yang dapat menabrak pesawat kapanpun, maupun daerah dekat matahari yang dapat menyemburkan lidah api secara berkala. Untuk melewati sebuah sektor, pesawat luar angkasa pemain harus tiba di *waypoint* bertanda exit menggunakan *warp-drive* milik pesawat. Setiap *warp* dilakukan, pesawat memerlukan satu point bahan bakar, dan jika bahan bakar habis, maka pemain tidak akan dapat melanjutkan permainan. Akan tetapi, pemain tidak bisa tinggal diam karena searmada kapal pemberontak yang mengejar pemain dan ikut maju setiap pemain melakukan *warp*, dan jika pesawat pemain terdapat di tengah-tengah armada tersebut, maka akan sulit untuk melarikan diri dengan selamat.

Gambar 3: Berbagai ras yang dapat ditemukan pada game ini dengan kelebihan dan kekurangan masing-masing (baik manusia maupun alien)

Jika pemain bertemu dengan pesawat musuh dalam satu sektor, maka mode permainan akan berubah ke mode pertempuran. Di sini pemain dapat memilih salah satu bagian dari pesawat musuh menggunakan senjata yang ada. Setiap ruangan yang terkena serangan akan menimbulkan efek tertentu, misalnya jika ruangan senjata terkena tembakan, maka pesawat tidak dapat menembakan senjatanya sampai ruang tersebut diperbaiki, dan lain-lain. Pemain juga harus selalu menjaga keselamatan kru pesawat, mencegah adanya peyusup dan mengalokasikan daya untuk setiap komponen/ruangan pesawat (mesin, senjata, perisai, suplai oksigen, dll) agar semua sistem pesawat tetap operasional.

Setibanya di sektor federasi, tugas pemain belum selesai, karena pemain harus menghadapi kapal induk milik pemberontak sekaligus mencegahnya untuk sampai di markas pusat.

Gambar 4: Tampilan pada saat pertempuran

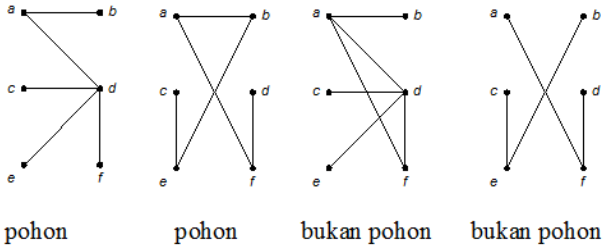
II. POHON

Pohon adalah graf tak berarah terhubung yang tidak mengandung sirkuit, sedangkan hutan adalah graf tidak terhubung yang tidak mengandung sirkuit, dengan semua komponen graf terhubung merupakan pohon.

Pohon memiliki sifat-sifat sebagai berikut:

Misalkan G = (V, E) adalah graf tak-berarah sederhana dan jumlah simpulnya n. Maka, semua pernyataan di bawah ini adalah ekivalen:

1. *G adalah pohon.*
2. *Setiap pasang simpul di dalam G terhubung dengan lintasan tunggal.*
3. *G terhubung dan memiliki m = n – 1 buah sisi.*
4. *G tidak mengandung sirkuit dan memiliki m = n – 1 buah sisi.*
5. *G tidak mengandung sirkuit dan penambahan satu sisi pada graf akan membuat hanya satu sirkuit.*
6. *G terhubung dan semua sisinya adalah jembatan.*



Gambar 5: Contoh Graf yang Merupakan Pohon dan Bukan Pohon

Ada pula pohon merentang/*spanning tree* yang merupakan hasil graf yang diperoleh dengan memutus sirkuit dalam graf. Pohon merentang yang memiliki bobot minimum disebut dinamakan *minimum spanning tree*. Terdapat beberapa algoritma untuk menemukan *minimum spanning tree* suatu pohon, antara lain algoritma Prim dan algoritma Kruskal.

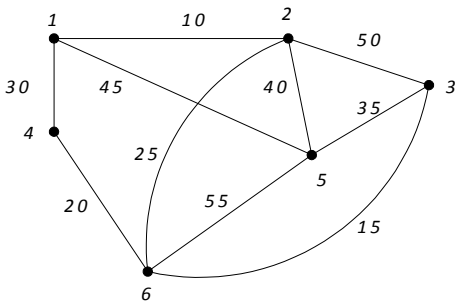
Algoritma Prim memiliki langkah-langkah sebagai berikut:

1. ambil sisi dari graf *G* yang berbobot minimum, masukkan ke dalam *T*.
2. Pilih sisi (u, v) yang mempunyai bobot minimum dan bersisian dengan simpul di *T*, tetapi (u, v) tidak membentuk sirkuit di *T*. Masukkan (u, v) ke dalam *T*.
3. Ulangi langkah 2 sebanyak $n - 2$ kali.

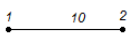
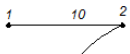
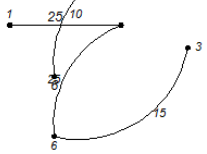
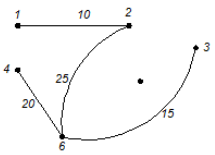
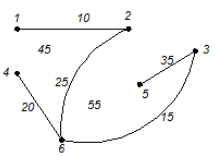
```
procedure Prim(input G : graf, output T : pohon)
{ Membentuk pohon merentang minimum T dari graf terhubung-berbobot G.
Masukan: graf-berbobot terhubung G = (V, E), dengan |V| = n
Keluaran: pohon rentang minimum T = (V, E')
}
Deklarasi
  i, p, q, u, v : integer

Algoritma
  Cari sisi (p,q) dari E yang berbobot terkecil
  T ← {(p,q)}
  for i←1 to n-2 do
    Pilih sisi (u,v) dari E yang bobotnya terkecil namun
    bersisian dengan simpul di T
    T ← T ∪ {(u,v)}
  endfor
```

Gambar 6: *Pseudo-code* algoritma Prim

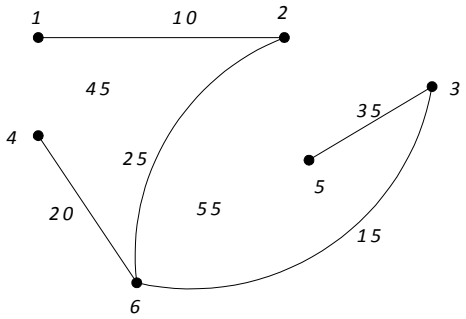
Berikut diberikan contoh penyelesaian menggunakan algoritma Prim, dengan pohon yang diberikan sebagai berikut.



Gambar 7: Contoh Pohon

Langkah	Sisi	Bobot	Pohon rentang
1	(1, 2)	10	
2	(2, 6)	25	
3	(3, 6)	15	
4	(4, 6)	20	
5	(3, 5)	35	

Gambar 8: Langkah-langkah pencarian pohon merentang minimum



Gambar 9: Hasil pohon merentang minimum

Dengan menggunakan algoritma Prim, ditemukan pohon merentang minimum seperti di atas dengan total nilai: 10 + 25 + 15 + 20 + 35 = 105.

Algoritma Kruskal memiliki langkah penyelesaian sebagai berikut:

1. Sisi-sisi dari graf sudah diurut menaik berdasarkan bobotnya – dari bobot kecil ke bobot besar
2. *T* masih kosong
3. Pilih sisi (*u*, *v*) dengan bobot minimum yang tidak membentuk sirkuit di *T*. Tambahkan (*u*, *v*) ke dalam *T*.
4. Ulangi langkah 3 sebanyak *n* – 1 kali.

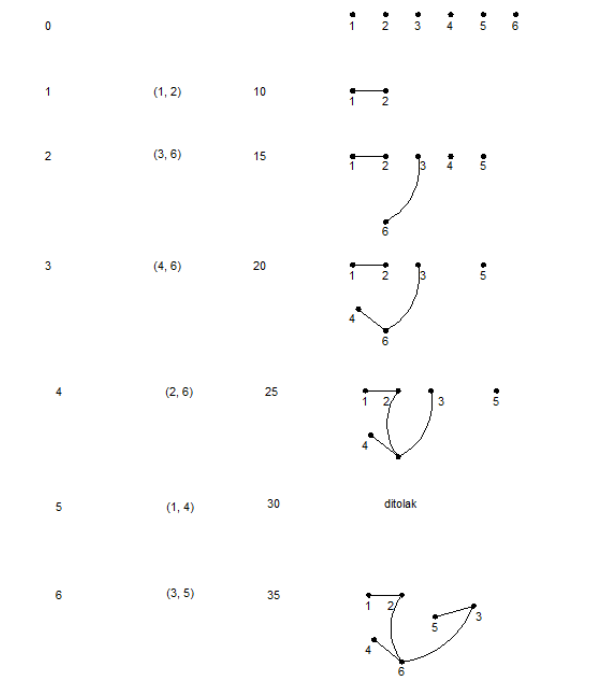
```
procedure Kruskal(input G : graf, output T : pohon)
{ Membentuk pohon merentang minimum T dari graf terhubung -berbobot G.
Masukan: graf-berbobot terhubung G = (V, E), dengan |V| = n
Keluaran: pohon rentang minimum T = (V, E')
}
Deklarasi
  i, p, q, u, v : integer

Algoritma
  ( Asumsi: sisi-sisi dari graf sudah diurut menaik
    berdasarkan bobotnya - dari bobot kecil ke bobot
    besar)
  T ← {}
  while jumlah sisi T < n-1 do
    Pilih sisi (u,v) dari E yang bobotnya terkecil
    if (u,v) tidak membentuk siklus di
```

```
T then
        T ← T ∪ {(u,v)}
    endif
  endfor
```

Gambar 10: *Pseudo-code algoritma Kruskal*

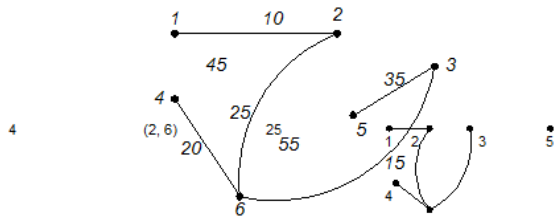
Berikut diberikan contoh penyelesaian menggunakan algoritma Kruskal pada pohon pada gambar 7.

Sisi	(1,2)	(3,6)	(4,6)	(2,6)	(1,4)	(3,5)	(2,5)	(1,5)	(2,3)	(5,6)
Bobot	10	15	20	25	30	35	40	45	50	55



Gambar 11: Langkah-langkah penyelesaian menggunakan algoritma Kruskal

Dan didapat hasil pohon merentang minimum sebagai berikut, dengan total nilai 10 + 25 + 15 + 20 + 35 = 105:



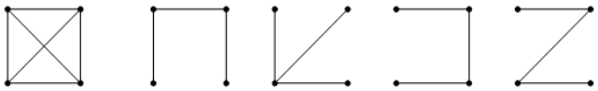
Gambar 12: Hasil pohon meerentang minimum menggunakan algoritma Kruskal

Pohon yang satu buah simpulnya diperlakukan sebagai akar dan sisi-sisinya diberi arah sehingga menjadi graf berarah dinamakan pohon berakar. Pohon berakar yang urutan anak-anaknya penting disebut pohon terurut.

III. METODE PEMECAHAN MASALAH

A. *Aplikasi Pohon Merentang untuk Mencari Jalur Terpendek/Shortest Path*

Masalah pencarian jalur terpendek/*shortest path* suatu pohon dapat juga diselesaikan dengan menggunakan pohon merentang. Untuk lebih jelasnya, diberikan ilustrasi-ilustrasi sebagai berikut:



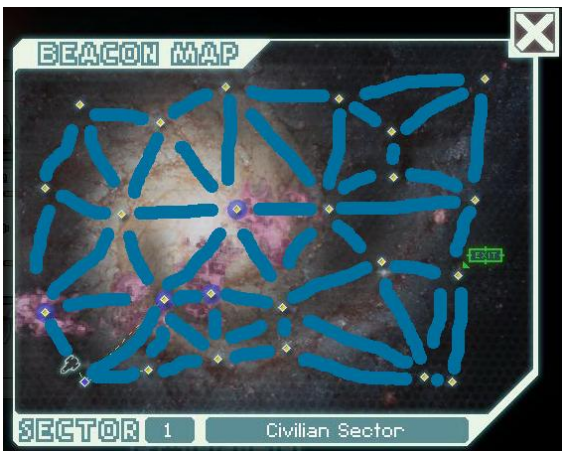
Gambar 13: Langkah menemukan berbagai jenis pohon merentang

B. *Aplikasi Pohon Merentang pada FTL*

Berikut merupakan salah satu potongan peta sektor pada game.

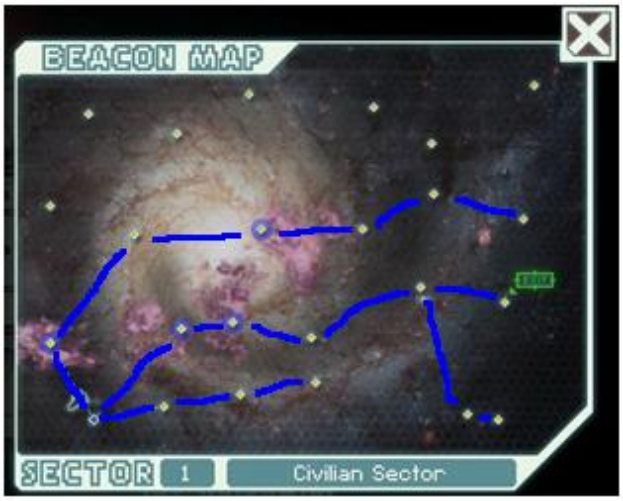
Gambar 14: Contoh Tampilan Peta Sektor 1

Kita akan menghitung jalur terpendek dari posisi kita ke simpul bertanda EXIT. Dengan asumsi kita akan mengabaikan semua simpul toko/*event* (jika ada).



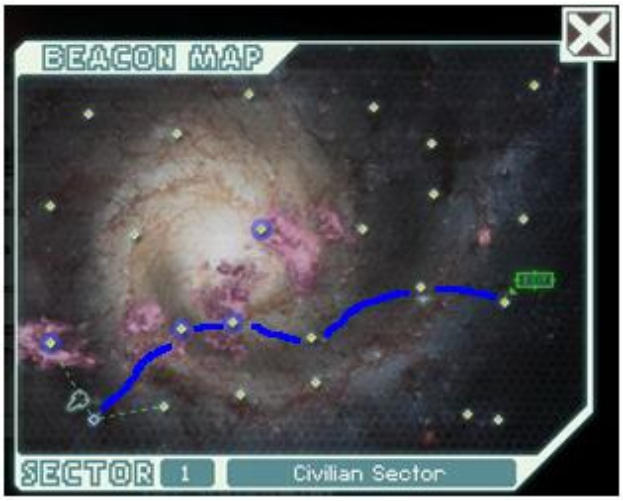
Gambar 15: Konversi Peta ke Graf

Untuk mendapatkan pohon merentang yang menuju ke arah EXIT, kita akan menghapus sirkuit yang dibentuk graf, dan meninggalkan graf-graf terdekat dengan EXIT saja.



Gambar 16: Graf setelah penghapusan

Setelah itu, kita dapat menghapus graf-graf dengan total nilai yang lebih tinggi (dengan asumsi setiap sisi bernilai 1 – diperlukan 1 poin bahan bakar untuk *warp* menuju simpul berikutnya) untuk mencari jalur mana yang dapat menghasilkan rute terpendek menuju EXIT, sehingga akhirnya didapat jalur sebagai berikut.



Gambar 13: Jarak terpendek yang didapat

IV. ANALISIS

Aplikasi pohon sangat membantu dalam pencarian jalan terpendek sebuah graf. Dengan menggunakan pohon merentang, maka kita dapat menemukan jalur terpendek suatu graf yang semua sisinya memiliki nilai sama (dalam kasus ini, semua sisi bernilai 1).

Dalam kasus ini, algoritma pencarian pohon merentang minimum seperti algoritma Prim dan Kruskal tidak dapat digunakan karena semua sisi bernilai sama, sedangkan kedua algoritma tersebut memerlukan pohon dengan nilai pada sisi berbeda-beda.

Pohon merentang dapat diaplikasikan pada permainan FTL: Faster Than Light untuk mendapat jalur terpendek untuk mencapai simpul EXIT, dengan asumsi kita mengabaikan simpul toko atau *event*.

REFERENSI

Munir, Rinaldi, Struktur Diskrit, Teknik Informatika ITB. Bandung. 2007.

http://www.gog.com/gamecard/faster_than_light
Tanggal akses 13 Desember 2012

http://www.ftlgame.com/
Tanggal akses 13 Desember 2012

http://informatika.stei.itb.ac.id/~rinaldi.munir/
Tanggal akses 14 Desember 2012

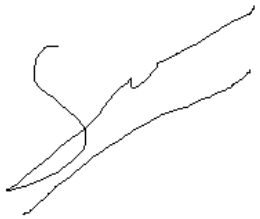
http://ftl.wikia.com/
Tanggal akses 14 Desember 2012

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, 12 Desember 2012

ttd



Sandy Gunawan Tanuwijaya
13510025