Analisis Algoritmik Fitur AutoComplete Tracks dari Video Game RollerCoaster™ Tycoon 3

Ahmad Naufal Farhan / 13513049
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13513049@std.stei.itb.ac.id

***Abstract*—RollerCoaster Tycoon 3 merupakan sebuah video game yang mensimulasikan pembangunan dan pengelolaan sebuah taman hiburan yang berisi permainan-permainan yang biasa ditemukan di taman hiburan pada umumnya. Contohnya membangun sebuah wahana *Roller Coaster*. Pada pembangunan wahana *roller coaster*, ada sebuah fitur unik pada video game ini yang dinamakan AutoComplete Tracks. Fitur ini memungkinkan pemainnya untuk membangun atau menyelesaikan sebuah *roller coaster* secara otomatis dengan hanya satu klik tombol saja. Pada makalah ini, penulis akan mencoba menjelaskan dengan menggunakan analisis algoritma Branch and Bound, bagaimana sebenarnya fitur ini dapat bekerja, dengan memperhatikan atribut dan kondisi yang ada di sekitarnya.**

***Keywords*—Branch and Bound, Fungsi Jarak, *Slope*, *Track Type*, RollerCoaster™ Tycoon 3, AutoComplete Tracks**

I. PENDAHULUAN

Seiring dengan kemajuan dan perkembangan teknologi informasi dalam kehidupan manusia, berbagai macam perangkat lunak (software) pun berkembang juga fungsinya. Para produsen perangkat lunak mulai berlomba-lomba untuk menawarkan berbagai macam fitur-fitur unik yang diusung perangkat lunaknya, untuk memenuhi kebutuhan dan kepuasan para konsumennya. Tak terkecuali pada industri video games.

Dalam konteks video game, ada beberapa hal yang memengaruhi kepuasan pemain dalam memainkan video game, diantaranya:

1. Gameplay

Gameplay yang unik dan menarik dapat membuat pemainnya memiliki rasa ketertarikan untuk terus memainkannya. Keunikan gameplay dapat ditinjau dari kepintaran Artificial Intelligence (AI), lingkungan permainan yang menarik, cerita atau storyline yang menarik untuk diikuti, atau antarmuka pengguna pada saat jalannya permainan.

2. Replay Value

Replay value dapat didefinisikan sebagai nilai kepuasan yang diperoleh dalam memainkan video game tersebut beberapa kali. Dapat didefinisikan juga sebagai "sampai berapa lama pemain akan merasa bosan memainkan game tersebut".

3. Game Display Graphics

Tampilan grafik yang memukau dari sebuah video game akan memanjakan mata pemain, sehingga pemain merasa nyaman untuk memainkannya.

Figure 1: Gameplay dan Video Quality memberikan kombinasi yang menarik untuk para pemain. (www.videogamesblogger.co)

Produsen video game berusaha untuk memajukan ketiga aspek tersebut, demi meraih perhatian pemain. Salah satunya adalah video game RollerCoaster Tycoon 3 yang dikembangkan oleh Frontier ini.

Fitur AutoComplete Tracks ini dipilih penulis karena setelah memainkan video game ini, penulis menemukan suatu relasi antar unsur pada AutoComplete Tracks ini, yang ternyata bisa dijelaskan menggunakan teori-teori yang dipelajari pada kuliah Strategi Algoritma ini.

II. DASAR TEORI

A. *Branch and Bound Algorithm*

Algoritma Branch and Bound merupakan versi lanjut (*advanced version*) dari algoritma Breadth First Search (BFS). Perbedaan yang signifikan antara BFS dengan Branch and Bound adalah, BFS menggunakan aturan First In, First Out (FIFO) untuk membangkitkan seluruh simpul yang akan dilakukan pencarian. Pada Branch and Bound, ia tidak hanya melakukan *branching* untuk setiap simpul, tetapi ia juga menghitung *cost*/ ongkos untuk pembentukan simpul tersebut, yang disebut dengan *least*

cost search. Simpul berikutnya yang akan dibangkitkan tidak lagi berdasarkan urutan pembangkitannya, tetapi berdasarkan *least cost search* yang sebelumnya telah dihitung untuk simpul yang telah dibangkitkan sebelumnya, misalkan mencari *cost* minimum untuk persoalan minimasi, atau mencari *cost* maksimum untuk persoalan maksimasi. Simpul yang memiliki *cost* terbaik yang nantinya akan di-*expand* (branch) untuk melanjutkan penyelesaian, sedangkan simpul lainnya akan dibunuh atau di-*bound*, sehingga tidak bisa dibangkitkan lagi.

Secara umum, inti dari algoritma Branch and Bound dapat diuraikan sebagai berikut:

1. Masukkan simpul akar ke dalam antrian Q. Jika simpul akar adalah simpul solusi (*goal node*), maka solusi telah ditemukan. ***Stop***.
2. Jika Q kosong, dapat disimpulkan bahwa tidak ada solusi yang memenuhi. ***Stop***.
3. Jika Q tidak kosong, pilih dari antrian Q simpul i yang mempunyai ĉ(i) paling kecil. Jika terdapat beberapa simpul i yang memenuhi, pilih satu secara sembarang.
4. Jika simpul i adalah simpul solusi, berarti solusi sudah ditemukan, stop. Jika simpul i bukan simpul solusi, maka bangkitkan semua anak-anaknya. Jika i tidak mempunyai anak, kembali ke langkah 2.
5. Untuk setiap anak j dari simpul i, hitung ĉ(j), dan masukkan semua anak-anak tersebut ke dalam Q.
6. Kembali ke langkah 2.

Dalam implementasinya, karena simpul pada *queue* akan diambil sesuai dengan nilai optimal dari *cost* untuk setiap simpul, *queue* ini dapat diimplementasikan dengan menggunakan *priority queue*.

Contoh permasalahan yang dapat diselesaikan dengan Branch and Bound adalah:

1. Travelling Salesman Problem,
2. N-Queens Problem, dan
3. 15-Puzzle Problem

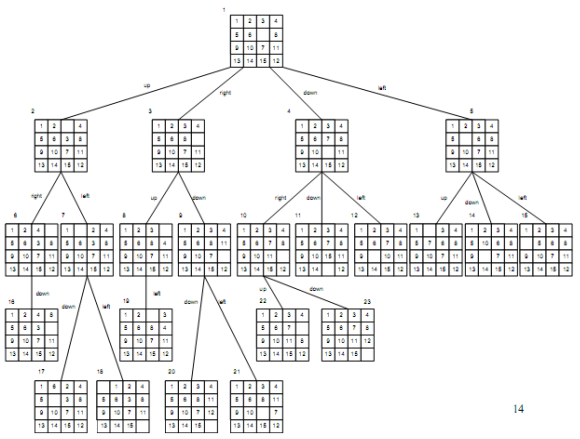

Figure 2: Penyelesaian Branch and Bound untuk 15-Puzzle Problem. [1]

Pada algoritma Branch and Bound, setidaknya ada dua hal yang menjadi basis utama dalam perancangannya, yaitu Fungsi Penghitung *Cost* Simpul dan Fungsi Pembatas (*bounding function*)

A. Menghitung Cost Simpul

Penghitungan *cost* dari sebuah simpul yang akan ditelusuri apakah simpul tersebut mengarahkan ke sebuah solusi, bergantung dari tipe permasalahan yang akan diselesaikan, dan hal apa yang diperhatikan dalam penyelesaian masalah tersebut. Jika sebuah permasalahan telah diketahui ruang status dan solusinya, biasanya *cost* dari simpul dapat berupa jarak simpul tersebut menuju simpul solusi (dalam pohon), contohnya pada persoalan *N-Queens*. Jika permasalahan belum diketahui ruang status dan solusinya, biasanya cost dapat berbentuk optimasi dari angka dan perintah yang lebih rumit. Sebagai contoh dalam persoalan TSP, yang menjadi cost untuk setiap simpulnya adalah nilai busur/jarak dari simpul satu ke simpul lainnya.

B. Fungsi Pembatas

Algoritma Branch and Bound menerapkan sebuah mekanisme yang disebut sebagai "pemangkasan" pada jalur yang dianggap sudah tidak lagi mengarah kepada solusi yang seharusnya. Dalam proses "pemangkasan" atau *bounding*, algoritma memperhatikan kriteria-kriteria suatu simpul dapat di-"pangkas", diantaranya adalah:

1. *Cost* dari simpul tersebut tidak lebih baik dibandingkan dengan *cost* yang menjadi *cost* terbaik untuk sejauh ini,
2. Simpul tersebut melanggar sejumlah *constraints* atau batasan, sehingga simpul tersebut dapat dikatakan tidak dapat merepresentasikan sebuah solusi terbaik yang cocok/*feasible*, atau
3. Solusi yang cocok/*feasible* pada simpul tersebut hanya terdiri dari *single entity* atau satu titik saja,

memungkinkan ketidakadaan pilihan lain untuk dijadikan sebuah solusi.

Fungsi pembatas / *bounding function* dapat mengarahkan program untuk mengabaikan dan tidak membangkitkan kembali fungsi yang menurutnya tidak *feasible*, dan hanya memfokuskan diri pada pencarian solusi di simpul tersebut.

III. ROLLERCOASTER TYCOON 3

RollerCoaster Tycoon 3 (RCT3) adalah sebuah video game platform PC bertema *world simulation theme park*, yang dikembangkan oleh Frontier dan dirilis oleh ATARI pada tahun 2006. Video game ini merupakan instalasi ke-3 dari serial *franchise RollerCoaster Tycoon* (sebelumnya dengan pengembang yang berbeda, Chris Sawyer dan Hasbro, merilis *RollerCoaster Tycoon* dan *RollerCoaster Tycoon 2*), dan merupakan instalasi pertama yang menggunakan 3D *rendering*, sehingga terkadang instalasi ini sering disebut sebagai *RollerCoaster Tycoon 3D*.

Seperti video game simulasi pada umumnya, dan franchise *RollerCoaster Tycoon* sebelumnya, *RCT3* tetap memfokuskan diri pada simulasi pengelolaan sebuah taman bermain (*venue park*). Pemain dapat mengelola berbagai aspek yang penting dalam mengurus sebuah taman bermain, seperti tata letak, pemandangan (*scenery*), tarif masuk, petugas pengelola, fungsi taman bemain itu sendiri dengan di dalamnya terdapat berbagai wahana yang telah dibangun oleh pemain, yang dapat dimainkan oleh pengujung taman, dan masih banyak lagi. Selain mengurus taman bermain, pemain pun dapat mengamati setiap pengunjung taman bermain dan melihat apa yang sedang dipikirkan olehnya, juga melihat status dari pengunjung itu, yang biasanya menunjukkan tingkat kelaparan, kehausan, dan kelelahan pengunjung itu.

Figure 3: Contoh gameplay dari RollerCoaster Tycoon 3. (www.ign.com)

Di tengah pengelolaan taman bermain, pemain dapat dihadapkan dengan berbagai macam masalah. Misalnya:

1. Karena taman dikunjungi banyak orang, lama kelamaan jalanan pada taman menjadi kotor dan tidak tertata rapi,
2. Beberapa wahana permainan menjadi mudah mengalami kerusakan (*broken down*) karena pemakaian dalam frekuensi yang besar,
3. Kedatangan seorang VIP, yang biasanya menuntut pengelola taman untuk menyediakan apa yang dia inginkan dengan benar, walaupun kondisi keuangan pemain sedang tidak baik, dan
4. Pengurangan jumlah pengunjung yang signifikan dikarenakan banyaknya kejahatan yang terjadi.

Untuk menanggulangi masalah tersebut, pemain dapat mempekerjakan beberapa karyawan yang akan membantu mengelola keberlangsungan taman bermainnya. Sebagai contoh, pemain dapat mempekerjakan seorang *janitor* untuk membersihkan taman, atau *engineer* untuk mengantisipasi terjadinya kerusakan pada wahana. Seringkali pemain bisa mendapatkan penghargaan untuk taman bermainnya, baik itu penghargaan yang baik ataupun yang buruk, ketika pemain dapat membuat taman bermainnya menjadi sesuatu yang baik ataupun buruk.

Fitur yang membedakan *RCT3* dengan dua instalasi lainnya adalah, pemain dapat melihat dengan jelas kondisi taman bermainnya melalui sudut pandang 3D, sehingga memungkinkan pemain untuk mengamati dengan baik semua hal yang terjadi pada layarnya. Selain itu, pemain dikenalkan dengan *peep view*, yaitu merasakan penglihatan dari sudut pandang seorang pengunjung, baik ketika ia sedang berjalan, ataupun sedang menaiki wahana permainan.

Selain fitur-fitur tadi, fitur tambahan teknis yang ada pada *RCT3* adalah kemampuan untuk menyelesaikan pembangunan *track* (lintasan) sebuah *roller coaster* dengan mudah. Fitur ini dikenal dengan sebutan AutoComplete Tracks, yang akan menjadi pusat penjelasan pada makalah ini.

A. AutoComplete Tracks

Auto-complete Tracks merupakan fitur yang mulai diperkenalkan pada video game RollerCoaster™ Tycoon 3, dan kemudian diperkenalkan kembali pada versi *mobile* dari RollerCoaster Tycoon 3D. Fitur unik ini memungkinkan para pemain untuk membangun dan menyelesaikan wahana *roller coaster* yang sedang dibangunnya hanya dengan menekan satu tombol saja. Dalam arti lain, *program* akan menyelesaikan pembuatan *roller coaster* secara otomatis dengan menempatkan beberapa *track* (lintasan).

Figure 4: Fitur AutoComplete pada gameplay RollerCoaster Tycoon 3. (koleksi pribadi)

Bagaimana caranya fitur ini bekerja? Sederhananya, fitur ini sebenarnya hanya menghubungkan satu ujung lintasan dengan ujung lintasan lainnya yang belum terhubung, dengan menggunakan beberapa pemahaman tentang lingkungan sekitarnya, yaitu:

1. Mengetahui tipe lintasan (*track type*) yang dapat digunakan dan atributnya (panjang, sudut belok, ketinggian), yang terdiri dari 3 jenis lintasan:
   a. *Straight* (lurus),
   b. *90 degree small bend* (lintasan belok kecil 90 derajat kiri dan kanan),
   c. *90 degree large bend* (lintasan belok besar 90 derajat kiri dan kanan), dan
   d. *45 degree bend* (lintasan belok 45 derajat kiri dan kanan).
2. Panjang lintasan yang akan diselesaikan,
3. Ketinggian antara satu ujung lintasan dengan ujung lintasan lainnya, dan
4. Total uang yang dimiliki oleh pemain.

Figure 5: Jenis lintasan yang disediakan permainan. (koleksi pribadi)

Dengan pengetahuan mengenai hal-hal tersebut, program dapat melakukan AutoComplete dengan tepat dan benar.

Namun, apakah semua kondisi lintasan dalam suatu lintasan *roller coaster* dapat diselesaikan dengan AutoComplete? Tentunya tidak. AutoComplete hanya dapat dilakukan apabila kondisi dari lintasan awal hingga lintasan akhir dapat ditemukan satu atau lebih solusinya oleh program, dengan menggunakan pengetahuan-pengetahuan tadi.

Ambil sebuah contoh, apabila pemain ingin melakukan AutoComplete untuk satu *roller coaster* yang dibangunnya, tetapi ia tidak memiliki cukup uang untuk menyelesaikannya, maka AutoComplete tidak akan berjalan, dan permainan akan menampilkan error bahwa AutoComplete tidak bisa dijalankan pada kondisi tersebut.

IV. PEMBAHASAN

Dengan mengetahui empat hal yang pada bab sebelumnya telah dijelaskan, program dapat menjalankan AutoComplete atau tidak menjalankannya. AutoComplete bisa tidak dijalankan karena tidak ada kondisi yang sesuai, misalkan uang pemain yang habis, jarak antar lintasan yang terlalu sempit, dan adanya perbedaan ketinggian antar lintasan yang sangat besar sehingga tidak dimungkinkan untuk membuat lintasan menurun diantara keduanya.

Penyelesaian AutoComplete Tracks dari video game *RollerCoaster Tycoon 3* dapat diuraikan dengan menggunakan algoritma Branch and Bound. Bila dimodelkan dengan model pohon, setiap simpul yang dibangkitkan adalah alternatif pemilihan lintasan, yang terdiri dari lintasan:

a. *Straight* (lurus),
b. *90 degree small bend* (lintasan belok kecil 90 derajat kiri dan kanan),
c. *90 degree large bend* (lintasan belok besar 90 derajat kiri dan kanan), dan
d. *45 degree bend* (lintasan belok 45 derajat kiri dan kanan).

Sehingga jumlah simpul yang dibangkitkan untuk satu kali pembangkitan adalah sekitar 7 simpul.

Simpul akar dari pohon merupakan lintasan awal dimana AutoComplete akan mulai berjalan, dan simpul solusi adalah lintasan akhir dari *roller coaster*, yang nantinya akan tersambung dengan lintasan hasil AutoComplete.

Bagaimana caranya memastikan bahwa Branch and Bound akan benar-benar membawa ke solusi yang benar dan tepat? Jawabannya adalah dengan menggunakan keempat karakteristik yang telah disebutkan pada bab sebelumnya. Dengan menjadikan keempat karakteristik tersebut sebagai parameter untuk penentu sebuah fungsi pembatas (sisa panjang lintasan, sisa uang, tipe lintasan, kondisi lintasan), fungsi pembatas dapat dibuat untuk memangkas arah track yang menurut kita tidak mengarah ke lintasan akhir.

Dengan mengasumsikan bahwa panjang dan lebar suatu benda direpresentasikan dengan satuan grid, anggap bahwa kita memiliki sebuah tipe bentukan bernama Tracks, dimana ia memiliki atribut-atribut sebagai berikut:

```
Tipe bentukan Tracks
  Width : integer
  Length : integer
  Start_point : Point
  End_point : Point
  Cost : integer
  OrientationLeft : true
```

Apabila kita memiliki sebuah Tracks T, yang merepresentasikan lintasan *90 degree small bend*, maka maksud dari atribut pada Tracks tersebut adalah:

1. Width: 2 (lebar dari lintasan tersebut adalah 2 grid)
2. Length: 2 (panjang dari lintasan tersebut adalah 2 grid)
3. Start_point: (1,1)
4. End_point: (2,2)
5. Cost: 20
6. OrientationLeft: true (lintasan menghadap ke sebelah kiri)

Lintasan *90 degree small bend* dapat diilustrasikan sebagai berikut (kotak gelap adalah lintasannya):

Point	1	2
1		
2		

Figure 6: Representasi 90 degree small bend dengan menggunakan sistem grid.

Lintasan *rollercoaster* dan lingkungannya yang akan dikenakan AutoComplete akan dimodelkan juga dengan satuan grid. Cara kerja AutoComplete dapat disederhanakan menjadi: bagaimana caranya menghubungkan satu lintasan awal ke lintasan akhir dengan mengisi grid-grid yang kosong, sehingga pada akhirnya lintasan awal dan lintasan akhir dapat terhubung, dengan menggunakan jumlah lintasan yang se-sedikit mungkin, dan pengeluaran biaya (*cost*) yang sekecil mungkin. Persoalan yang telah dijelaskan dapat menjadi sebuah persoalan yang dapat diselesaikan dengan menggunakan algoritma Branch and Bound, dengan meminimalisasi *cost* untuk mencapai simpul solusi.

Sebagai suatu kasus contoh, misalkan di bawah ini merupakan representasi lintasan *roller coaster* yang belum selesai, dan akan dilakukan AutoComplete terhadap lintasan tersebut, dalam bentuk grid.

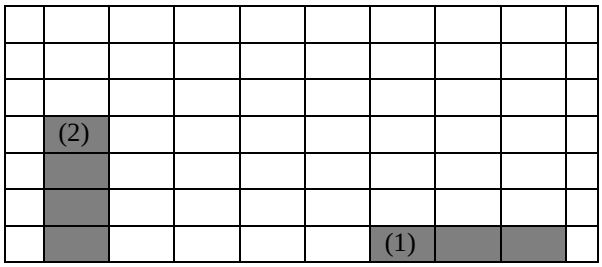

Figure 7: Representasi lintasan yang belum selesai. (1) adalah lintasan awal (simpul akar), dan (2) adalah lintasan akhir (simpul solusi).

V. KESIMPULAN

Cara kerja fitur AutoComplete dapat diimplementasikan menggunakan algoritma Branch and Bound dengan baik. Akan tetapi, untuk persoalan lintasan yang lebih kompleks lagi, seperti adanya *upward slope* (lintasan menanjak) dan *downward slope* (lintasan menurun) diperlukan algoritma yang dapat memodelkan penyelesaian persoalan lintasan tersebut dengan lebih baik menggunakan fitur AutoComplete dari permainan.

VI. UCAPAN TERIMA KASIH

Pertama, penulis ingin mengucapkan terima kasih dan puji syukur kepada Tuhan Yang Maha Pengasih, karena atas kesempatan dan kemampuan yang telah diberikan-Nya kepada penulis, sehingga penulis dapat menyelesaikan makalah ini dengan baik. Tak lupa penulis pun mengucapkan terima kasih kepada dua dosen IF2211 Strategi Algoritma, Pak Rinaldi Munir dan Ibu Nur Ulfa, atas semua ilmu dan bimbingannya selama perkuliahan di kelas maupun dalam pembuatan tugas mata kuliah ini. Penulis juga mengucapkan banyak terima kasih kepada teman-teman IF 2013, atas kerjasama dan motivasinya dalam menjalankan seluruh kewajiban kuliah penulis, termasuk tugas makalah ini.

REFERENCES

[1]Munir, Rinaldi. 2007. Diktat Kuliah IF2211. Strategi Algoritma: Bandung: Institut Teknologi Bandung.
[2] http://www.cse.ohiotate.edu/~gurari/course/cis680/cis680Ch 20.html "Branch and Bound Algorithm" diakses pada 4 Mei 2015 pukul 16.29
[3] http://www.themeparkreview.com/forum/viewtopic.php?t=17640 "What exactly do people have against RCT3 ?" diakses pada 4 Mei 2015 pukul 18.25
[4] www.frontier.co.uk/games/rollercoaster_tycoon "RollerCoaster Tycoon 3 Frontier Developments" diakses pada tanggal 4 Mei 2015 pukul 18.25

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, 6 Mei 2015

Ahmad Naufal Farhan
13513049