Penerapan Algoritma *Branch and Bound* dalam Menentukan Jalur Terpendek pada Permainan *Clash of Clans*

Naufal Malik Rabbani (13514052)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganeca No. 10, Bandung 40132
13514052@std.stei.itb.ac.id

***Abstract*—Keahlian dalam menyusun strategi sangat dibutuhkan dalam permainan *Clash of Clans*. Keahlian tersebut dapat diterapkan diantaranya pada saat pengaturan letak bangunan untuk bertahan dari serangan musuh, penentuan banyaknya jumlah pasukan yang akan dibawa untuk melawan desa lain, dan penentuan posisi serta waktu yang tepat untuk menerjunkan pasukan saat sedang berada di dalam pertarungan. Khusus untuk meletakkan pasukan saat bertarung, pemain harus memiliki strategi jitu agar pasukan bisa langsung menghancurkan desa dan merampok hartanya. Konsep algoritma *Branch and Bound* sangat dibutuhkan untuk strategi penyerangan tersebut, hal ini bertujuan agar pemain tidak meleset dalam memprediksi jalur yang akan ditempuh oleh pasukan.**

Keywords—bnb; clash of clans; coc; jalur; pertahanan; pertarungan; strategi;

I. PENDAHULUAN

Gambar 1.1 Tampilan awal *Clash of Cians*

Clash of Clans, atau yang biasa disebut COC, aplikasi besutan Supercell ini berhasil menduduki peringkat 1 sebagai permainan terpopuler dan terlaris sepanjang tahun 2015. Bagaimana tidak, menurut data yang tertampil pada Google Play Store dan Google Play Games, COC telah diunduh lebih dari 100 juta pengguna di seluruh dunia, dengan poin penilaian 4,6 dari total 5 poin, terhitung berdasarkan lebih dari 25 juta penilai.[1] Permainan bergenre strategi ini telah menghipnotis para penggunanya, dari segi tampilan, kelancaran, mode permainan berdasarkan waktu nyata (*real-time online game*), fasilitas akun premium, dan fasilitas lainnya, membuat pengguna semakin merasakan efek kecanduan terhadap permainan ini.

Banyak sekali strategi-strategi yang harus disusun oleh pemain agar desa yang dimilikinya menjadi hebat dari segi pertahanan maupun penyerangan. Diantaranya yang pertama, pemain harus memiliki strategi yang bagus untuk mengatur letak bangunan-bangunan pada desanya. Letak antar bangunan ini sangat menentukan kesuksesan dalam mempertahankan desa dari serangan lawan. *Town Hall* adalah jantung dari desa, bangunan ini harus ditempatkan pada tempat yang paling aman. *Storages* adalah sumber kekayaan desa, bangunan ini harus ditempatkan pada tempat yang aman agar penyerang sulit untuk merampoknya. *Defenses* adalah bangunan yang secara aktif mempertahankan desa saat terjadi penyerangan oleh musuh. *Traps* adalah properti yang secara pasif mambantu mempertahankan desa dari serangan musuh.

Kedua, pemain harus memiliki strategi yang bagus untuk menentukan banyaknya musuh yang akan dibawa untuk bertarung. Pemain harus bisa mendapatkan kelemahan dari susunan desa sang musuh. Selain itu pemain juga harus mengetahui karakteristik dari setiap pasukan yang akan dibawanya. Dengan kolaborasi antar jenis pasukan, pemain bisa menerobos ke tengah desa dan menghancurkannya dengan mudah. Namun jika salah dalam pemilihan pasukan yang akan dibawanya, pasukan akan dengan mudah dihalang oleh *defenses* dan *traps* desa lawan, hal ini merugikan pemain karena sudah mengeluarkan biaya untuk melatih para pasukan penyerang.

Yang terakhir adalah strategi saat penyerangan sedang berlangsung, ini merupakan strategi yang paling penting.

Pemain harus mampu menganalisis kondisi sekitar, menentukan letak yang aman serta waktu yang tepat agar pasukan tidak mati saat berhadapan dengan alat pertahanan maupun perangkap.

II. DASAR TEORI

A. *Algoritma Branch and Bound*

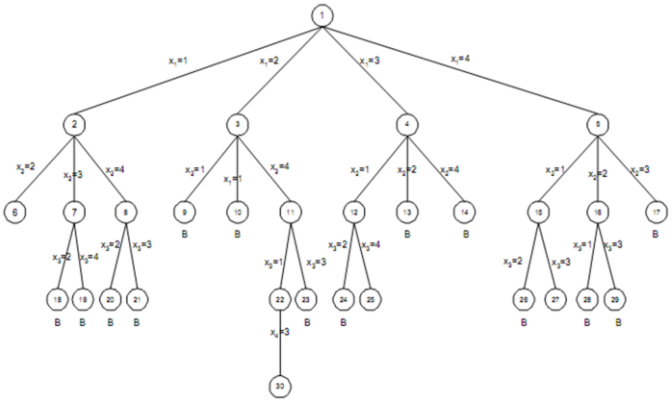
Metode *Branch and Bound* (BNB) diusulkan pertama kali oleh **A. H. Land** dan **A. G. Doig** pada tahun 1960. Sebenarnya metode ini dibuat untuk pemrograman linier (*linier programming*) baik pemrograman linier murni maupun pemrograman linier campuran. Namun kenyataanya metode ini mampu menyelesaikan permasalahan seperti TSP (*traveling salesman problem*) dan beberapa masalah lain.[2] BNB merupakan metode pencarian di dalam ruang solusi secara sistematis. Ruang solusi diorganisasikan ke dalam pohon ruang status. Ruang solusi dibangun menggunakan skema BFS, dan untuk mempercepat pencarian ke simpul sousi, maka setiap simpul diberi sebuah nilai ongkos (*cost*). Simpul berikutnya yang akan diekspansi tidak lagi berdasarkan FIFO (*first in first out*), melainkan simpul yang memiliki ongkos paling kecil. Metode ini digunakan untuk masalah optimasi (*optimazion problem*), yaitu meminimalkan atau memaksimalkan suatu fungsi objektif yang tidak melanggar batasan (*constraints*) persoalan.[3]

Nilai ongkos pada setiap simpul *i* menyatakan taksiran ongkos termurah lintasan dari simpul *i* ke simpul solusi (*goal node*).

c(i) = nilai taksiran lintasan termurah dari simpul status *i* ke status tujuan

Dengan kata lain, *c(i)* menyatakan batas bawah (*lower bound*) dari ongkos pencarian solusi dari status *i*. Ongkos ini dihitung dengan suatu funsi pembatas. Fungsi pembatas digunakan untuk membatasi pembangkitan simpul yang tidak mengarah ke simpul solusi.

Tinjau kembali persoalan **4-Ratu** dengan menggunakan metode BFS.



Gambar 2.1.1 Pohon ruang status dengan metode BFS

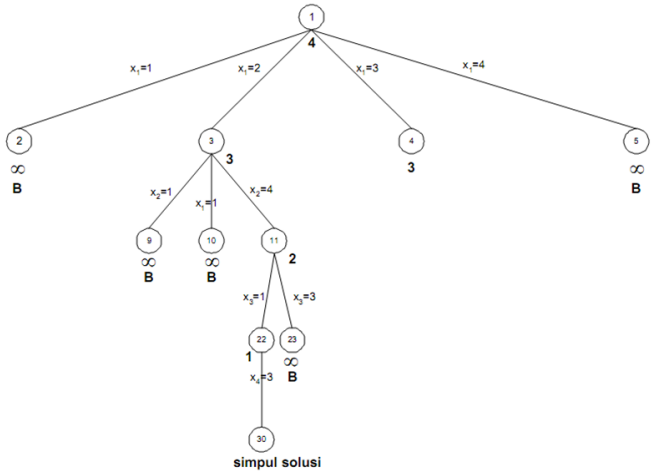
Pada algoritma BNB, pencarian ke simpul solusi dapat dipercepat dengan memilih simpul hidup berdasarkan nilai ongkos. Setiap simpul diasosiasikan dengan sebuah ongkos yang menyatakan nilai batas (*bound*). Untuk setiap simpul *X*, nilai batas ini dapat berupa :

a. Jumlah simpul dalam upapohon *X* yang perlu dibangkitkan sebelum simpul solusi ditemukan.
b. Panjang lintasan dari simpul *X* ke simpul solusi terdekat.

Misalnya digunakan ukuran (b), maka :

- Simpul akar diberi ongkos 4 (panjang lintasan dari 1 ke 30 adalah 4)
- Simpul 3 dan 4 diberi ongkos 3 (di upapohon simpul 4 juga ada simpul solusi)
- Simpul 2 dan 5 diberi nilai tak terhingga karena di upapohonnya tidak ada solusi.

Untuk memilih simpul hidup yang akan menjadi simpul ekspan (simpul-E), simpul-simpul hidup diurutkan berdasarkan nilai batasnya dari kecil ke besar. Simpul hidup yang menjadi simpul-E adalah simpul yang mempunyai nilai batas terkecil. Strategi memilih simpul-E seperti ini dinamakan strategi **pencarian berdasarkan biaya terkecil** (*least cost seacrh*).



Gambar 2.1.2 Pohon ruang status dengan metode BNB

Dengan demikian,

i. Setelah perluasan simpul 1, urutan simpul hidup adalah : 3, 4, 2, 5. Simpul-E sekarang adalah simpul 3.
ii. Simpul 3 diperluas dengan membangkitkan simpul 9, 10, dan 11. Simpul 9 dan 10 diberi ongkos tak terhingga karena di upapohonnya tidak ada simpul solusi. Simpul 11 diberi ongkos 2. Setelah perluasan simpul 3, urutan simpul hidup adalah : 11, 4, 2, 5, 9, 10. Simpul-E sekarang adalah simpul 11.
iii. Simpul 11 diperluas dengan membangkitkan simpul 22 dan 23. Simpul 22 diberi ongkos tak terhingga karena di upapohonnya tidak ada simpul solusi. Simpul 22 diberi ongkos 1. Setelah perluasan simpul 11, urutan simpul hidup adalah : 22, 4, 2, 5, 9, 10, 23. Simpul-E sekarang adalah simpul 22.
iv. Simpul 22 diperluas dengan membangkitkan simpul 30. Karena simpul 30 adalah simpul solusi, maka solusi pertama telah ditemukan.

Proses pembentukan pohon ruang status dengan algoritma BNB diperlihatkan pada Gambar 2.1.2. Mudah dilihat bahwa jika ukuran (b) digunakan maka simpul yang menjadi simpul-E adalah simpul pada lintasan dari akar ke simpul jawaban terdekat.

Pemberian nilai batas seperti pada persoalan N-Ratu di atas adalah nilai batas yang ideal, karena letak simpul solusi sudah diketahui. Pada umumnya, untuk kebanyakan persoalan, letak simpul solusi tidak diketahui, karena itu, dalam prakteknya, nilai batas untuk setiap simpul umumnya berupa taksiran atau perkiraan.

Fungsi heuristik untuk menghitung taksiran nilai tersebut dinyatakan secara umum sebagai :

$c(i) = f(i) + g(i)$

yang dalam hal ini,

$c(i)$ = ongkos untuk simpul i

$f(i)$ = ongkos dari akar mencapai simpul i

$g(i)$ = ongkos dari simpul i mencapai simpul tujuan

Nilai c digunakan untuk mengurutkan pencarian. Simpul berikutnya yang dipilih untuk diekspansi adalah simpul yang memiliki c minimum.

Algoritma BNB :

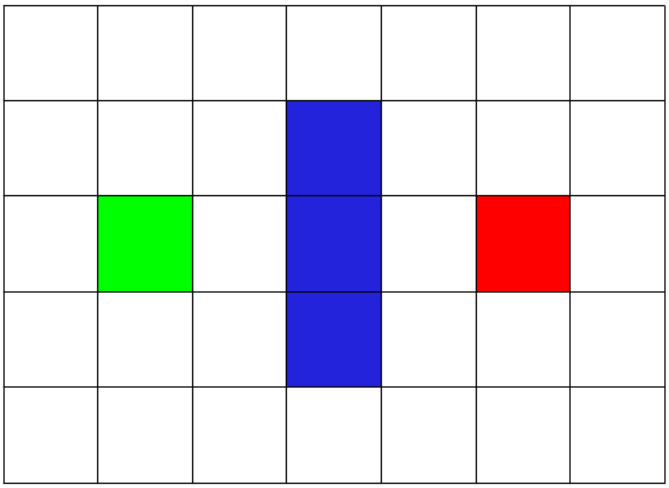
1. Masukkan simpul akar ke dalam antrian Q. Jika simpul akar adalah solusi maka berhenti.
2. Jika Q kosong maka tidak ada solusi.
3. Jika Q tidak kosong, pilih dari antrian Q simpul i yang mempunyai $c(i)$ terkecil, jika terdapat beberapa pilih salah satu secara sembarang.
4. Jika simpul i adalah solusi maka berhenti. Jika simpul i bukan solusi, maka bangkitkan anak-anaknya. Jika i tidak mempunyai anak, kembali ke poin 2.
5. Untuk setiap anak j dari simpul i, hitung $c(j)$, dan masukkan semua anak tersebut kedalam Q.
6. Kembali ke poin 2.

B. *Pencarian Jalur Terpendek*

Pencarian jalur (*pathfinding*) adalah proses mencari rute/jalur terpendek dari suatu area yang memiliki batasan-batasan (*constraint*) berupa benda-benda yang menghalangi jalur lurus antara titik awal dengan titik tujuan. Penghalang tersebut dapat berupa tembok, sungai, dan sebagainya. Tujuan dari *pathfinding* ini adalah mencari jalur **paling efisien** dengan menghindari penghalang yang ada.

Pathfinding banyak diterapkan dalam membuat AI (*artificial intellegence*) dari suatu permainan. Misalnya agar AI tersebut dapat mengejar musuh secara efisien tanpa menabrak tembok/penghalang lainnya. Terdapat beberapa metode yang sering digunakan, salah satunya adalah A*, yaitu BNB dengan *cost* dan *bound* yang sama persis.[4]

Sebagai langkah pertama untuk ilustrasi pencarian jalur ini, berikut adalah contoh sederhana area yang akan digunakan. Warna hijau adalah titik awal, warna merah adalah titik tujuan, dan warna biru adalah penghalang. Tujuan dari ilustrasi ini adalah menemukan rute dari titik hijau ke titik merah tanpa melewati penghalang biru.



Gambar 2.2.1 Arena

Fungsi heuristik untuk menghitung taksiran nilai dari area tersebut berupa :

$f(i) = g(i) + h(i)$

yang dalam hal ini,

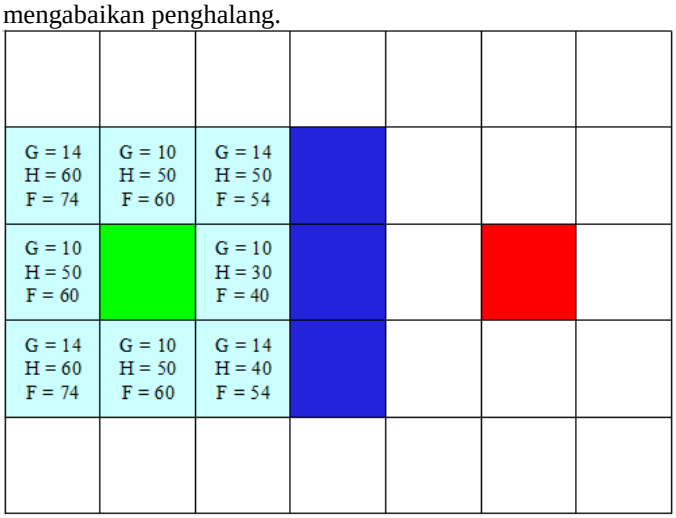
$f(i)$ = ongkos untuk titik i

$g(i)$ = ongkos dari titik awal menuju titik i

$h(i)$ = ongkos dari titik i menuju titik tujuan

Asumsikan setiap langkah dari hijau adalah legal baik secara vertikal, horizontal, maupun diagonal, dengan catatan tidak membentur penghalang. Setiap langkah yang legal berikan nilai g, 10 untuk vertikal dan horizontal, dan 14 untuk diagonal.

Untuk memberikan nilai taksiran h pada titik i, dapat diperoleh secara singkat berdasarkan pergerakan dari titik i menuju titik tujuan secara vertikal atau horizintal saja, serta mengabaikan penghalang.



Gambar 2.2.2 *cost* dan *bound* setiap titik

Berdasarkan gambar di atas, didapatkan bahwa ruang status pohon BNB akan memiliki akar berisikan titik awal (warna hijau), dengan anak-anaknya berjumlah 9 (warna biru muda). Setelah mendapatkan pohon pencarian, algoritma BNB dapat langsung diterapkan.

Ambil status-status yang hidup dengan *cost* (f) terkecil, yaitu posisi timur dari titik awal (bernilai *cost* f = 40), lakukan

terus sampai menemukan solusi (titik tujuan warna merah) atau sampai tidak menemukan solusi.

G = 14 H = 60 F = 74	G = 10 H = 50 F = 60	G = 14 H = 50 F = 54				
G = 10 H = 50 F = 60		G = 10 H = 30 F = 40				
G = 14 H = 60 F = 74	G = 10 H = 50 F = 60	G = 14 H = 40 F = 54				

Gambar 2.2.3 Langkah 1

G = 14 H = 60 F = 74	G = 10 H = 50 F = 60	G = 14 H = 50 F = 54				
G = 10 H = 50 F = 60		G = 10 H = 30 F = 40				
G = 14 H = 60 F = 74	G = 10 H = 50 F = 60	G = 14 H = 40 F = 54				
	G = 28 H = 60 F = 88	G = 24 H = 50 F = 74	G = 28 H = 40 F = 68			

Gambar 2.2.4 Langkah 2

G = 14 H = 60 F = 74	G = 10 H = 50 F = 60	G = 14 H = 50 F = 54				
G = 10 H = 50 F = 60		G = 10 H = 30 F = 40				
G = 14 H = 60 F = 74	G = 10 H = 50 F = 60	G = 14 H = 40 F = 54		G = 42 H = 20 F = 62		
	G = 28 H = 60 F = 88	G = 24 H = 50 F = 74	G = 28 H = 40 F = 68	G = 38 H = 30 F = 68		

Gambar 2.2.5 Langkah 3

G = 14 H = 60 F = 74	G = 10 H = 50 F = 60	G = 14 H = 50 F = 54				
G = 10 H = 50 F = 60		G = 10 H = 30 F = 40		G = 52 H = 10 F = 62		
G = 14 H = 60 F = 74	G = 10 H = 50 F = 60	G = 14 H = 40 F = 54		G = 42 H = 20 F = 62	G = 52 H = 10 F = 62	
	G = 28 H = 60 F = 88	G = 24 H = 50 F = 74	G = 28 H = 40 F = 68	G = 38 H = 30 F = 68		

Gambar 2.2.6 Langkah 4

Setelah algoritma BNB diterapkan, ternyata didapat solusi untuk perjalanan terdekat dari titik hijau menuju titik merah, yaitu bergerak ke arah timur, tenggara, tenggara, timur laut, lalu terakhir timur laut.

Jika kita perhatikan dengan lebih seksama, jalur lain juga akan menghasilkan rute yang sama yaitu bergerak ke arah timur, timur laut, timur laut, tenggara, lalu terakhir tenggara.

C. Karakteristik pada COC

Gambar 2.3.1 Sebuah desa dalam *Clash of Clans*

Dalam permainan COC, terdapat beberapa hal yang harus diperhatikan saat sedang melakukan penyerangan. Untuk desa yang sedang bertahan, terdapat *walls* yaitu tembok yang menghalangi pasukan musuh agar sulit menembus ke bagian tengah desa. Untuk desa yang sedang menyerang, terdapat 2 golongan yang dapat digunakan untuk menyerang. Pertama adalah *troops* yaitu pasukan yang secara aktif menyerang desa. Kedua adalah *spells* yaitu sejenis ramuan ajaib yang dapat meningkatkan efektifitas dalam penyerangan.

Berikut ditampilkan tabel karakteristik pasukan pada COC yang berpengaruh langsung dalam penentuan heuristik BNB.

Tabel 2.3.1 Karakteristik pasukan darat

Troop/Spell	Level	Damage	Movement
Barbarian	6	26	16
Archer	6	22	24
Giant	6	43	12
Goblin	6	42	32
Wall Breaker	5	46	24

Wizzard	5	170	16
P.E.K.K.A.	3	300	16
Valkyrie	2	106	24
Golem	2	42	12
Witch	1	25	12
Barbarian King	13	152	16
Archer Queen	11	204	24
Jump Spell	2	-	-

Berdasarkan nama-nama pasukan yang tertera pada tabel di atas, mereka semua adalah pasukan darat, yang menyerang dengan cara jalan kaki (kecuali Jump Spell, benda itu termasuk golongan *Spells*).

Terdapat beberapa nama pasukan lagi seperti *Balloon*, *Dragon*, *Healer*, *Minion*, *Hog Rider*, dan *Lava Hound*, tetapi mereka menyerang dengan cara terbang ataupun melompati tembok, sehingga tidak memerlukan algoritma BNB untuk mendapatkan jalur terpendeknya.

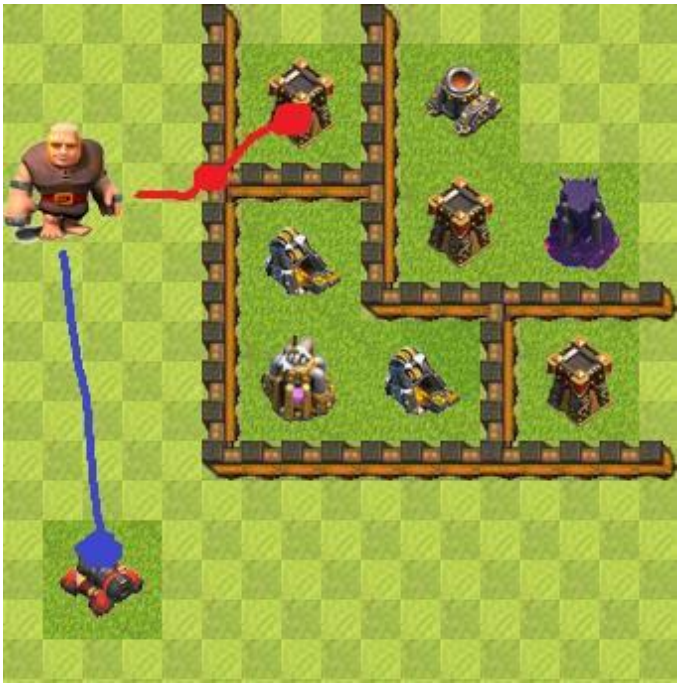
Wall Breaker sebagai salah satu golongan pasukan darat, mempunyai kemampuan khusus yaitu ia selalu menjadikan *Walls* sebagai target utamanya. Penyerangan *Wall Breaker* terhadap *Walls* menyebabkan *Walls* tersebut hancur dan pasukan lain bisa memanfaatkan lubang tembok tersebut untuk masuk lebih dalam ke bagian tengah desa. Kemampuan ini juga dimiliki oleh salah satu golongan *Spells* yaitu *Earthquake Spell*.

Selain itu, *Jump Spell* juga mempunyai efek tersendiri yaitu selama ramuan ajaib tersebut ditaburkan di atas tembok lawan, maka pasukan kita dapat melompati tembok tersbut. Namun tidak seperti *Wall Breaker* yang menghancurkan tembok secara permanen, *Jump Spell* mempunyai durasi terbatas untuk membuat tembok **seakan-akan menghilang**, sehingga bisa dilompati. Durasi tersebut bertambah seiring bertambahnya level ramuan ajaib tersebut.

III. ANALISIS DAN HASIL PENGAMATAN

Clash of Clans mempunyai area persis seperti pada contoh sebelumnya yaitu berupa petak-petak yang terdiri dari titik penerjunan pasukan (titik awal), bangunan sebagai target untuk dihancurkan (titik tujuan), dan ada juga beberapa jenis benda penghalang.

Untuk menguji coba kemampuan *artificial intellegence* dan kemampuan komputasi jalur terpendek yang terdapat di pasukan, penulis akan memberikan 2 kasus unik yang sering terjadi.



Gambar 3.1 Penyerangan oleh *Giant*

Gambar 3.1 merupakan salah satu contoh yang paling sering dialami oleh pemain, yaitu pasukan tidak mengambil jalur sesuai yang diharapkan oleh pemain.

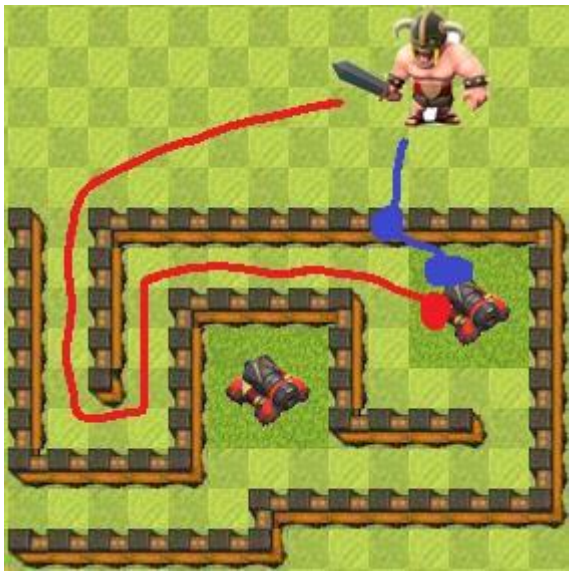
Garis berwarna merah adalah jalur yang diinginkan oleh pemain, yaitu sekumpulan pasukan *Giant* menghancurkan tembok yang ada didepannya, lalu masuk dan menghancurkan bengunan didalamnya. Namun yang terjadi adalah pasukan *Giant* bergerak ke arah selatan untuk menghancurkan bangunan yang ada di luar tembok terlebih dahulu. Hal ini dapat merugikan karena selama pasukan berjalan, pasukan akan ditembaki hingga mati ataupun terkena ranjau.

Sebagai proses komputasi pencarian jalur terpendek, saat pasukan diterjunkan, pasukan akan melakukan langkah-langkah sebagai berikut :

- Mencari target terdekat dengan persamaan pitagoras yaitu `sqrt(c) = sqrt(pow(a, 2) + pow(b, 2))`. Dimana *c* adalah jarak jarak antara titik 1 dengan titik 2, *a* adalah selisih sumbu *x* dan *b* adalah selisih sumbu adalah selisih sumbu *y* antara titik 1 dengan titik 2. Penghitungan ini dilakukan untuk ke semua target yang ada pada desa.
- Setelah mendapat beberapa target dengan jarak paling minimal, pasukan akan mencari jalur terpendek untuk menuju bangunan-bangunan tersebut, menggunakan algoritma BNB atau A*.
- Setelah mendapat jalur terpendek, tentukan 1 bangunan yang memiliki panjang lintasan terefisien berdasarkan jalur yang telah didapat.

Dalam contoh kali ini, terdapat 6 buah bangunan sebagai target, yaitu 1 buah paling luar (*cannon*), 2 buah berada di dalam 1 tembok (*archer tower*), 3 buah berada di dalam 2 tembok berlapis (*mortar, archer tower, wizzard tower*).

Dalam penghitungan target terdekat, dapat dilihat bahwa *Giant* lebih dekat menuju *archer tower* yang ada di dalam tembok. Namun karena hasil dari algoritma BNB menuju *cannon* **lebih efisien** dari pada yang menuju *archer tower*, maka *Giant* bergerak untuk menghancurkan *cannon* terlebih dahulu, baru kemudian menghancurkan tembok dan *archer tower* yang ada di dalamnya.



Gambar 3.2 Penyerangan oleh *Barbarian*

Contoh kasus kedua adalah kasus yang bisa jadi menguntungkan pemain, karena bisa mendapatkan jalur yang lebih singkat berkat kemampuan AI dari pasukan yang canggih.

Hal ini ditunjukkan bahwa pada kenyataannya pasukan tidak mengambil jalur memutar (garis merah) namun menghancurkan tembok langsung untuk masuk ke dalam (garis biru). Tentunya hal ini sudah diperhitungkan dengan matang oleh AI pasukan, terutama dalam kekuatan untuk menghancurkan tembok.

Untuk lebih jelasnya, setiap pasukan mempunyai kekuatan masing-masing (*damage*), apakah jika menghancurkan tembok dengan kekuatan tersebut memakan waktu lebih lama, dibandingkan berjalan memutar? Jika tidak, maka pengambilan keputusan untuk menghancurkan tembok sudah tepat.

Sebagai hasil pengamatan, penulis menduga bahwa semua komputasi pencarian jalur terpendek dilakukan oleh *server*. Karena jika hal ini dibebankan ke perangkat pemain, tentu akan membuat kinerja menurun. Mengingat banyaknya proses AI untuk **setiap pasukan** *cross section* dengan **setiap target**.

IV. KESIMPULAN

Dalam membangun *artificial intellegence* yang ditanamkan pada setiap pasukan, COC telah memberikan algoritma dan komputasi khusus terutama dalam penentuan parameter fungsi *cost* dan *bound*. Parameter-parameter tersebut sangat bergantung pada banyak hal diantara lain penempatan tembok, kekuatan dari pasukan tersebut, dan jumlah pasukan yang diterjunkan apakah ramai-ramai atau individu untuk mendapatkan total kekuatan. Sebagai contoh, tembok bukanlah sebuah benda penghalang, namun area di dalam tembok memiliki heuristik nilai taksiran yang sangat besar.

Artificial intellegence pada COC juga dibekali dengan kemampuan untuk melihat kondisi sekitar setiap saat. Hasilnya adalah setiap ada *event* yang terjadi, COC akan menghitung ulang jalur terpendek pada pasukan-pasukannya. Misalnya, saat *Wall Breaker* berhasil membobol salah satu bagian tembok, maka pasukan lain langsung menghitung ulang *cost* dan *bound* jika melewati lubang tembok tersebut. Begitu juga dengan *event* saat ramuan ajaib *Jump Spell* ditaburkan.

V. UCAPAN TERIMA KASIH

Pertama-tama penulis ingin mengucapkan Alhamdulillaahi rabbil ‘aalamiin, karena rahmat-Nya yang selalu menyertai penulis hingga pembuatan makalah ini selesai. Penulis juga ingin mengucapkan terima kasih kepada orang tua yang selalu mendukung dan memotivasi penulis dalam perkuliahan di ITB. Penulis juga ingin mengucapkan terima kasih kepada Bu Ulfa dan Pak Rinaldi selaku dosen pembimbing mata kuliah IF2211 Strategi Algoritma yang telah menuangkan segala ilmu berkaitan dengan strategi algoritma ini, khususnya algoritma *Branch and Bound*. Semoga kita semua tetap selalu dalam limpahan rahmat Allah subhaanahu wa ta’ala, aamiin.

REFERENSI

[1] https://play.google.com/store/apps/details?id=com.supercell.clashofclans (diakses pada 7 Mei 2016 pukul 16:00 WIB)

[2] http://trisadinidaswan.blogspot.co.id/2012/01/algoritmabranch-and-bound-metode-branch.html (diakses pada 7 Mei 2016 pukul 23:59 WIB)

[3] Diktat Kuliah IF2211 Strategi Algoritma, Dr. Ir. Rinaldi Munir, M.T.

[4] http://duniadigit.blogspot.co.id/2013/08/belajar-algoritma-untuk-pencarian-jalur.html (diakses pada 8 Mei 2016 pukul 01:43 WIB)

[5] http://www.clashofclans-tools.com/Layout-Builder (diakses pada 8 Mei 2016 pukul 08:00 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, 8 Mei 2015

Naufal Malik Rabbani (13514052)