Penerapan Algoritma Greedy Untuk Memenangkan Permainan 'Go'

Innani Yudho'afi (13511054)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
innani.yudhoafi@students.itb.ac.id

***Abstract*—Pada makalah ini dibahas mengenai algoritma Greedy yang digunakan untuk memenangkan permainan go melawan komputer go. Terdapat pula informasi tentang permainan go dan penjelasan dari implementasi algoritma Greedy dalam menyelesaikan permainan ini. Pada akhir makalah akan disimpulkan apakah menyelesaikan permainan go dengan menggunakan algoritma Greedy merupakan cara yang cukup efektif dan menghasilkan hasil yang diharapkan.**

***Index Terms*—Algoritma Greedy, permainan go, score maksimum.**

I. PENDAHULUAN

Permainan go merupakan permainan yang sudah ada sejak ribuan tahun yang lalu dan sampai sekarang pun masih banyak orang memainkannya. Bahkan, permainan go ini telah dijadikan games pada berbagai platform, misalnya android. Peraturan permainan yang unik dan tingkat kesulitan yang lumayan tinggi, membuat go menjadi salah satu permainan yang digemari karena cukup menantang sehingga tidak membosankan. Pemain dari permainan ini melawan komputer go, dan biasanya pemain hanya mengandalkan pengalaman dan intuisinya demi memenangkan permainan ini. Menyelesaikan permainan go akan lebih efektif jika menggunakan algoritma tertentu, misalnya Greedy.

II. DASAR TEORI

A. Algoritma Greedy

Sebuah algoritma Greedy adalah algoritma yang mengikuti pemecahan masalah heuristik membuat pilihan yang optimal secara lokal pada setiap tahap dengan harapan menemukan optimum global. Dalam banyak masalah, strategi Greedy tidak secara umum menghasilkan solusi optimal, tapi tetap heuristik Greedy dapat menghasilkan solusi optimal lokal yang mendekati solusi optimal global dalam waktu yang wajar.

Sebagai contoh, strategi Greedy untuk masalah salesman keliling (yang dari kompleksitas komputasi tinggi) adalah heuristik berikut: "Pada setiap tahap mengunjungi kota yang belum dikunjungi terdekat ke kota saat ini". Heuristik ini tidak perlu menemukan solusi terbaik tetapi berakhir dalam jumlah yang masuk akal langkah, mencari solusi optimal biasanya memerlukan banyak langkah masuk akal. Dalam optimasi matematika, algoritma Greedy memecahkan masalah kombinatorial yang memiliki sifat-sifat matroids.

Secara umum, algoritma Greedy memiliki lima komponen:

1. Satu set kandidat, dari mana solusi yang dibuat
2. Sebuah fungsi seleksi, yang memilih kandidat terbaik yang akan ditambahkan ke larutan
3. Fungsi kelayakan, yang digunakan untuk menentukan apakah seorang kandidat dapat digunakan untuk berkontribusi pada solusi
4. Fungsi tujuan, yang memberikan nilai pada solusi, atau solusi parsial, dan
5. Fungsi solusi, yang akan menunjukkan bila kita telah menemukan solusi lengkap

Algoritma Greedy menghasilkan solusi yang baik pada beberapa masalah matematika, tapi tidak pada orang lain. Sebagian besar masalah yang mereka bekerja, akan memiliki dua sifat :

Properti pilihan Greedy

Kita bisa membuat pilihan apa pun yang tampaknya terbaik saat ini dan kemudian memecahkan submasalah yang timbul kemudian. Pilihan yang dibuat oleh sebuah algoritma yang tamak mungkin tergantung pada pilihan yang dibuat sejauh ini tetapi tidak pada pilihan masa depan atau semua solusi untuk subproblem tersebut. Ini iteratif membuat satu pilihan Greedy demi satu, mengurangi setiap soal yang diberikan menjadi lebih kecil. Dengan kata lain, algoritma greedy tidak pernah mempertimbangkan kembali pilihannya. Ini adalah perbedaan utama dari pemrograman dinamis, yang lengkap dan dijamin untuk menemukan solusinya. Setelah setiap tahap, program dinamis membuat keputusan berdasarkan semua keputusan yang dibuat pada tahap sebelumnya, dan mungkin kembali jalur algoritmik tahap sebelumnya untuk solusi.

substruktur optimal

"Sebuah masalah menunjukkan substruktur optimal jika solusi optimal untuk masalah ini memuat solusi optimal untuk sub - masalah."

Contoh tentang bagaimana algoritma Greedy mungkin gagal untuk mencapai solusi optimal.

Mulai dari A, sebuah algoritma yang tamak akan menemukan maksimum lokal di "m", tidak menyadari maksimum global pada " M ".

Dengan tujuan mencapai terbesar -sum, pada setiap langkah, algoritma Greedy akan memilih apa yang tampaknya menjadi pilihan yang langsung optimal, sehingga akan memilih 12 bukan 3 pada langkah kedua, dan tidak akan mencapai solusi terbaik, yang berisi 99.
Bagi banyak masalah lain, algoritma Greedy gagal untuk menghasilkan solusi optimal, dan bahkan dapat menghasilkan solusi yang unik terburuk mungkin. Salah satu contoh adalah masalah salesman keliling yang disebutkan di atas. Untuk setiap nomor dari kota, ada tugas dari jarak antara kota-kota yang tetangga terdekat heuristic menghasilkan kemungkinan tur terburuk yang unik.

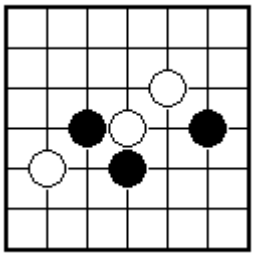
Bayangkan contoh koin dengan hanya 25 sen, 10 sen, dan koin 4 sen. Algoritma greedy tidak akan mampu untuk membuat perubahan untuk 41 sen, karena setelah melakukan menggunakan satu koin 25 sen dan satu koin 10 sen mustahil untuk menggunakan koin 4 sen untuk keseimbangan 6 sen, sedangkan seseorang atau algoritma yang lebih canggih bisa membuat perubahan bagi 41 sen dengan satu koin 25 sen dan empat koin 4 sen.

B. Games Go Dan Aturannya

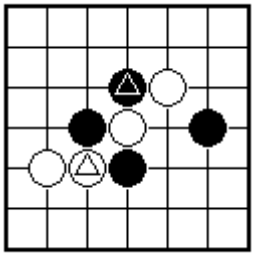
Go (Chinese : Weiqi, Jepang : igo, Korean : baduk, Vietnam : CO Vay, makna umum : " mengelilingi game") adalah permainan papan untuk dua pemain yang berasal dari Cina lebih dari 2.500 tahun yang lalu. Permainan ini terkenal karena menjadi kaya dalam strategi meskipun aturan yang relatif sederhana.

Kedua pemain bergantian menempatkan potongan hitam dan putih bermain, yang disebut "batu", di persimpangan kosong (disebut "titik") dari grid 19 × 19 baris (pemula sering bermain di kecil 9 × 9 dan 13 × 13 papan). Tujuan dari permainan ini adalah untuk menggunakan batu seseorang untuk mengelilingi total area yang lebih besar dari papan daripada lawan. Setelah ditempatkan pada papan, batu tidak dapat bergerak, tapi batu-batu yang dihapus dari papan jika tertangkap, hal ini dilakukan dengan mengelilingi batu menentang atau kelompok batu dengan menduduki poin semua ortogonal - berdekatan pemain melanjutkan dengan cara ini sampai pemain tidak ingin pindah lagi, permainan memiliki set kondisi berakhir. Ketika permainan menyimpulkan, poin dikendalikan (wilayah) dihitung bersama dengan batu-batu yang diambil untuk menentukan siapa yang memiliki poin lebih. Permainan juga dapat dimenangkan oleh pengunduran diri. Penjelasan permainan dan aturannya lebih rinci akan dijelaskan dengan menggunakan contoh kasus berikut.

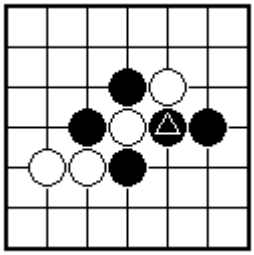
Posisi setelah beberapa drama pada grid kecil mungkin terlihat seperti ini:

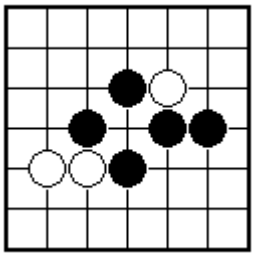


Sampai titik ini dalam permainan belum ada kesempatan untuk menangkap. Setelah Hitam and Putih telah menambahkan batu dengan segitiga pada mereka, kita akan mendapatkan ini:



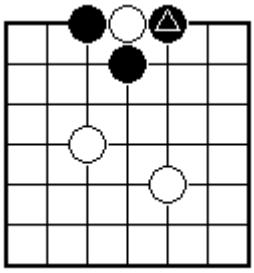
Di mana Hitam memiliki kesempatan untuk menangkap batu putih pada titik tengah. Itu terjadi seperti ini:

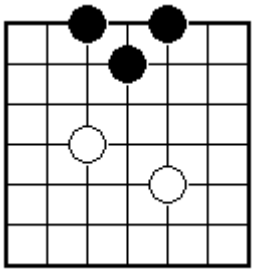




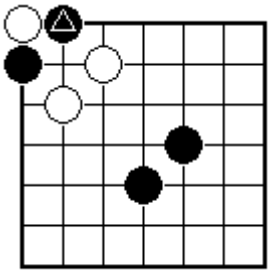
Hitam menempatkan batu yang benar-benar mengelilingi batu putih tengah, dan kemudian menghapusnya dari papan.

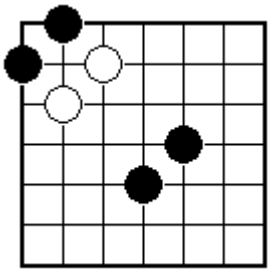
Aturan untuk menangkap satu batu karena itu dinyatakan seperti ini : ketika seorang pemain menempatkan batu yang menyebabkan batu musuh dikelilingi empat sisi (di tengah papan), batu yang diambil dari. Aturan ini meluas ke tepi batu yang menjadi dikelilingi pada tiga sisi:



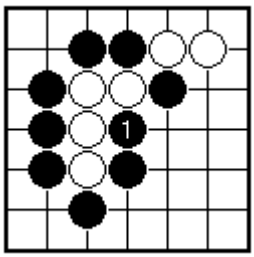


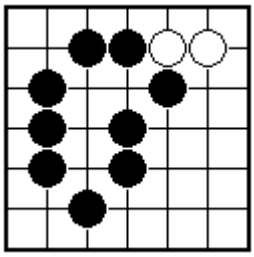
dan batu sudut dikelilingi di dua sisi:





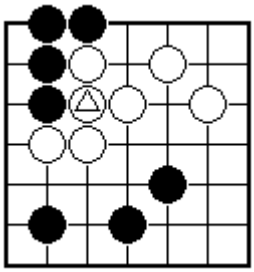
Itu tidak semua, tentu saja. Untuk membuat permainan yang menarik Anda harus memungkinkan untuk menangkap unit yang lebih besar. Yang biasanya dijelaskan dengan mengatakan bahwa Anda dapat menangkap rantai solid-terhubung batu musuh dengan mengisi titik kosong yang berdekatan akhir, seperti ini:

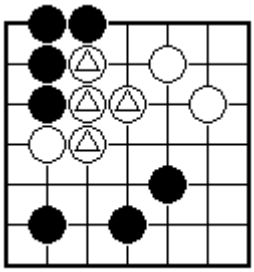


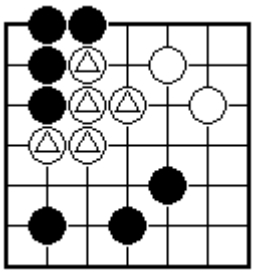


di sini empat batu putih ditangkap oleh Hitam.

Untuk mengatakan ini lebih hati-hati: hubungan yang terhubung berlaku pertama yang titik yang berdekatan dari grid, dianggap sebagai suatu jaringan, yang ditempati oleh batu dengan warna yang sama. Dan kemudian satu memperkenalkan konsep rantai karena semua batu dari satu warna yang satu mencapai dengan memulai di batu di papan tulis, dan bekerja ke luar melalui batu tetangga yang terhubung, dan tetangga mereka terhubung, sampai semua batu termasuk yang menghubungkan 'bersama garis' dengan yang awal. Sebagai contoh:







rantai termasuk batu putih segitiga ditandai diakui dalam tiga langkah.

Rantai ini mungkin besar atau hanya sebuah batu yang terisolasi, mereka juga mungkin memiliki bentuk yang sangat bervariasi (sebanyak ada polyominoes, Anda bisa mengatakan, yang cukup bagi siapa pun). Keuntungan untuk berbicara tentang rantai untuk aturan Go adalah bahwa penangkapan dapat dijelaskan secara sederhana. Setiap rantai di papan akan memiliki minimal satu titik

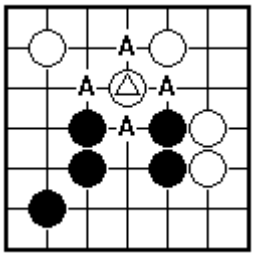
kosong berdekatan dengan itu : poin tersebut disebut dengan 'kebebasan'. Sebuah rantai ditangkap saat lawan mengisi kebebasan akhir. Tidak pernah ada rantai tanpa kebebasan pada papan di akhir giliran - posisi dengan rantai libertyless tidak dapat terjadi sebagai akibat dari bermain hukum.

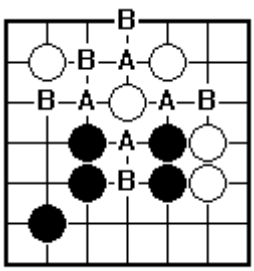
Dasar dari menangkap aturan Go baru-baru ini telah digunakan dalam papan permainan tentang gangster mengambil alih sebuah kota dibagi menjadi blok, seperti New York. Ketika geng mengambil semua blok yang mengelilingi daerah, dibutuhkan itu juga.

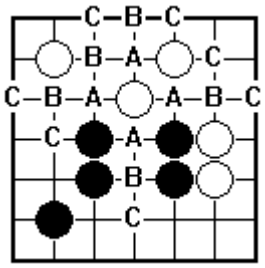
Ide koneksi juga dapat digunakan untuk memperjelas jauh lebih berduri pertanyaan : wilayah sekitarnya. Selama permainan pemain mencoba untuk dinding dari bagian papan sebagai mereka sendiri : skor akhir tergantung pada wilayah tersebut. Salah satu cara untuk mengenali wilayah 'saya' adalah seperti ini : jika lawan saya bermain di dalamnya, saya akan dapat menangkap semua batu menyerang seperti pada waktunya. Bagaimana hal ini berdamai dengan ide lebih intuitif bahwa area berdinding - off memiliki 'dalam' dan ' luar ' ? Ini adalah sesuatu yang sering menyebabkan masalah untuk pemain pemula.

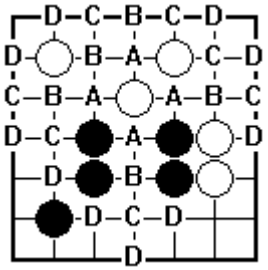
Salah satu cara untuk mengatasi masalah ini adalah dengan menggunakan ide koneksi dengan cara yang mungkin tidak begitu akrab, tapi secara matematis tidak lebih. Dalam ilmu komputer itu disebut 'banjir'. Hal ini lebih menarik untuk berpikir potongan merah dan biru daripada batu hitam dan putih, mungkin. Bayangkan bahwa potongan-potongan merah dapat membanjiri keluar air merah, dan potongan biru air biru. Air dapat dijalankan di mana saja di sepanjang saluran antara titik kosong, tapi diblokir oleh poin yang diduduki.

Diagram ini banjir dari satu batu ditandai akan memberikan ide :









poin ditandai A dibanjiri dalam satu langkah, B poin dari titik A, dan seterusnya.

Maka Anda tidak memerlukan gagasan 'dalam' dan 'luar' atau 'sekitarnya' untuk menjelaskan wilayah di Go. Titik A adalah wilayah tak terbantahkan untuk Putih dalam posisi papan diberikan jika dapat dicapai dengan banjir dari batu putih, tetapi dari batu hitam; andvice versa untuk wilayah tak terbantahkan untuk Hitam. Ini adalah definisi yang sistematis yang cukup baik, tidak perlu ada masukan intuitif, seperti yang Anda harapkan dari sudut pandang komputasi. Itu membuat dipahami ide tentang 'menghubungkan melalui daerah kosong', yang banyak orang menemukan off-menempatkan pada pertemuan pertama.

Ini bukan bagaimana pemain Go biasanya berpikir tentang hal itu. Wilayah adalah daerah seperti benteng dewan yang off-batas untuk penyusup, dan gagasan yang diusulkan wilayah hanya bekerja ketika semua operasi-operasi pembersihan telah menyimpulkan. Juga wilayah hanya dihitung pada akhir permainan. Permainan Go selesai ketika kedua pemain senang bahwa tidak ada yang lebih berharga untuk dilakukan, dan semua orang mengepel-up bergerak akan memperpanjang masalah (pemain berpengalaman hampir selalu membawa mereka sebagai read).

Aspek strategis dasar meliputi :

- Koneksi: Menjaga batu sendiri terhubung berarti bahwa kelompok-kelompok lebih sedikit perlu membuat bentuk hidup, dan satu kelompok memiliki lebih sedikit untuk membela.
- Potong : Menjaga menentang batu terputus berarti lawan harus mempertahankan dan membuat bentuk hidup untuk lebih kelompok.
- Tetap hidup : Cara termudah untuk tetap hidup adalah untuk membangun pijakan di sudut atau di sepanjang salah satu sisi. Minimal, kelompok harus memiliki dua mata (titik terbuka terpisah) untuk menjadi " hidup ". Lawan tidak dapat mengisi salah

satu mata, karena setiap langkah tersebut adalah bunuh diri dan dilarang dalam aturan.

- Kehidupan Mutual lebih baik daripada mati : Situasi di mana pemain tidak dapat bermain pada titik tertentu tanpa kemudian memungkinkan pemain lain untuk bermain di titik lain untuk menangkap. Contoh yang paling umum adalah bahwa dari kelompok yang berdekatan yang berbagi mereka beberapa terakhir kebebasan - jika salah satu pemain bermain di kebebasan bersama, mereka dapat mengurangi kelompok mereka sendiri untuk kebebasan tunggal (menempatkan diri mereka di atari), yang memungkinkan lawan mereka untuk menangkap itu di langkah selanjutnya.
- Kematian : Sebuah kelompok yang tidak memiliki bentuk hidup (yang berarti satu dengan kurang dari dua mata) akhirnya dihapus dari papan seperti yang ditangkap.
- Invasion : Membentuk kelompok hidup baru di dalam suatu wilayah di mana lawan memiliki pengaruh yang lebih besar, berarti satu mengurangi lawan mencetak gol dalam proporsi ke daerah satu menempati.
- Pengurangan : Menempatkan batu cukup jauh ke daerah lawan pengaruh untuk mengurangi jumlah wilayah mereka akhirnya mendapatkan, tapi tidak begitu jauh bahwa hal itu dapat terputus dari batu ramah luar.
- Sente : Sebuah permainan yang memaksa lawan seseorang untuk merespon (Gote), seperti menempatkan kelompok lawan di atari (bahaya penangkapan). Seorang pemain yang bisa bermain secara teratur sente memiliki inisiatif, seperti dalam catur, dan dapat mengontrol aliran permainan.
- Pengorbanan : Membiarkan kelompok untuk mati dalam rangka untuk melaksanakan bermain, atau rencana, di daerah yang lebih penting.

Cara perhitungan score (nilai) :

Setelah kedua pemain telah berlalu berurutan, batu-batu yang masih di papan tetapi tidak mampu menghindari penangkapan, yang disebut batu mati, akan dihapus.

Scoring daerah (termasuk China) : skor Seorang pemain adalah jumlah batu yang telah di papan, ditambah jumlah persimpangan kosong yang dikelilingi oleh batu-batu yang pemain.

Scoring Territory (termasuk Jepang dan Korea) : Dalam perjalanan dari permainan, setiap pemain mempertahankan batu mereka menangkap, disebut tahanan. Setiap batu mati dihapus pada akhir pertandingan menjadi tahanan. Skor tersebut adalah jumlah titik kosong tertutup oleh batu pemain, ditambah jumlah tahanan ditangkap oleh pemain itu.

Jika ada ketidaksepakatan tentang yang batu mati, maka di bawah aturan scoring area, para pemain hanya melanjutkan bermain untuk menyelesaikan masalah ini. Skor tersebut dihitung dengan menggunakan posisi setelah waktu berikutnya para pemain lulus berturut-turut. Di bawah wilayah penilaian, aturan yang jauh lebih kompleks, namun, dalam prakteknya, pemain umumnya bermain di, dan, setelah status masing-masing batu telah ditentukan, kembali ke posisi pada saat pertama dua melewati berturut-turut terjadi dan menghapus mati batu. Untuk informasi lebih lanjut, lihat Aturan Go.

Mengingat bahwa jumlah batu pemain memiliki di papan secara langsung berkaitan dengan jumlah tahanan lawan mereka telah diambil, skor bersih yang dihasilkan, yaitu perbedaan antara Hitam dan skor Putih, identik bawah kedua set aturan (kecuali pemain memiliki melewati nomor yang berbeda kali selama permainan). Dengan demikian, hasil bersih yang diberikan oleh dua sistem penilaian jarang berbeda lebih dari satu titik.

III. IMPLEMENTASI DAN ANALISIS

Strategi Greedy yang digunakan dalam menyelesaikan permainan go adalah memilih dimana batu akan diletakkan pada papan sehingga jumlah daerah batu tersebut lebih banyak dari pada jumlah daerah batu lawan, dan diambil maksimal dari semua kemungkinan tersebut. Jadi, diusahakan pemain menang di setiap langkahnya. Tetapi jika tidak ada tempat yang memungkinkan untuk pemain mendapat score lebih tinggi daripada lawan, maka penempatan batu pada papan sehingga score yang di dapat pemain paling tinggi dibandingkan dengan posisi lain di papan.

Misalkan papan go berukuran n x n, berarti terdapat $(n+1)^2$ buah titik, dengan setiap titiknya direpresentasikan dengan X_{ij}, i merepresentasikan baris dan j merepresentasikan kolom. Misal, pemain dengan batu putih dan lawan dengan batu hitam.

1. Himpunan kandidat : semua titik di papan yang masih kosong.
2. Fungsi seleksi : pilihlah sebuah titik pada papan yang masih available untuk batu tersebut, sehingga menghasilkan score maksimum untuk pemain.
3. Fungsi kelayakan : apakah titik tersebut available (bukan merupakan one-eye).
4. Fungsi objektif : score yang dihasilkan maksimum.
5. Fungsi solusi : Sebuah titik yang menghasilkan score tertinggi yang mungkin dicapai oleh pemain (jumlah daerah putih lebih benyak daripada hitam).

Pseudocode pada setiap langkah, asumsi pemain menggunakan batu putih :

```
procedure greedyGo
        int max <- 0
        int scoreHitam <- scoreHitamPrev
        int scorePutih <- scorePutihPrev
        int n <- ukuran papan + 1
        boolean bool <- false  // kalah
        for i=0 to n do
                for j=0 to n do
```

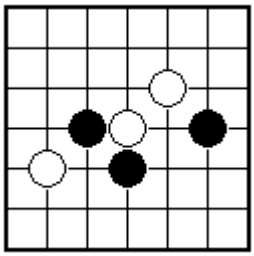
```
                    if cekAvailable(X[i][j])
                    then
                    // cekAvailable()
menghasilkan nilai true jika Xij available untuk
ditempati oleh batu putih
                        hitungScore()
                        // hitungScore()
merupakan prosedur untuk menghitung score,
variable scoreHitam dan scorePutih akan berisi
nilai perhitungan nilai permainan go
                        if scorePutih >
max then
                            max <-
scorePutih
      if scorePutih >= scoreHitam then
            bool <- true    // menang
```

Kompleksitas algoritma => O(n)

Contoh kasus :

Misalkan papan go berukuran 6x6.

Setelah beberapa saat permainan, kondisi papan seperti di bawah ini :



Misalkan saat ini adalah giliran pemain dengan batu putih untuk main. Misalkan setelah dijalankan procedure greedyGo di atas, nilai maksimum score didapat pemain jika batu putih terletak pada titik X_{53} (baris 5 kolom 3). Maka batu putih diletakkan pada koordinat tersebut. Lalu lanjutkan permainan hingga papan terisi penuh atau salah satu pemain menyerah.

Hasil dari contoh kasus ini tidak dapat diketahui, karena tidak dibuat program untuk implementasi algoritma greedy ini.

Menyelesaikan permainan go dengan menggunakan algoritma greedy yang telah didefinisikan di atas tidak dapat dijamin selalu menghasilkan score yang maksimum atau menyebabkan pemain memenangkan permainan.

IV. KESIMPULAN

Penggunaan algoritma greedy untuk menyelesaikan permainan go tidak selalu menghasilkan hasil optimum atau menghasilkan status menang bagi pemain. Ini karena, di setiap langkah permainan, pemain tidak selalu mendapatkan state menang.

Jika algoritma greedy digabungkan dengan strategi pada poin bab 2 bagian B, membuat kemungkinan atau peluang pemain menang menjadi lebih besar.

REFERENSI

[1] http://senseis.xmp.net (waktu akses : 21 Desember 2013 10.00 WIB)

[2] http://nrich.maths.org/1433 (waktu akses : 21 Desember 2013 11.00 WIB)

[3] http://www.princeton.edu/~achaney/tmve/wiki100k/docs/Breadth-first_search.html (waktu akses : 21 Desember 2013 13.00 WIB)

[4] Introduction to Algorithms (Cormen, Leiserson, and Rivest) 1990, Chapter 17 "Greedy Algorithms" p. 329.

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, 20 Desember 2013

ttd

Innani Yudho'afi
13511054