Penerapan Algoritma Greedy pada Permainan Checker

Ahmad Mutawalli – 13517026
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13517026@std.stei.itb.ac.id

Abstrak—Algoritma greedy merupakan salah satu algoritma yang popular untuk memecahkan persoalan optimasi. Greedy yang berarti tamak adalah algoritma yang akan mengambil sebuah keputusan terbaik pada suatu state tanpa bisa kembali ke state sebelumnya apabila ternyata keputusan yang sebelumnya sudah diambil bukan merupakan keputusan optimum global. Oleh karena itu, hasil dari algoritma greedy merupakan keputusan optimum lokal dengan harapan keputusan tersebut akan menghasilkan keputusan yang optimum global.

Permainan Checker adalah sejenis permainan papan yang dimainkan diatas papan berukuran 8x8 kotak dengan 12 buah bidak di masing-masing pihak yang hanya diizinkan melangkah. Seperti halnya permainan papan lainnya, Checker dimainkan oleh dua orang yang saling berhadapan-hadapan dengan sebuah papan permainan seperti catur dengan melangkah bergantian.

Makalah ini akan membahas penggunaan algoritma greedy dalam permainan greedy untuk menghabiskan semua bidak musuh.

Kata Kunci*; greedy, checker,* optimasi

I. PENDAHULUAN

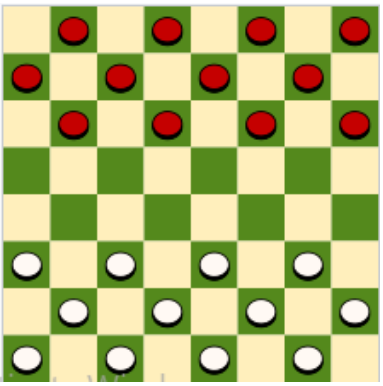
Permainan Checker adalah sejenis permainan papan yang dimainkan diatas papan berukuran 8x8 kotak dengan 12 buah bidak di masing-masing pihak yang hanya diizinkan sekali melangkah. Seperti halnya permainan papan lainnya, Checker dimainkan oleh dua orang yang saling berhadapan-hadapan dengan sebuah papan permainan seperti catur dengan melangkah bergantian.

Pemain yang satu memainkan buah bewarna hitam, yang lainnya bewarna putih. Bidak biasanya dibuat dari kayu, berbentuk bulat dan pipih. Bidak biasanya dibagi menjadi bidak yang bewarna gelap dan yang lebih terang untuk membedakan. Biasanya, warna-warnanya merah dan putih. Ada dua jenis bidak yaitu “prajurit” dan “ratu”. “Ratu” dibedakan dari “prajurit” dengan cara menumpukan dua buah bidak yang bewarna sama. “Prajurit” dapat menjadi “ratu” dengan cara berjalan terus kearea permainan musuh.

Permainan ini dimainkan pada papan berukuran 8x8 kotak, dengan warna-warna hitam dan merah berganti-ganti. Permainan hanya dapat dimainkan pada 32 kotak hitam saja. Bidak melangkah dengan cara diagonal dan pemain dapat “memakan” bidak pemain lawan dengan cara melangkahinya.

Secara singkat, objektif dari permainan ini adalah pemain harus menghabisi semua bidak pemain musuh seperti catur. Ketika permainan dimulai, masing-masing pemain mulai dengan 12 buah bidak yang ditempatkan dalam tiga barisan yang terdekat dengan mereka.

Papan permainan checker



Posisi awal permaianan

Pada permainan ini, diperlukan strategi untuk menentukan langkah yang akan diambil pemain untuk memenangkan permainan dengan cara "memakan" semua bidak lawan. Pada makalah ini, strategi tersebut dilakukan berdasarkan algoritma greedy. Algoritma greedy berusaha menghasilkan solusi optimal, namun tidak selalu dapat menghasilkan solusi yang optimal karena prinsip algoritma ini adalah "take what you can get now".

II. DASAR TEORI

Algoritma greedy akan digunakan sebagai dasar penentuan langkah selanjutnya yang akan dipilih oleh pemain.

Berdasarkan arti harafiahnya, greedy berarti tamak. Sesuai dengan artinya, prinsip dari algoritma greedy adalah memilih keputusan terbaik pada setiap state atau langkah (akan menghasilkan keputusan optimum lokal) dengan harapan akan menghasilkan keputusan optimum global. Namun, apabila keputusan terbaik pada suatu langkah ternyata tidak mengarahkan pada optimum global, tidak dapat dilakukan peninjauan kembali (backtracking) terhadap keputusan tersebut. Oleh karena itu, hasil keputusan dari algoritma greedy mungkin tidak optimal.

Algoritma greedy memiliki beberapa elemen yang harus dupenuhi, yaitu sebagai berikut:

- Himpunan Kandidat

  Himpunan kandidat adalah semua ruang sampel yang dapat dipilih sebagai solusi.

- Himpunan Solusi

  Himpunan solusi adalah himpunan bagian dari himpunan kandidat yang merupakan solusi dari permasalahan yang ada.

- Fungsi Seleksi

  Fungsi seleksi merupakan unsure khas dari algoritma ini. Fungsi ini digunakan untuk menyeleksi anggota-anggota dari himpunan kandidat yang akan termasuk ke dalam tahap selanjutnya yang akan diperiksa menggunakan fungsi kelayakan agar dapat masuk ke dalam himpunan solusi.

- Fungsi Layak

  Fungsi layak digunakan untuk memeriksa apakah solusi yang dipilih merupakan solusi yang layak untuk dimasukkan ke dalam himpunan solusi atau tidak.

- Fungsi Objektif

  Fungsi objektif adalah solusi optimum yang dihasilkan.

Berikut adalah pseudocode dari algoritma greedy:

```
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
      x ← SELEKSI(C)        { pilih sebuah kandidat dari C}
      C ← C - {x}               { elemen himpunan kandidat berkurang satu }
      if LAYAK(S ∪ {x}) then
          S ← S ∪ {x}
      endif
    endwhile
    {SOLUSI(S) or C = {} }

    if SOLUSI(S) then
      return S
    else
      write('tidak ada solusi')
    endif
```

Algoritma greedy tidak selalu menghasilkan solusi optimum global. Hal ini disebabkan oleh algoritma greedy tidak memeriksa semua kemungkinan solusi yang mungkin dan hanya mengambil solusi terbaik relatif (yang dapat disebut sebagai optimum lokal) pada setiap langkah atau stage-nya.

Namun karena algoritma greedy ini hampir selalu menghasilkan solusi yang mendekati optimal global, maka algoritma ini sangat cocok digunakan untuk memecahkan persoalan optimasi. Agar kemungkinan hasil optimum global didapatkan, hal yang dapat dilakukan adalah memilih fungsi seleksi yang baik karena fungsi seleksilah yang menentukan solusi mana yang akan diambil dan dimasukkan ke dalam himpunan solusi pada setiap langkah atau stage.

Hasil dari algoritma greedy mungkin lebih dari satu sehingga solusi yang dihasilkan mungkin berbeda-beda. Karena algoritma greedy tidak menjamin optimalitas solusi, maka pemilihan fungsi seleksi pada algoritma ini menjadi sangat penting.

III. IMPLEMENTASI

Pada kasus ini, terdapat skor sebuah papan yang akan memberikan evaluasi apakah kondisi papan setelah pemain mengerakan bidak merupakan kondisi optimal atau tidak.

Sebagai contoh, apabila pemain mengerakan bidak putih pada kotak A3 ke kotak B4 itu akan mengakibatkan bidak tersebut akan memiliki potensi "dimakan" oleh bidak hitam pada kotak C5 dan bidak hitam tersebut tidak dapat dimakan balik oleh bidak putih pada kotak B2. Hal tersebut mengakibatkan evaluasi pergerakan bidak putih dari kotak A3 ke B4 menjadi rendah karena pemain akan kehilangan bidak-nya secara gratis. Selanjutnya, skor itu akan diistilahkan sebagai scoreBoardAfterMove.

Elemen-elemen algoritma greedy pada permasalahan optimasi pengerakan pemain pada permainan checker ini adalah sebagai berikut:

- Himpunan Kandidat

  Semua kondisi papan

- Himpunan Solusi

  Himpunan kondisi papan yang dipilih sebagai kondisi papan setelah melakukan pengerakan bidak

- Fungsi Seleksi

  Kondisi papan memiliki nilai evaluasi scoreBoardAfterMove tertinggi

- Fungsi Layak

  Kondisi papan setelah pengerakan masih sesuai aturan papan permainan checker

- Fungsi Objektif

  Memakan semua bidak permainan musuh

Sebelum memilih pengerakan bidak yang dipilih menggunakan algoritma greedy, perlu digenerate semua pengerakan yang dapat dilakukan pemain yang akan menghasilkan kumpulan kondisi papan setelah pengerakan bidak pemain.

Langkah-langkah untuk mendapatkan Himpunan Solusi dari Himpunan Kandidat adalah sebagai berikut:

1. Mengenerate semua kondisi papan dari semua pengerakan yang dapat dilakukan pemain untuk kondisi papan saat ini
2. Himpunan semua kondisi papan tersebut dapat digenereate menggunakan fungsi generateAllBoardAfterMove
3. Evaluasi semua kondisi papan tersebut satu persatu untuk mendapatkan scoreBoardAfterMove setiap papan
4. Evaluasi tersebut dapat dilakukan dengan menggunakan fungsi evalBoardAfterMove.
5. Pilihlah board dengan nilai scoreBoardAfterMove terbesar untuk dijadikan kondisi papan selanjutnya menggunakan fungsi changeBoard
6. Lakukan langkah 1 sampai 5 hingga permainan berakhir

Daftar Fungsi

1. Fungsi generateAllBoardAfterMove

```
function generateAllBoardAfterMove(input Now:Board) -> ArrayOfBoard

{ Fungsi ini akan mengembalikan array of board dari kondisi papan saat ini dengan kondisi-kondisi papan setelah semua pengerakan bidak yang mungkin dilakukan saat ini }
```

2. Fungsi evalBoardAfterMove

```
Function evalBoardAfterMove(input B:Board) -> Integer

{ Fungsi ini akan menghasilkan nilai scoreBoardAfterMove dengan cara mengevaluasi semua kotak pada Board B dengan kondisi yang telah ditentukan pembuat program }
```

3. Prosedur changeBoard

```
procedure changeBoard(input/output Now:Board, input: chosenBoard: Board)

{ Prosedur ini akan mengubah kondisi papan saat ini menjadi kondisi papan yang memiliki
```

nilai scoreBoardAfterMove terbesar dari array of board yang digenerate oleh fungsi generateAllBoardAfterMove }

IV. KESIMPULAN

Algoritma greedy merupakan algoritma yang cocok untuk persoalan optimasi. Walaupun ada kemungkinan solusi yang diberikan algoritma ini bukan merupakan solusi optimal. Dalam implementasinya, algoritma greedy dapat dikatakan sederhana dibandingkan algoritma lain.

Algoritma greedy sangat beresiko jika digunakan untuk mengharapkan solusi yang paling optimum karena harus dapat dibuktikan secara matematis terlebih dahulu bahwa hasil yang didapatkan dari algoritma ini merupakan solusi yang paling optimum.

Solusi dari persoalan algoritma greedy yang terkadang tidak menghasilkan solusi optimum global, dapat digunakan algoritma lain yaitu algoritma exhaustive search. Dengan menggunakan algoritma exhaustive search, semua solusi yang mungkin dicapai akan didata satu-satu, lalu kemudian diseleksi sehingga ditemukan satu solusi yang paling optimal. Dengan cara ini, solusi optimal pasti tercapai karena semua solusi yang mungkin dicatat.

VIDEO LINK AT YOUTUBE

https://youtu.be/ITiC4OYhtIA

UCAPAN TERIMA KASIH

Pertama penulis ingin mengucapkan terimakasih kepada Tuhan YME atas berkat dan hidayat-Nya penulis dapat menyelesaikan makalah ini. Terima kasih kepada Dr. Masayu Leylia Khodra, Dr. Nur Ulfa Maulidevi, S.t, M.Sc., dan Dr. Ir. Rinaldi Munir, M.T. sebagai dosen pengajar mata kuliah Strategi Algoritma yang telah memberikan ilmunya. Penulis juga ingin berterima kasih kepada teman-teman yang telah membantu dan memberikan dukungan kepada penulis dalam penulisan makalah ini.

REFERENSI

[1] Munir, Rinaldi. Diktat Kuliah IF2211 Strategi Algoritma. Bandung: Program Studi Teknik Informatika Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung. 2009.

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, 04 Mei 2020

Ahmad Mutawalli / 13517026