Penerapan Algoritma Greedy dalam Strategi Permainan Battleship

Edia Zaki Naufal Ilman - 13521141
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13521141@std.stei.itb.ac.id

***Abstract*—Battleship merupakan sebuah permainan yang bertujuan menebak lokasi semua kapal pemain lain. Meskipun memiliki elemen keberuntungan dalam menebak, pemain juga dapat menggunakan suatu lojik yang dapat membantu mendapatkan probabilitas yang tinggi dalam menebak. Strategi dan algoritma yang dapat diimplementasikan dalam permainan ini salah satunya adalah dengan algoritma greedy.**

Keywords—battlehip, greedy, probabilitas

I. PENDAHULUAN

Battleship merupakan sebuah permainan klasik yang dimainkan oleh dua pemain yang tujuannya adalah menghancurkan kapal milik pemain lawan dengan cara menebak lokasi dari kapal lawan. Meskipun terdapat elemen keberuntungan pada permainan ini, pemain juga dapat memperhitungkan tebakan lokasi kapal lawan dengan menggunakan suatu lojik probabilitas. Salah satu cara adalah menghitung probabilitas penempatan kapal pada suatu grid atau matrix sehingga terbentuk matrix. Dengan membentuk matrix probabilitas, dapat ditentukan lokasi mana yang lebih mungkin terdapat kapal lawan sehingga dapat mengurangi probabilitas pada lokasi lain dan mengurangi jumlah tebakan pada permainan. Pada makalah ini, akan diterapkan Algoritma Greedy sebagai upaya untuk meminimalisasi dan mengoptimasi tebakan pada permainan Battleship.

Fig. 1. Bentuk Permainan Battleship

II. PENGETAHUAN DASAR BATTLESHIP

A. *Gameplay*

Battleship dimainkan dengan 2 (dua) pemain dan giliran dilakukan secara bergantian. Pada awal permainan, kedua pemain akan diminta untuk menempatkan kapal-kapal yang diberikan pada panel atau grid miliknya. Penempatan kapal hanya bisa dilakukan secara horizontal atau vertikal dan tidak boleh secara diagonal. Setelah penematan kapal, permainan dapat dimulai.

Pada giliran pemain, pemain hanya perlu melakukan satu hal, yaitu menebak lokasi dari dari kapal lawan. Pemain akan menyebutkan sebuah lokasi sesuai dengan koordinat pada panel kepada pemail lawan dan akan memberikan feedback terdahap tebakan tersebut.

Jika tebakan pemain mengenai bagian kapal lawan, maka pemain lawan akan memberikan feedback “Hit”, dan jika tidak mengenai satu pun kapal, maka akan diberikan feeback “Miss”. Dan jika tebakan yang mengenai kapal tersebut berhasil menenggelamkan kapal lawan dengan cara seluruh bagian kapal telah ditembak, maka akan diberikan feedback “Sink” yang berarti suatu kapal lawan berhasil ditenggelamkan.

Giliran pemain terus berlanjut dan bergantian hingga seluruh kapal dari 1 (satu) pemain berhasil ditenggelamkan.

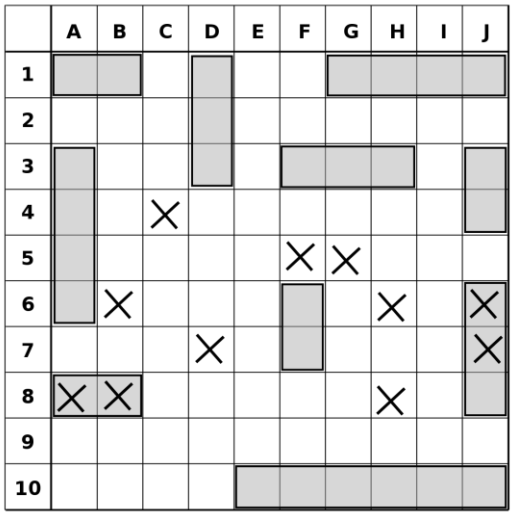


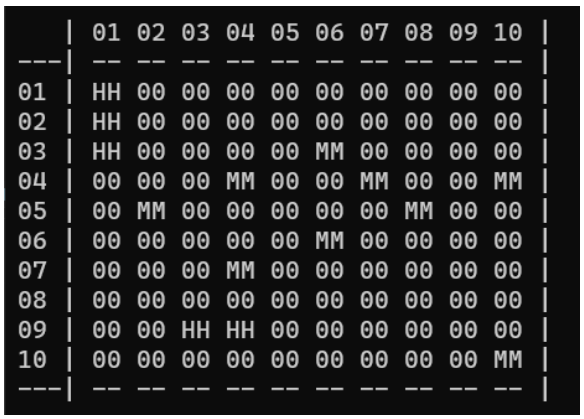
Fig. 2. Contoh Bentuk Format Gameplay Battleship

B. Game Elements

1. Map

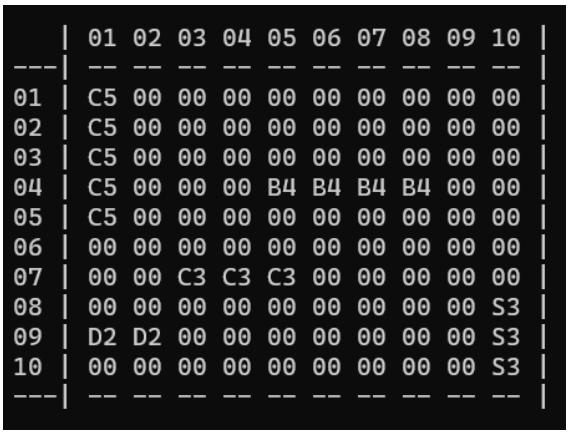
Pada permainan ini, setiap pemain memiliki 2 (dua) jenis map, yaitu map untuk menempatkan kapal diri sendiri dan map untuk melihat hasil tembakan kepada lawan. Pada map untuk menempatkan kapal sendiri, akan diberikan marker untuk setiap lokasi kapal. Dan untuk map hasil tembakan akan diberikan marker untuk setiap feedback Hit dan Miss yang didapat setelah melakukan tembakan.

Berikut merupakan contoh map dari program permainan Battleship yang dibuat penulis:



```
   | 01 02 03 04 05 06 07 08 09 10 |
---| -- -- -- -- -- -- -- -- -- -- |
01 | HH 00 00 00 00 00 00 00 00 00 |
02 | HH 00 00 00 00 00 00 00 00 00 |
03 | HH 00 00 00 00 MM 00 00 00 00 |
04 | 00 00 00 MM 00 00 MM 00 00 MM |
05 | 00 MM 00 00 00 00 00 MM 00 00 |
06 | 00 00 00 00 00 MM 00 00 00 00 |
07 | 00 00 00 MM 00 00 00 00 00 00 |
08 | 00 00 00 00 00 00 00 00 00 00 |
09 | 00 00 HH HH 00 00 00 00 00 00 |
10 | 00 00 00 00 00 00 00 00 00 MM |
---| -- -- -- -- -- -- -- -- -- -- |
```

Fig. 3. Map untuk Menandakan Hasil Tembakan Pemain



```
   | 01 02 03 04 05 06 07 08 09 10 |
---| -- -- -- -- -- -- -- -- -- -- |
01 | C5 00 00 00 00 00 00 00 00 00 |
02 | C5 00 00 00 00 00 00 00 00 00 |
03 | C5 00 00 00 00 00 00 00 00 00 |
04 | C5 00 00 00 B4 B4 B4 B4 00 00 |
05 | C5 00 00 00 00 00 00 00 00 00 |
06 | 00 00 00 00 00 00 00 00 00 00 |
07 | 00 00 C3 C3 C3 00 00 00 00 00 |
08 | 00 00 00 00 00 00 00 00 00 S3 |
09 | D2 D2 00 00 00 00 00 00 00 S3 |
10 | 00 00 00 00 00 00 00 00 00 S3 |
---| -- -- -- -- -- -- -- -- -- -- |
```

Fig. 4. Map untuk Menempatkan Kapal Pemain

1. Kapal

Setiap pemain akan memiliki 5 (lima) kapal pada awal permainan. Kelima kapal tersebut akan memiliki panjang yang berbeda. Kapal-kapal ini akan ditemmpatkan pada map masing-masing pemain dan akan diberi marker pada map. Detail dari panjang dan nama setiap kapal dapat dilihat sebagai berikut:

Nama Kapal	***Marker pada Program***	***Panjang Kapal***
Carrier	C5	5
Battleship	B4	4
Cruiser	C3	3
Submarine	S3	3
Destroyer	D2	2

TABLE I. TABEL DETAIL KAPAL-KAPAL

III. TEORI DASAR ALGORITMA

A. Greedy Algorithm

Greedy Algorithm atau Algoritma Greedy adalah suatu algoritma untuk memecahkan suatu masalah atau persoalan dengan mengambil keputusan dengan value paling tinggi pada setiap langkahnya. Dengan kata lain, Algoritma Greedy memilih keputusan optimum lokal pada setiap tahap penyelesaian permasalahan atau persoalan dengan harapan mendapatkan hasil optimum global.

Sesuai dengan Namanya yaitu "*Greedy"* yang berarti rakus atau tamak, algoritma greedy mengimplementasikan prinsip "*take what you can get now!"*. Prinsip kalimat ini memiliki makna untuk mengambil yang terbaik setiap saat, atau dalam kasus ini, setiap langkah. Algoritma Greedy tidak selalu menghasilkan solusi yang optimal karena, solusi optimum local tidak selalu menghasilkan solusi yang optimum secara global.

Akan tetapi, algoritma ini cukup mendekati solusi optimum global dalam waktu dan tingkat kesulitan yang lebih baik dari algoritma dasar lain seperti algoritma *Brute Force*.

Algoritma Greedy memiliki banyak elemen-elemen umum, dalam implementasinya, yaitu:

2. Himpunan kandidat

   Suatu himpunan yang berisi kumpulan kandidat-kandidat yang bisa dijadikan sebagai solusi.

3. Himpunan solusi

   Suatu himpunan yang berisi kumpulan kandidat-kandidat yang terpilih sebagai solusi.

4. Fungsi solusi

   Suatu fungsi yang menentukan solusi dari kumpulan kandidat pada himpunan kandidat.

5. Fungsi seleksi

   Suatu fungsi untuk memilih elemen-elemen solusi dari himpunan kandidat menggunakan strategi Greedy yang diimplementasikan.

6. Fungsi kelayakan

   Suatu fungsi yang menentukan apakah kumpulan kandidat pada himpunan kandidat layak dimasukkan kedalah himpunan solusi.

7. Fungsi objektif

   Suatu fungsi yang bertujuan untuk lebih mengoptimasi hasil solusi yang didapatkan.

```
function greedy(C : himpunan_kandidat) → himpunan_solusi
{ Mengembalikan solusi dari persoalan optimasi dengan algoritma greedy }
Deklarasi
   x : kandidat
   S : himpunan_solusi

Algoritma:
   S ← {}     { inisialisasi S dengan kosong }
   while (not SOLUSI(S)) and (C ≠ {} ) do
      x ← SELEKSI(C)     { pilih sebuah kandidat dari C}
      C ← C – {x}          { buang x dari C karena sudah dipilih }
      if LAYAK(S ∪ {x}) then      { x memenuhi kelayakan untuk dimasukkan ke dalam himpunan solusi }
         S ← S ∪ {x}      { masukkan x ke dalam himpunan solusi }
      endif
   endwhile
   {SOLUSI(S) or C = {} }

   if SOLUSI(S) then    { solusi sudah lengkap }
      return S
   else
      write('tidak ada solusi')
   endif
```

Fig. 5. Skema Umum Algoritma Greedy

IV. IMPLEMENTASI

Penerapan Algoritma Greedy pada strategi permainan Battleship dilakukan pada pemilihan koordinat lokasi dengan probabilitas tertinggi diantara lokasi lain. Probabilitas tersebut dihitung dan disimpan pada suatu matriks yang setiap elemennya mengindikasikan value probabilitas dari ditempatkannya suatu kapal lawan dengan Panjang tertentu.

Implementasi dari strategi ini secara umum terbagi menjadi dua tahap, yaitu tahap perhitungan dan tahap pemilihan.

A. Tahap Perhitungan

Untuk dapat memilih koordinat lokasi dengan value tertinggi, perlu diperhitungkan terlebih dahulu value-value tersebut. Hal ini dilakukan dengan menghitung berapa banyak posisi berbagai kapal yang dapat ditempatkan pada suatu lokasi. Untuk setiap posisi kapal yang berhasil ditempatkan, akan ditambahkan value bernilai 1 (satu) di mana setiap elemen (i, j) pada matriks merupakan representasi dari probabilitas penempatan kapal pada lokasi (i, j). Tahap perhitungan ini dilakukan setiap sebelum melakukan tahap pemilihan.

Berikut merupakan langkah perhitungan dari matriks probabilitas dari strategi yang diimplementasikan:

1) Pilih elemen pertama pada baris atau kolom pertama dari matriks (i, j).

2) Tempatkan suatu kapal dengan panjang L secara horizontal atau vertikal dimulai dari elemen yang sebelumnya dipilih (i, j).

3) Apabila posisi kapal dapat diterima, maka tambahkan nilai sebesar 1 (satu) pada setiap elemen yang berhasil ditempati oleh kapal tersebut.

4) Ulang langkah 3 (tiga) dengan kapal berbeda dengan panjang L yang berbeda

5) Pilih elemen berikutnya pada baris atau kolom yang sama dan ulangi langkah 3 (tiga) sampai dengan 4 (empat).

6) Lakukan untuk seluruh baris dan kolom baik secara horizontal maupun vertikal

Setelah melakukan seluruh langkah perhitungan di atas, akan didapatkan matriks seperti berikut:

```
    | 01 02 03 04 05 06 07 08 09 10 |
----|-- -- -- -- -- -- -- -- -- -- --|
01  | 10 15 19 21 22 22 21 19 15 10 |
02  | 15 20 24 26 27 27 26 24 20 15 |
03  | 19 24 28 30 31 31 30 28 24 19 |
04  | 21 26 30 32 33 33 32 30 26 21 |
05  | 22 27 31 33 34 34 33 31 27 22 |
06  | 22 27 31 33 34 34 33 31 27 22 |
07  | 21 26 30 32 33 33 32 30 26 21 |
08  | 19 24 28 30 31 31 30 28 24 19 |
09  | 15 20 24 26 27 27 26 24 20 15 |
10  | 10 15 19 21 22 22 21 19 15 10 |
----|-- -- -- -- -- -- -- -- -- -- --|
```

Fig. 6. Matriks Probabilitas pada Awal Permainan

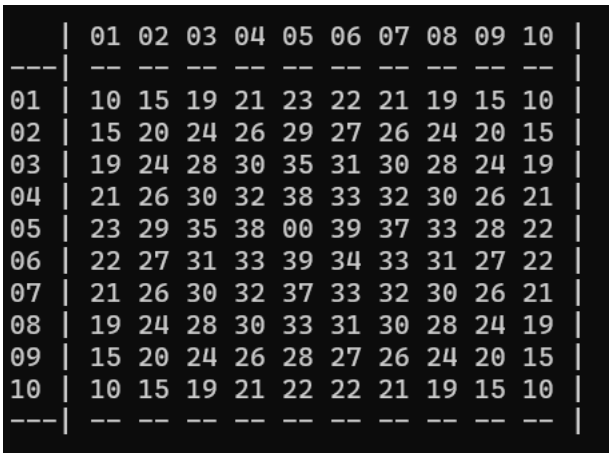
Dapat dilihat bahwa elemen di tengah matriks akan lebih bernilai besar dibandingan elemen yang berada di pinggir matriks.

Apabila tahap perhitungan dilakukan bukan pada awal permainan atau pertengahan permainan di mana sudah sudah ada elemen yang “ditembak”, maka akan ada sedikit perubahan pada langkah perhitungan matriks. Beberapa perbedaan yang perlu dilakukan adalah sebagai berikut:

1) Apabila terdapat elemen yang sudah “ditembak” sebelumnya, maka elemen tersebut diberi nilai 0 (nol) karena sudah tidak mungkin ditembak lagi.

2) Elemen yang sudah ditembak dapat mempengaruhi nilai pada elemen-elemen disekitarnya.

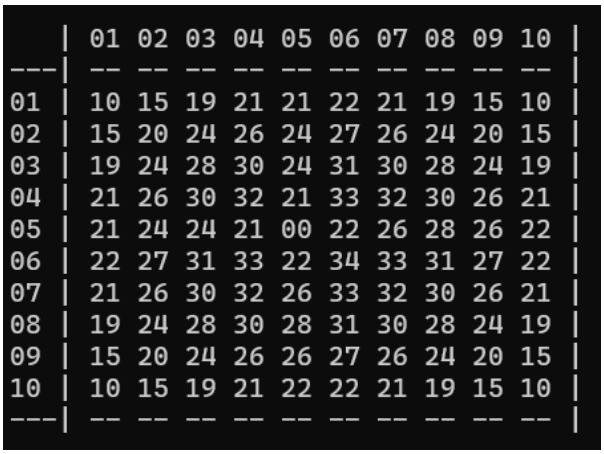
- Apabila elemen yang sudah ditembak tersebut menghasilkan “Hit”, maka seluruh elemen-elemen disekitarnya yang mungkin dapat ditempatkan kapal dengan panjang L yang *intersect* dengan elemen hit tersebut bertambah nilainya sebesar 1 (satu).



	01	02	03	04	05	06	07	08	09	10
01	10	15	19	21	23	22	21	19	15	10
02	15	20	24	26	29	27	26	24	20	15
03	19	24	28	30	35	31	30	28	24	19
04	21	26	30	32	38	33	32	30	26	21
05	23	29	35	38	00	39	37	33	28	22
06	22	27	31	33	39	34	33	31	27	22
07	21	26	30	32	37	33	32	30	26	21
08	19	24	28	30	33	31	30	28	24	19
09	15	20	24	26	28	27	26	24	20	15
10	10	15	19	21	22	22	21	19	15	10

Fig. 7. Matriks Probabilitas Setelah Terjadi Hit

- Apabila elemen yang sudah ditembak tersebut menghasilkan “Miss”, maka elemen tersebut dianggap sebagai “halangan”. Sehingga pada saat penempatan posisi kapal, jika suatu bagian kapal *intersect* dengan elemen tersebut, akan tidak diterima.



	01	02	03	04	05	06	07	08	09	10
01	10	15	19	21	21	22	21	19	15	10
02	15	20	24	26	24	27	26	24	20	15
03	19	24	28	30	24	31	30	28	24	19
04	21	26	30	32	21	33	32	30	26	21
05	21	24	24	21	00	22	26	28	26	22
06	22	27	31	33	22	34	33	31	27	22
07	21	26	30	32	26	33	32	30	26	21
08	19	24	28	30	28	31	30	28	24	19
09	15	20	24	26	26	27	26	24	20	15
10	10	15	19	21	22	22	21	19	15	10

Fig. 8. Matriks Probabilitas Setelah Terjadi Miss

Dengan perubahan-perubahan tersebut, algoritma akan lebih memilih lokasi di sekitar elemen yang Hit dan menghindari elemen yang Miss.

B. Tahap Pemilihan

Setelah didapatkan matriks probabilitas, maka strategi akan lanjut ke tahap pemilihan. Pada tahap ini akan dicari lokasi dengan nilai tertinggi. Apabila terdapat lebih dari 1 (satu) elemen dengan nilai tertinggi yang sama, maka elemen pertama yang didapat akan menjadi elemen yang terpilih.

Jika sudah didapatkan lokasi elemen tertinggi, akan dilakukan tebakan atau “tembakan” pada lokasi tersebut. Pada akhir penembakan akan didapatkan feeback berupa “Hit” atau “Miss”, di mana Hit berarti lokasi yang ditembak berhasil mengenai suatu kapal lawan dan Miss berarti lokasi yang ditembak gagal atau tidak mengenai kapal lawan.

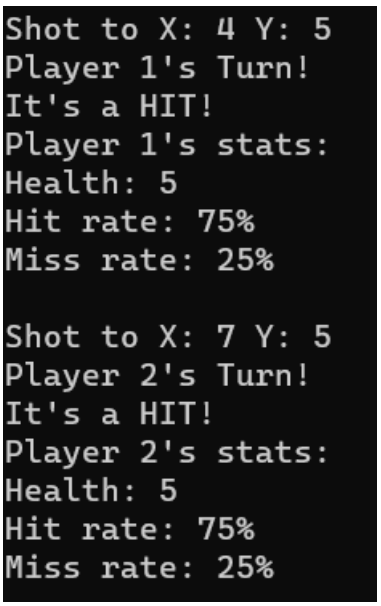


Fig. 9. Feedback Saat Terjadi Hit

Setelah menembak, algoritma akan menuliskan hasil feedback pada panel player untuk menandakan lokasi yang sudah ditembak. Jika Hit, akan diberi marker “HH” dan jika Miss, akan diberi marker “MM” seperti contoh pada Bab II.

Pada akhir giliran, program akan kembali menjalankan tahap perhitungan untuk menghitung kembali matriks probabilitas setelah terjadinya perubahan pada map. Kedua tahap ini akan kemudian terus diulang hingga salah satu pemain berhasil mengalahkan pemain lain.

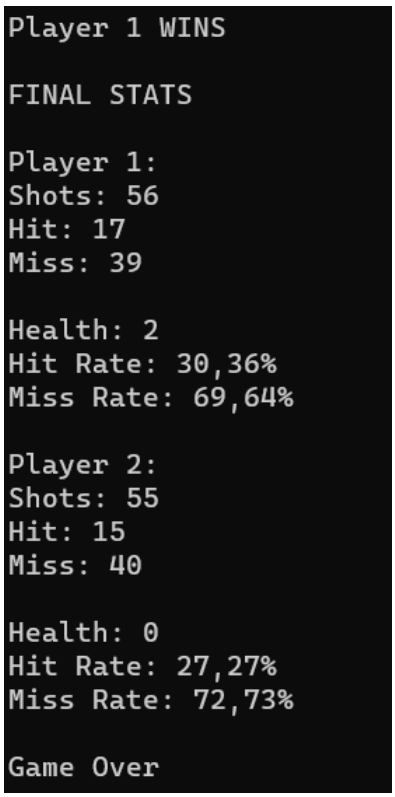


Fig. 10. Tampilan Hasil Akhir Setelah Permainan Berakhir

V. Uji Coba dan Analisis

A. *Uji Coba*

Untuk uji coba, kapal-kapal setiap pemain akan ditempatkan dengan formasi yang sama seperti berikut:

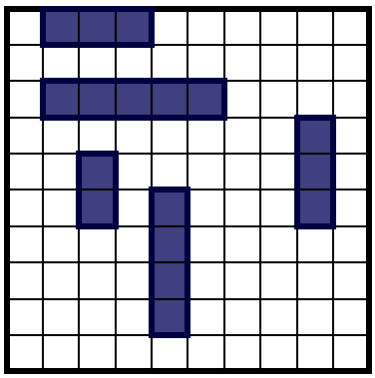


Fig. 11. Penempatan Kapal-Kapal pada Uji Coba

Setelah penempatan kapal-kapal, permainan dimulai dan menghasilkan hasil sebagai berikut:

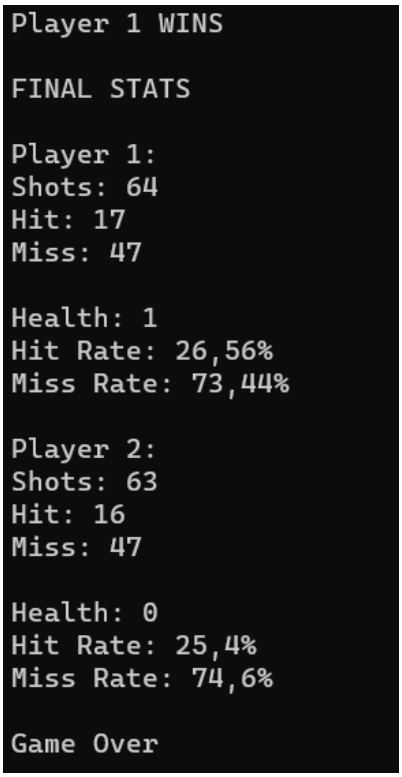


Fig. 12. Tampilan Hasil Akhir Uji Coba

B. *Analisis Pembahasan*

Dapat dilihat dari hasil uji coba bahwa hit rate pemain masih cukup rendah di sekitar 25% dan miss ratenya masih cukup tinggi di sekitar 75%. Salah satu alasan rendahnya hit rate adalah karena algoritma yang pada dasarnya masih “menebak” lokasi kapal. Meskipun memilih probabilitas yang lebih tinggi, algoritma tidak memperhitungkan probabilitas dari penempatan kapal oleh manusia atau pemain yang menempatkannya. Sebagai gambarannya, jika seseorang orang tahu bahwa jika dihitung dari probabilitas kemungkinan penempatan kapal, daerah di sekitar batas map memiliki nilai terendah, pemain akan dengan sengaja menempatkan kapal tersebut disekitar batas atau daerah lain yang memiliki probabilitas rendah. Hasilnya, algoritma akan mengalami kesulitan dalam menemukan kapal yang ditempatkan karena daerah dengan nilai kecil akan diperiksa di paling akhir.

Namun, hasil yang didapatkan hanya dari satu uji coba sehingga diperlukan lebih banyak case sample untuk lebih memvalidasi data. Tak hanya itu, hasil jumlah tebakan yang bernilai sekitar 60 (akumulatif dari pemenang pada Fig. 10 dan Fig. 12) cukup lebih baik dibandingkan jumlah percobaan rata-rata normal tanpa algoritma yang bernilai sekitar sekitar 65. Maka dapat diambil kesimpulan bahwa algoritma greedy menghasilkan solusi yang cukup optimal namun tidak optimum secara global.

VI. Kesimpulan dan Saran

Algoritma greedy dapat diterapkan pada banyak persoalan di kehidupan sehari-hari, termasuk untuk permainan hiburan seperti Battleship. Pada makalah ini diperoleh kelebihan dan kekurangan dalam penerapan algoritma greedy sebagai strategi untuk bermain secara optimal. Pada makalah ini juga telah dilakukan uji coba secara langsung untuk mendapatkan bukti hasil yang konkrit. Hasil dari uji coba menunjukkan bahwa algoritma greedy tidak sepenuhnya memberikan solusi yang optimal atau dalam kata lain, solusi yang diberikan oleh algoritma greedy bukanlah solusi optimum global. Pada uji coba terlihat bahwa masih banyak kasus yang menghasilkan gerakan yang tidak efisien dan efektif. Meskipun begitu, algoritma greedy ini cukup optimum dibandingkan permainan yang dilakukan secara normal pada biasanya. Apabila ingin didapatkan solusi yang lebih baik, sebaiknya menggunakan algoritma lain yang lebih tepat dan terjamin. Dengan demikian, dapat disimpulkan bahwa apabila mengutamakan kompleksitas algoritma yang relatif sederhana dan batas waktu yang wajar, maka algoritma greedy menjadi alternatif solusi yang baik.

Penulis juga perlu menambahkan bahwa algoritma greedy yang diimplementasikan oleh penulis bukanlah penerapan yang sempurna dan masih dapat dikembangkan. Salah satu saran untuk pengembangan dalam algoritma greedy ini adalah dengan menambahkan suatu alur algoritma dimana lokasi pada pinggir map tidak terlantarkan sepenuhnya.

VII. Pranala Video Makalah

Berikut merupakan pranala video penjelasan makalah yang dibuat oleh penulis: https://youtu.be/UIr9nGhv0gc.

VIII. Ucapan Terima Kasih

Puji syukur penulis panjatkan kepada Allah SWT yang telah memberikan rahmat serta karunianya sehingga pada akhirnya, penulis dapat menyelesaikan makalah ini tepat pada waktunya. Penulis sadar bahwa makalah ini tidak akan terwujud tanpa adanya bantuan dan dorongan dari berbagai pihak. Oleh karena itu, penulis mengucapkan terima kasih dan penghargaan sebesar-besarnya kepada Dr. Ir. Rinaldi Munir, M.T. sebagai dosen pengampu mata kuliah IF2211 Strategi Algoritma kelas 01 yang telah mendidik penulis selama satu semester. Penulis juga mengucapkan terima kasih kepada keluarga dan teman-teman penulis yang telah memberikan dukungan yang luar biasa kepada penulis.

References

[1] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf . Diakses pada 22 Mei 2023

[2] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag2.pdf . Diakses pada 22 Mei 2023

[3] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Greedy-(2022)-Bag3.pdf . Diakses pada 22 Mei 2023

[4] Matlin, C. (2012, May 16). *How to win at battleship*. Slate Magazine. https://slate.com/culture/2012/05/how-to-win-at-battleship.html . Diakses pada 22 Mei 2023

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, 22 Mei 2023

Edia Zaki Naufal Ilman 13521141