Aplikasi Algoritma Greedy dalam pemecahan solusi permainan 2048

Muhamad Rifki Virziadeili Harisman - 135522120
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): rifkivirziadeili@gmail.com

***Abstract*—Permainan 2048 adalah permainan teka-teki berbasis grid yang telah menarik perhatian banyak peneliti di bidang kecerdasan buatan dan algoritma pencarian. Dalam penelitian ini, akan dilakukan eksplorasi pendekatan algoritma greedy untuk menyelesaikan permainan 2048. Algoritma greedy adalah metode yang memilih langkah terbaik pada setiap langkah tanpa mempertimbangkan konsekuensi jangka panjang. Penelitian ini bertujuan untuk menganalisis efektivitas heuristik greedy dalam mencapai skor tertinggi di permainan 2048. (*Abstract*)**

Kata kunci—greedy, permainan 2048, teka-teki

I. Pendahuluan

Permainan 2048, adalah permainan yang memperlihatkan blok blok angka pada grid 4x4. Permainan ini telah menjadi salah satu permainan teka-teki yang paling populer di kalangan pengguna perangkat digital. Permainan ini mengharuskan pemain untuk menggeser ubin bernomor pada grid dengan tujuan menggabungkan ubin yang sama hingga mencapai angka 2048 atau lebih tinggi. Kesederhanaan konsep dan tantangan strategis yang ditawarkan oleh permainan ini membuat penulis tertarik untuk menyelesaikan permainan tersebut dengan menggunakan algoritma greedy.

Setelah mencari di berbagai sumber, ternyata jarang sekali yang memilih penyelesaian permainan 2048 menggunakan algoritma greedy. Algoritma ini merupakan salah satu algoritma pencarian solusi yang cukup sederhana. Algoritma greedy bekerja dengan memilih langkah yang secara lokal optimal pada setiap langkah, tanpa mempertimbangkan dampak jangka panjang dari langkah tersebut. Meskipun pendekatan ini intuitif dan mudah diimplementasikan, efektivitasnya dalam mencapai skor tinggi pada permainan 2048 masih belum banyak dieksplorasi secara mendalam.

Penelitian ini bertujuan untuk mengeksplorasi dan menganalisis kinerja algoritma greedy dalam menyelesaikan permainan 2048. Penulis akan mengimplementasikan beberapa variasi heuristik greedy dan mengevaluasi sejauh mana algoritma ini mampu mencapai skor yang tinggi dibandingkan dengan algoritma pencarian lainnya. Dalam studi ini, penulis juga akan mengidentifikasi kelemahan dan kekuatan dari pendekatan greedy serta memberikan rekomendasi untuk perbaikan di masa mendatang.

Dengan memahami bagaimana algoritma greedy berperilaku dalam konteks permainan 2048, penulis berharap dapat memberikan kontribusi dalam pengembangan strategi AI yang lebih efisien dan efektif untuk menyelesaikan permainan teka-teki serupa. Penelitian ini tidak hanya memperkaya literatur mengenai algoritma pencarian dalam permainan, tetapi juga membuka peluang untuk penerapan teknik greedy yang dioptimalkan dalam berbagai aplikasi AI lainnya.

II. Teori Dasar

1. Permainan 2048

Permainan 2048 adalah permainan teka-teki berbasis grid yang dikembangkan oleh Gabriele Cirulli pada tahun 2014 . Permainan ini dimainkan dengan tujuan menggabungkan ubin bernomor yang sama untuk mencapai nilai 2048. Setiap pergerakan pemain menggeser semua ubin pada arah tertentu (atas, bawah, kiri, atau kanan), dan setiap gerakan menghasilkan ubin baru dengan nilai 2 atau 4 yang muncul secara acak di grid. Permainan berakhir ketika tidak ada gerakan yang mungkin dilakukan lagi.

Gambar 1. Tampilan permainan 2048
(source:dokumen pribadi)

2. Algoritma Greedy

Algoritma greedy merupakan metode yang paling populer dan sederhana untuk memecahkan persoalan optimasi. Masalah optimasi dapat berupa maksimasi ataupun minimasi. Algoritma greedy adalah pendekatan heuristik yang memilih solusi lokal terbaik di setiap langkah dengan harapan bahwa pilihan ini akan mengarah pada solusi global yang optimal. Karakteristik utama dari algoritma greedy adalah bahwa ia membuat keputusan miopik (myopic) berdasarkan informasi yang tersedia saat itu tanpa mempertimbangkan dampak jangka panjang dari keputusan tersebut. Meskipun algoritma greedy sering kali tidak memberikan solusi optimal untuk semua jenis masalah, algoritma ini sangat cepat dan mudah diimplementasikan.

Pada algoritma greedy terdapat beberapa elemen penting seperti

- Himpunan kandidat, yang dinyatakan sebagai C, merujuk pada sekumpulan opsi atau alternatif yang tersedia untuk dipilih pada setiap tahap proses. Contoh kandidat ini meliputi simpul atau sisi dalam graf, pekerjaan, tugas, koin, benda, karakter, dan sebagainya.

- Sebaliknya, himpunan solusi, yang direpresentasikan sebagai S, terdiri dari kandidat yang telah dipilih dalam proses tersebut. Ini adalah kumpulan kandidat yang telah terpilih sebagai bagian dari solusi yang dihasilkan.

- Fungsi solusi bertugas untuk mengevaluasi apakah kumpulan kandidat yang dipilih berhasil memberikan solusi yang memenuhi kriteria yang telah ditetapkan. Ini merupakan penilaian akhir terhadap apakah solusi yang diberikan sudah memenuhi kebutuhan atau tidak.

- Untuk memilih kandidat secara berurutan, digunakan fungsi seleksi yang mengikuti suatu strategi greedy tertentu. Strategi ini bersifat heuristik, yang berarti keputusan diambil berdasarkan aturan yang tampaknya paling menguntungkan pada setiap langkah, tanpa mempertimbangkan implikasi jangka panjang.

- Kemudian, fungsi kelayakan memiliki peran dalam memeriksa apakah kandidat yang dipilih memenuhi syarat untuk dimasukkan ke dalam himpunan solusi. Fungsi ini menilai apakah kandidat tersebut layak atau tidak untuk menjadi bagian dari solusi yang dihasilkan.

- Terakhir, fungsi objektif memiliki tujuan untuk memaksimalkan atau meminimalkan suatu nilai tertentu. Ini mungkin berupa fungsi tujuan dalam konteks mengoptimalkan, yang menentukan kriteria untuk mengevaluasi kualitas dari solusi yang dihasilkan.

3. Heuristik

Heuristik adalah sebuah metode yang meningkatkan efisiensi dalam pencarian solusi, tanpa menjamin kelengkapan (completeness). Ini merupakan sebuah tindakan yang membantu menemukan jalur dalam pohon pencarian yang mengarah pada solusi dari suatu masalah. Heuristik didesain untuk menyelesaikan permasalahan dengan mengabaikan kepastian matematis dari solusi. Contoh dari teknik ini termasuk penggunaan intuisi, penilaian berdasarkan pengalaman, atau logika praktis. Heuristik merujuk pada teknik pemecahan masalah yang didasarkan pada pengalaman, pembelajaran, dan penemuan solusi, meskipun solusi tersebut tidak dijamin optimal. Heuristik bukanlah sebuah algoritma, melainkan sebuah panduan. Meskipun tidak selalu menghasilkan solusi optimal, heuristik sangat berguna dalam menyelesaikan masalah.

Teknik Heuristik yang efektif dapat secara signifikan mengurangi waktu yang diperlukan untuk menemukan solusi dengan menghilangkan kebutuhan untuk mempertimbangkan solusi yang tidak relevan. Selain itu, heuristik yang baik juga dapat membantu algoritma mencapai solusi yang lebih optimal dan mendekati hasil yang diinginkan.

4. Kelemahan Algoritma Greedy

Walaupun algoritma greedy menawarkan keunggulan dalam kesederhanaan dan kecepatan, namun memiliki kekurangan yang perlu diperhatikan dalam konteks pemecahan masalah. Salah satu kelemahannya terletak pada kurangnya kemampuan untuk melihat implikasi jangka panjang dari setiap langkah yang diambil. Hal ini menyebabkan algoritma greedy cenderung membuat keputusan berdasarkan informasi lokal tanpa mempertimbangkan konsekuensi global dari setiap langkah tersebut. Selain itu, hasil dari algoritma greedy sangat bergantung pada urutan masukan yang diberikan. Urutan masukan yang berbeda dapat menghasilkan solusi yang berbeda, bahkan jika solusi tersebut tidak optimal secara keseluruhan. Selain itu, algoritma greedy tidak selalu mampu menangani masalah yang kompleks dengan baik, terutama yang melibatkan ketergantungan antar langkah atau kondisi yang kompleks. Ini menyebabkan algoritma ini rentan terhadap kesalahan atau kegagalan dalam menyelesaikan masalah dengan efektif, terutama dalam kasus di mana solusi optimal memerlukan pemilihan langkah yang lebih strategis dan terencana secara lebih komprehensif.

III. Analisis Permasalahan

1. Analisis Permasalahan

Permainan 2048, meskipun tampak sederhana, memiliki kompleksitas yang cukup tinggi dalam hal strategi dan pemilihan langkah yang optimal. Permasalahan utama dalam menyelesaikan permainan 2048 adalah bagaimana memilih langkah yang tepat di setiap saat sehingga dapat mencapai ubin bernilai 2048 atau lebih tinggi sambil meminimalkan risiko kebuntuan. Berikut adalah beberapa aspek utama dari permasalahan ini:

- Kemunculan Ubin Baru: Setiap langkah yang diambil oleh pemain menghasilkan ubin baru dengan nilai 2 atau 4 yang muncul di lokasi acak di grid. Ini menambah elemen ketidakpastian dan membuat prediksi langkah menjadi lebih sulit.

- Pilihan Langkah yang Banyak: Pada setiap posisi dalam permainan, pemain memiliki empat kemungkinan gerakan (atas, bawah, kiri, kanan). Memilih langkah yang salah dapat mengakibatkan kehilangan peluang untuk menggabungkan ubin dan meningkatkan risiko kebuntuan.

- Kombinasi Ubin: Strategi yang baik harus mempertimbangkan bagaimana menggabungkan ubin dengan cara yang paling efisien untuk menghasilkan nilai yang lebih tinggi tanpa mengorbankan fleksibilitas pergerakan di masa depan.

- Manajemen Ruang: Memastikan bahwa grid tidak penuh adalah kunci untuk kelangsungan permainan. Pemain harus menjaga ruang kosong agar memiliki lebih banyak pilihan langkah di masa depan.

2. Strategi Penyelesaian Permainan

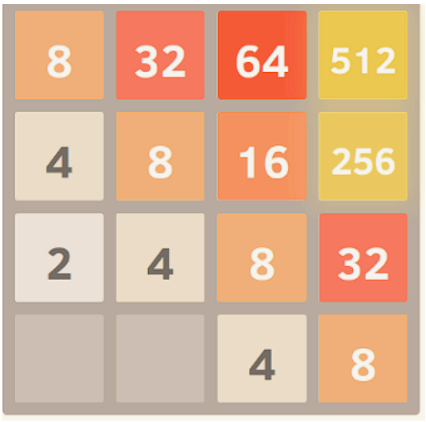
Dari segi cara bermain sebenarnya permainan ini cukup sederhana. Akan tetapi, untuk mencapai skor yang cukup tinggi tidaklah mudah. Berdasarkan pengalaman bermain, terdapat beberapa strategi yang dapat digunakan untuk mendapatkan skor tertinggi.

1. Jangan terfokus dengan nilai ubin yang besar

Memang pada dasarnya kita akan mendapatkan skor yang tinggi apabila memiliki nilai ubin yang cukup besar. Akan tetapi, kita perlu hati hati. Kadang-kala selalu mengejar ubin yang besar justru akan membuat kita terjebak pada situasi sulit di masa yang akan datang.

2. Susunlah ubin dari pojok

Untuk mempertahankan permainan agar ubin tidak cepat penuh, kita dapat membuat struktur ubin terlebih dahulu. Struktur yang akan mempermudah kita adalah dengan membentuknya dari salah satu pojok yang kita inginkan.



Gambar 2. Ilustrasi strategi 2
(source: dokumen pribadi)

Dengan menyusun ubin dengan struktur seperti ini, kita menghindari munculnya ubin bernilai kecil yang bisa saja muncul di tengah tengah ubin besar. Jika hal itu terjadi, maka akan menjadi sulit karena ubin kecil tersebut jadi tidak bisa dijumlahkan dengan ubin di sekelilingnya. Justru malah akan membuat penuh ruang yang tersedia.

3. Rencanakan pergerakan

Sebetulnya dalam permainan ini dibutuhkan kesabaran dan kemampuan perencanaan yang baik. Kita harus bisa berpikir panjang terhadap dampak apa saja yang akan kita dapatkan jika kita melakukan pergerakan tertentu. Sekali melakukan pergerakan tanpa perencanaan dapat mengakibatkan rusaknya struktur dari ubin yang telah kita susun. Hal ini justru akan mempersulit kita dalam membentuk struktur yang ideal kembali.

Setiap pergerakan diusahakan memiliki makna. Tujuan utama kita pada tahap tertentu adalah untuk membentuk struktur ideal seperti susunan ubin pada gambar.

4. Lakukan pergerakan yang menghasilkan keuntungan

Setiap pergerakan juga diusahakan untuk selalu menghasilkan jumlah ubin yang tergabung maksimal. Tentu dengan memerhatikan

3. Implementasi Algoritma Greedy

Untuk mengatasi permasalahan di atas, penulis mengimplementasikan algoritma greedy yang menggunakan berbagai heuristik untuk menentukan langkah terbaik pada setiap saat. Berikut adalah

tahapan implementasi algoritma greedy untuk permainan 2048:

1. Definisikan semua komponen greedy
    - Himpunan kandidat

      C = {“left”, “right”, “up”, “down”}

    - Himpunan solusi

      S = {x1, x2, … , xn}, xi ∈ C.

      Pada kondisi ini terdapat nilai ubin yang telah mencapai 2048

    - Fungsi solusi

      Dapat didefinisikan sebagai fungsi yang memeriksa apakah kondisi sudah mencapai solusi atau justru game over.

    - Fungsi seleksi (selection function)

      Dalam implementasi ini, kita menggunakan heuristik yang memaksimalkan skor saat ini dan jumlah ruang kosong. Serta dengan mengutamakan perpindahan ke salah satu pojok ruang yang memiliki nilai ubin terbesar.

    - Fungsi kelayakan (feasible)

      Dalam konteks permainan 2048 kita akan memeriksa apakah gerakan yang dipilih akan menghasilkan gerakan yang valid atau tidak. Gerakan valid didefinisikan sebagai gerakan yang menghasilkan perubahan.

    - Fungsi obyektif

      Kita akan memaksimalkan penjumlahan dari nilai pertambahan ubin dan jumlah ruang kosong yang tersedia pada masing masing calon pergerakan. Nilai tambahan juga diberikan sebagai pertimbangan agar gerakan masih menuju salah satu pojok ubin yang memiliki nilai terbesar.

2. Definisikan heuristik

Heuristik adalah metode yang digunakan untuk membuat proses pencarian solusi lebih efisien dengan mengurangi ruang pencarian. Dalam konteks permainan 2048, berbagai heuristik greedy dapat diterapkan untuk memandu pemain dalam memilih langkah yang dapat memaksimalkan skor atau menjaga struktur grid yang memungkinkan penggabungan ubin lebih lanjut. Pada implementasi kali ini kita akan mencoba strategi greedy dengan menggunakan beberapa pendekatan heuristik seperti berikut,

- Maximize Immediate Score: Pilih langkah yang menghasilkan skor tertinggi pada langkah tersebut.
- Corner Strategy: Usahakan agar ubin dengan nilai tertinggi tetap berada di salah satu sudut grid.
- Empty Space Maximization: Pilih langkah yang menghasilkan jumlah ruang kosong terbanyak setelah bergerak

3. Evaluasi Langkah

Untuk setiap langkah yang mungkin (atas, bawah, kiri, kanan), hitung nilai heuristik yang sesuai. Implementasi ini akan mengevaluasi setiap langkah berdasarkan heuristik yang telah didefinisikan.

4. Pilih Langkah Terbaik

Dari hasil evaluasi langkah, pilih langkah dengan nilai heuristik tertinggi. Lakukan langkah tersebut dan ulangi proses evaluasi.

5. Iterasi dan Simulasi

Lakukan iterasi langkah-langkah di atas hingga permainan berakhir (kebuntuan atau mencapai ubin 2048). Simulasikan beberapa kali untuk mendapatkan rata-rata kinerja dari algoritma greedy.

IV. Implementasi

Setelah melakukan analisis terkait permasalahan yang ada, pada bagian ini akan dibahas terkait implementasi dari strategi yang telah dibahas sebelumnya. Implementasi dilakukan dengan menggunakan bahasa pemrograman python.

1. Persiapan model permainan

   Model permainan dibuat dengan memanfaatkan library dalam python yaitu *pygame*. Dengan menggunakan *pygame* kita dapat membuat GUI sederhana untuk memvisualisasikan model permainan 2048. Adapun beberapa hal yang perlu dibuat untuk keperluan pembuatan model game 2048 seperti, Model *board/grid*, Model ubin (t*ile*), Algoritma pembangkitan ubin dengan nilai 2 atau 4 pada ruang yang kosong, Fungsi untuk mengatur gerak dari ubin ketika ada event tertentu.

   Pada mulanya model permainan didesain untuk gerakan manual dari keyboard. Hal ini bertujuan untuk memudahkan dalam pengujian permainan yang sudah dibuat agar sesuai dengan spesifikasi yang seharusnya.

2. Implementasi komponen greedy

   Komponen algoritma greedy yang telah dirancang sebelumnya diimplementasikan menjadi program agar dapat diuji keberhasilannya.

    - Himpunan kandidat

Dalam implementasinya, himpunan kandidat disimpan sebagai sebuah dictionary (*Map*).

```
moves = {
    "left": "left",
    "down":"down",
    "right": "right",
    "up": "up"
    }
```

Gambar 3. Implementasi himpunan kandidat
(source: dokumen pribadi)

- Himpunan solusi

  Sementara untuk himpunan solusi tidak secara eksplisit dituliskan dalam program. Himpunan solusi lebih didefinisikan kepada suatu kondisi ketika nilai ubin telah mencapai tujuan yang dicapai yaitu 2048.
- Fungsi solusi

```
def end_move(tiles):
    for tile in tiles.values():
        if(tile.value == 16):
            return "win"
    if len(tiles) == 16:
        return "lose"

    row, col = get_random_pos(tiles)
    tiles[f"{row}{col}"] = Tile(
                            random.choice([2, 4]),
                            row,
                            col)
    return "continue"
```

Gambar 4. Implementasi fungsi solusi
(source: dokumen pribadi)

Implementasi fungsi solusi pada program dibuat sebagai pengecekan apakah kondisi saat ini telah mencapai solusi. Fungsi ini juga sekaligus bertugas mengecek apakah kondisi saat ini telah mencapai kondisi "*Game Over*" atau permainan masih berlanjut dengan ditandai munculnya ubin dengan nilai 2 atau 4 di posisi acak yang kosong.

- Fungsi seleksi

  Fungsi seleksi ini diimplementasikan sebagai fungsi *heuristic*. Sesuai dengan namanya, fungsi ini akan menghitung nilai heuristik dari setiap kandidat pergerakan (*move*) yang diberikan. Skema pembentukan nilainya adalah sebagai berikut
  - Setiap pergerakan akan dihitung nilai penjumlahan yang terjadi apabila gerakan tersebut dilakukan. Nilai tersebut akan disimpan sebagai *weight*.

Gambar 5. Ilustrasi penerapan heuristic
(source: dokumen pribadi)

UP → *weight* = 4
LEFT → *weight* = 16
RIGHT → *weight* = 16
DOWN → *weight* = 4

  - Setiap pergerakan akan dihitung jumlah ubin yang akan kosong ketika penjumlahan tersebut dilakukan. Nilai jumlah ubin kosong akan disimpan sebagai *empty_cells*.

UP → *empty_cells* = 11
LEFT → *empty_cells* = 11
RIGHT → *empty_cells* = 11
DOWN → *empty_cells* = 11

Seluruh *empty_cells* bernilai sama karena semua pergerakan menghasilkan satu perubahan ubin.

  - Setiap pergerakan akan dihitung pembobotan terhadap nilai ubin yang ada di pojok ruang. Semakin besar nilai ubin maka semakin besar bobot yang dihasilkan.

```
max_tile = Tile(-999, -1, -1)
for tile in sorted_tiles:
    if tile.value > max_tile.value:
        max_tile = tile

if((max_tile.row==0 and max_tile.col==0)
   and (direction=="left" or direction=="up")):
    weight += max_tile.value*2.5
elif((max_tile.row==0 and max_tile.col==3)
     and (direction=="right" or direction=="up")) :
    weight += max_tile.value*2.5
elif((max_tile.row==3 and max_tile.col==0)
     and (direction=="left" or direction=="down")):
    weight += max_tile.value*2.5
elif((max_tile.row==3 and max_tile.col==3)
     and (direction=="right" or direction=="down")):
    weight += max_tile.value*2.5
```

Gambar 6. Implementasi penerapan heuristic 2
(source: dokumen pribadi)

Untuk skema pembobotan diimplementasikan dengan memprioritaskan pergerakan menuju pojok apabila di pojok

tersebut memiliki nilai ubin yang cukup besar.

- Fungsi kelayakan
  Fungsi kelayakan diimplementasikan sebagai fungsi *boundary_check*. Fungsi ini digunakan pada fungsi *move* dan fungsi *heuristic*. Tujuannya adalah untuk memastikan pergerakan yang akan dilakukan selalu valid dan tidak akan menabrak batas ruang 4x4 yang dibentuk.
- Fungsi objektif
  Untuk fungsi ini diimplementasikan sebagai pengurutan nilai *weight* + *empty_cells* dari tiap tiap kandidat *move*. Pergerakan yang memiliki nilai maksimum akan dipilih sebagai solusi maksimum lokal (digunakan sebagai penggerak ke kondisi berikutnya).

3. Implementasi keseluruhan
   Setelah didefinisikan seluruh komponen algoritma greedy, maka gabungkan seluruh fungsionalitas tersebut menjadi satu program utuh. Gabungan program ini harapannya dapat menyelesaikan permainan 2048 dengan algoritma greedy.

Program Penyelesaian Permainan 2048

`Inisiasi game model`
`Inisiasi tiles`
`Inisiasi move`
`while(run) do`
`  move ← greedy_decision(moves, tiles)`
`  result ← move(tiles, move)`
`  if(result= "win" or result = "lose")`
`do`
`  // kondisi kalah dan menang`
`  // program berhenti`

Tabel 1. Implementasi program
(source: dokumen pribadi)

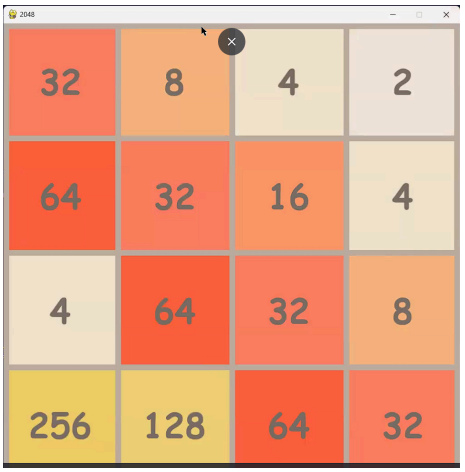
Implementasi greedy
func greedy_decision(moves, tiles) → move // dapatkan semua nilai heuristik dari tiap moves → move dengan nilai heuristic paling tinggi

Tabel 2. Implementasi program
(source: dokumen pribadi)

V. Uji Coba

Uji coba terhadap program yang telah dibuat dilakukan beberapa kali. Hal ini dilakukan karena untuk memastikan konsistensi dari program. Perolehan yang dihasilkan dari uji cukup variatif. Mulai dari kasus terburuk hingga kasus terbaik. Akan tetapi, dari berbagai uji program, program yang dibuat masih belum bisa menyentuh nilai ubin 2048.

1. Kasus terbaik
   Kasus ini dapat menyentuh nilai ubin maksimal di angka 256



Gambar 7. Hasil uji kasus terbaik
(source: dokumen pribadi)

Sebenarnya pada kasus ini hanya tinggal sedikit langkah lagi bisa menyentuh angka 512. Akan tetapi, kemunculan 4 di pojok kiri bawah cukup menyulitkan proses proses selanjutnya.

2. Kasus terburuk
   Sebetulnya pada kasus ini dapat menghasilkan ubin maksimal dengan nilai 128. Akan tetapi, struktur dari susunan ubin bukanlah yang diharapkan. Ini disebabkan oleh kemunculan angka 2 di pojok kiri bawah di tengah tengah proses. Hal ini menyebabkan ubin tersebut mati (tidak bisa dijumlahkan), memenuhi kapasitas ruang, serta menyulitkan membuat susunan yang diharapkan.

Gambar 8. Hasil uji kasus terburuk
(source: dokumen pribadi)

3. Kasus yang paling sering terjadi

Dari sekian banyak percobaan yang dilakukan, kasus yang paling sering terjadi itu berada pada nilai ubin maksimal 128. Hal ini menjadi ukuran bahwa program yang dibuat telah stabil memperoleh nilai ubin maksimal paling tidak 128 pada permainan 2048 ini.

Gambar 9. Hasil uji kasus yang sering terjadi (source: dokumen pribadi)

VI. Kesimpulan dan Saran

Permainan 2048 adalah permainan logika berbasis matematika yang memerlukan perencanaan dalam memainkannya. Mirip dengan halnya bermain catur. Kita tidak bisa dengan asal menggerakkan ke mana arah pergerakan kita ke kondisi berikutnya.

Penggunaan algoritma greedy nampaknya kurang memberikan hasil yang optimal. Kemungkinan terbesar kegagalan algoritma greedy dalam memecahkan permainan ini adalah algoritma ini kurang mampu untuk melihat implikasi jangka panjang dari setiap langkah yang diambil. Hal ini membuat algoritma ini menghasilkan langkah berdasarkan informasi lokal tanpa mempertimbangkan konsekuensi global. Sementara untuk menyelesaikan permainan 2048 perlu perencanaan yang matang untuk memilih langkah yang optimal dengan memperhatikan konsekuensi di kondisi di masa yang akan datang.

Dari kegagalan ini penulis juga telah mempelajari algoritma yang sekiranya cocok untuk menyelesaikan permainan 2048. Algoritma alpha-beta pruning adalah algoritma AI yang perlu dicoba untuk memecahkan permainan 2048. Algoritma ini dapat menghitung nilai kemungkinan terbaik dan nilai kemungkinan terburuk pada suatu node. Hal ini akan sangat membantu jika ingin membuat pohon pencarian solusi dengan mencari bobot maksimal dan melakukan pruning pada node yang tidak memenuhi ketentuan.

Karena keterbatasan waktu dan cakupan materi, penulis masih belum dapat mencoba mengimplementasikan algoritma tersebut. Diharapkan untuk pembaca dapat mencoba mengimplementasikan penyelesaian permainan 2048 menggunakan algoritma alpha-beta pruning sebagai hasil pembanding.

VII. Ucapan Terima Kasih

Segala puji dan syukur penulis panjatkan kepada Tuhan Yang Maha Esa, karena hanya dengan berkat dan rahmat-Nya, penulis dapat menyelesaikan makalah ini dengan baik. Ucapan terima kasih yang tulus juga penulis sampaikan kepada Ibu Dr. Nur Ulfa Maulidevi, S.T, M.Sc., selaku dosen pengampu mata kuliah IF2211 Strategi Algoritma kelas K02, atas ilmu dan bimbingan yang telah diberikan sehingga penulis dapat menyelesaikan makalah ini. Selain itu, penulis juga ingin menyampaikan terima kasih kepada kedua orang tua yang selalu memberikan dukungan dan motivasi.

VIII. Lampiran

Video Link at Youtube

link youtube

Mohon maaf karena proses upload video youtube mengalami kendala

References

[1] Cirulli, G. (2014). "2048". [Online]. Available: https://gabrielecirulli.github.io/2048/

[2] Markovich, J. (2014). "The Fascinating Mathematical Science Behind 2048". [Online]. Available: https://www.popularmechanics.com/culture/gaming/a10388/the-fascinating-mathematical-science-behind-2048-16514546/

[3] Franco, L. (2014). "The Game Design of 2048". [Online]. Available: https://www.gamasutra.com/blogs/LarsFranco/20140416/214340/The_Game_Design_of_2048.php

[4] Norvig, P. (2012). "Paradigms of Artificial Intelligence Programming: Case Studies in Common Lisp". Morgan Kaufmann.

[5] Pearl, J. (1984). "Heuristics: Intelligent Search Strategies for Computer Problem Solving". Addison-Wesley.

[6] Russell, S., & Norvig, P. (2010). "Artificial Intelligence: A Modern Approach". Prentice Hall.

[7] Filipowicz, Luke 2023. The best 2048 Strategy to get your high score!, diakses pada 11 Juni 2024 pada laman https://www.imore.com/2048-tips-and-tricks

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, 12 Juni 2024

Muhamad Rifki Virziadeili Harisman
13522120