Penerapan Algoritma Greedy Untuk Menyelesaikan Game Snake

Jovan Karuna Cahyadi 13518024
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13518024@std.stei.itb.ac.id

***Abstract*—Snake adalah nama umum untuk konsep permainan video dimana pemain mengendalikan sebuah garis yang tumbuh memanjang, dengan garis itu sendiri menjadi rintangan utama. Algoritma greedy adalah jenis algoritma yang menggunakan pendekatan penyelesaian masalah dengan mencari nilai maksimum (atau minimum) sementara pada setiap langkahnya.**

Keywords—Snake, Classic game, Greedy, Game Snake.

I. PENDAHULUAN

Game snake merupakan game yang hampir semua orang tahu. Game ini merupakan game klasik yang diciptakan sekitar tahun 1970 pada arcade, kemudian game tersebut dapat dimainkan pada hp nokia dan menjadi salah satu game terpopuler saat itu. Permainan ini cukup sulit untuk dimainkan karena semakin banyak makanan yang dimakan maka ular akan bertambah panjang, sedangkan kita tidak boleh menabrak tubuh ular kita sendiri. Dengan adanya kemajuan ilmu komputer permainan ini dapat diselesaikan dengan penggunaan komputer tersebut menggunakan algoritma-algoritma yang telah diajarkan pada mata kuliah Strategi Algoritma.

Salah satu algoritma yang dapat digunakan untuk menyelesaikan berbagai macam permainan ialah algoritma Greedy. Dengan penggunaan algoritma greedy ini, dapat dicari solusi untuk menyelesaikan game snake. Dalam makalah ini akan dilakukan penyelesaian game snake yang memiliki ukuran bidang 10x10 dengan menggunakan algoritma greedy.

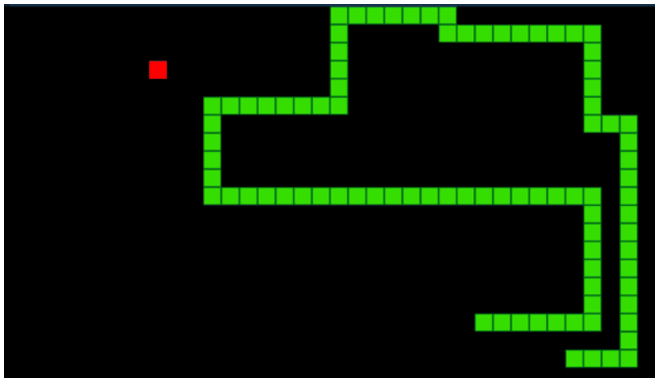
II. DASAR TEORI

A. *Snake*

Snake adalah nama umum untuk konsep permainan video dimana pemain mengendalikan sebuah garis yang tumbuh memanjang, dengan garis itu sendiri menjadi rintangan utama. Game sederhana ini diciptakan pada akhir tahun 1970 pada arcade. Arcade sendiri adalah platform video game seperti di bioskop-bioskop untuk game tekken. Sejak diciptakannya, kepopularannya terus meningkat dan akhirnya terkenal sebagai game klasik. Lalu sejak dirilis di Handphone Nokia pada tahun 1998, kepopularannya terus meningkat.

Permainan ini sangat simpel yaitu sebagai pemain, kita akan mengendalikan sebuah makhluk yang menyerupai ular makanya game ini disebut snake yang dalam bahasa Indonesia artinya ular. Makhluk ini akan bergerak mengitari sebuah bidang berbentuk kotak, dengan tujuan mengambil makanan yang aslinya berbentuk titik atau kotak. Selama bermain, si pemain harus berusaha untuk tidak menabrak dinding atau ekornya sendiri dan itu akan semakin susah, karena setiap kali si pemain memakan makanan, ekornya akan bertambah panjang. Kontrol-pun sangat mudah, yakni hanya atas, bawah, kiri dan kanan, ular akan berjalan secara otomatis dan tidak dapat dihentikan.

Snake ini di publish dan dibuat oleh Gremlin pada tahun 1976, game ini di desain dan dibuat untuk game arcade. Pembuatan game snake di komputer pertama kalinya terjadi pada tahun 1978 dan di program oleh Peter Trefonas dengan judul Worm.



Gambar 1. Game snake
(Sumber : https://tilcode.blog/2019/05/15/ngu-update-1-a-twist-to-the-classic-snake-game/ diakses pada 3 Mei 2020)

B. *Greedy*

Algoritma *Greedy* adalah algoritma yang membangun solusi potong demi potong. Hal ini membuat solusi yang diambil adalah optimum lokal, sehingga bisa jadi bukan merupakan optimum global. Seperti contoh dalam persoalan *knapsack*, dengan strategi *Greedy* bisa dilakukan dengan mengambil barang yang memiliki harga dibagi berat paling tinggi terlebih dahulu, atau bisa juga mengambil barang yang memiliki nilai jual lebih dahulu, atau dengan strategi lainnya.

Contoh lain dalam algoritma *Greedy* adalah seperti membuat strategi pada suatu permainan. Semisal ada sebuah permainan poker, kita bisa melakukan strategi melakukan fold jika tidak ada pasangan sama sekali dengan kartunya dan melakukan call saat ada kartu berpasangan di tangan.

Dalam kehidupan sehari-hari, banyak terdapat persoalan yang menuntut pencarian solusi optimum. Persoalan tersebut dinamakan pesoalan optimasi (optimization problems). Pesoalan optimasi adalah persoalan yang tidak hanya mencari sekedar solusi, tetapi mencari solusi terbaik (best). Solusi terbaik adalah solusi yang bernilai minimum atau maksimum dari sekumpulan alternatif solusi yang mungkin. Contohnya menentukan lintasan terpendek dalam sebuah graf, menentukan total keuntungan maksimum dari pemilihan beberapa objek, dan sebagainya. Pada persoalan optimasi, kita diberiakn sejumlah kendala (constraint) dan fungsi

optimasi. Solusi yang memenuhi semua kendala adalah solusi layak (feasible solution). Solusi layak yang mengoptimumkan fungsi optimasi disebut solusi optimum.

Algoritma Greedy membentuk solusi langkah per langkah (step by step). Terdapat banyak pilihan yang perlu dieksplorasi pada setiap langkah solusi. Oleh karena itu, pada setiap langkah harus dibuat keputusan yang terbaik dalam menentukan pilihan. Keputusan yang telah diambil pada suatu langkah tidak dapat diubah lagi pada langkah selanjutnya. Sebagai contoh, jika kita menggunaka algoritma Greedy untuk menempatkan komponen di atas papan sirkuit (circuit board), sekali sebuah komponen telah ditetapkan posisinya, komponen tersebut tidak dapat dipindahkan lagi.

Pendekatan yang digunakan di dalam algoritma Greedy adalah membuat pilihan yang "tampaknya" memberikan perolehan terbaik, yaitu dengan membuat pilihan optimum lokal (local optimum) pada setiap langkah dengan harapan bahwa sisanya mengarah ke solusi optimum global (global optimum).

Berangkat dari pendekatan yang disebutkan diatas, kita dapat mendefinisikan algoritma Greedy sebagai berikut:

Algoritma Greedy adalah algoritma yang memecahkan masalah langkah per langkah, pada setiap langkah:

1. Mengambil pilihan yang terbaik yang dapat diperoleh pada saat itu tanpa memperhatikan konsekuensi ke depan (prinsip "take what you can get now!")
2. Berharap bahwa dengan memilih optimum lokal (local optimum) pada setiap langkah akan berakhir dengan optimum global (global optimum)

Pada setiap langkah di dalam algoritma Greedy kita baru memperoleh optimum lokal (local optimum) menjadi optimum global (global optimum). Algoritma Greedy mengasumsikan bahwa optimum lokal merupakan bagian dari optimum global.

Skema Umum Algoritma Greedy Persoalan optimasi dalam konteks algoritma Greedy disusun oleh elemen-elemen sebagai berikut:

1. Himpunan kandidat, C.
   Himpunan ini berisi elemen-elemen pembentuk solusi. Contohnya adalah himpunan koin, himpunan job yang akan dikerjakan, himpunan simpul di dalah graf, dan lain-lain. Pada setiap langkah, satu buah kandidat diambil dari himpunannya.

2. Himpunan solusi, S.
   Himpunan ini berisi kandidat-kandidat yang terpilih sebagai solusi persoalan. Dengan kata lain, himpunan solusi adalah himpunan bagian dari himpunan kandidat.

3. Fungsi seleksi
   Dinyatakan dengan predikat SELEKSI, yaitu dungsi yang pada setiap langkah memilih kandidat yang paling memungkinkan mencapai solusi optimal. Kandidat yang sudah dipilih pada suatu langkah tidak pernah dipertimbangkan lagi pada langkah selanjutnya. Biasanya setiap kandidat, x, di-assign sebuah nilai numerik, dan fungsi seleksi memilih x yang mempunyai nilai terbesar atau memilih x yang mempunyai nilai tekecil.

4. Fungsi kelayakan (feasible)
   Dinyatakan dengan predikat LAYAK, yang memeriksa apakah suatu kandidat yang telah dipilih dapat memberikan solusi yang layak, yakni kandidat tersebut bersama-sama dengan himpunan solusi yang sudah terbentuk tidak melanggar kendala (constraints) yang ada. Kandidat yang layak dimasukkan ke dalam himpunan solusi, sedangkan kandidat yang tidak layak dibuang dan tidak pernah dipertimbangkan lagi.

5. Fungsi obyektif
   Fungsi yang memaksimumkan atau meminimumkan nilai solusi (misalnya panjang lintasan, keuntungan, dan lain-lain).

```
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
```

Gambar 2. Skema umum algoritma Greedy
(Sumber : http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2019-2020/Algoritma-Greedy-(2020).pdf diakses pada 3 Mei 2020)

Dengan kata lain, persoalan optimasi yang diselesaikan dengan algoritma Greedy melibatkan pencarian sebuah himpunan bagian S, dari himpunan kandidat C; yang dalam hal ini, S harus memenuhi beberapa kriteria yang ditentukan, yaitu menyatakan suatu solusi dan S dioptimisasi oleh fungsi objektif.

Kata "optimisasi" dapat berarti minimisasi atau maksimasi, bergantung pada persoalan yang dipecahkan. Semua algoritma Greedy mempunya skema umum yang sama.

Secara umum, skema algoritma Greedy dapat kita rumuskan sebagai berikut:

1. Inisialisasi S dengan kosong,

2. Pilih sebuah kandidat (dengan fungsi SELEKSI) dari C,
3. Kurangi C dengan kandidat yang sudah dipilih dari langkah 2 di atas,

4. Periksa apakah kandidat yang dipilih tersebut bersamasama dengan himpunan solusi membentuk solusi yang layak atau feasible (dilakukan oleh fungsi LAYAK). Jika YA, masukkan kandidat tersebut ke dalah himpunan solusi; Jika TIDAK, buang kandidat tersebut dan tidak perlu dipertimbangkan lagi.

5. Periksa apakah himpunan solusi sudah memberikan solusi yang lengkap (dengan menggunakan fungsi SOLUSI). Jika YA, berhenti (selesai); Jika TIDAK, ulangi lagi dari langkah 2.

Pada sebagian masalah, algoritma Greedy tidak selalu berhasil memberikan solusi yang benar-benar optimum.

Meskipun demikian, jika jawaban terbaik mutlak (benar-benar optimum) tidak diperlukan, maka algoritma Greedy sering berguna sebagai solusi yang menghampiri (approximation) optimum, daripada menggunakan algoritma yang lebih kompleks untuk menghasilkan solusi yang eksak.

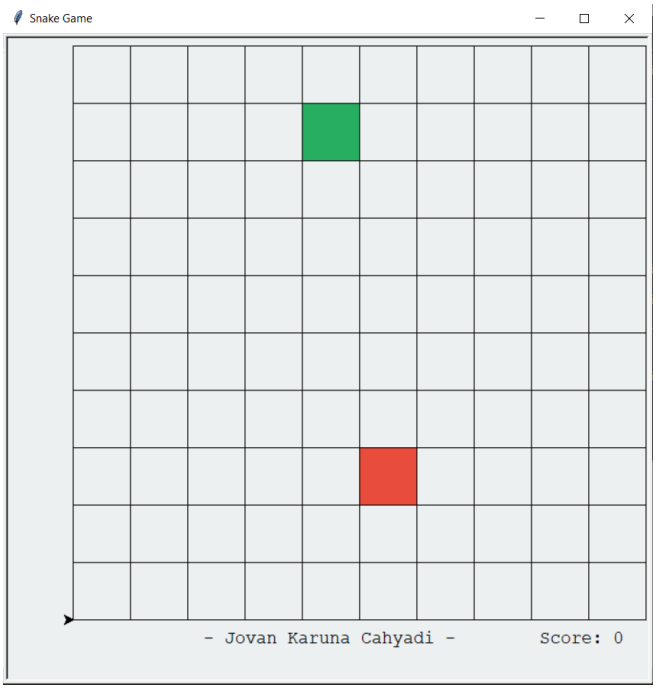
Bila algoritma Greedy berhasil menemukan solusi yang eksak, maka algoritma tesebut harus dapat dibuktikan secara matematis menghasilkan solusi optimum. Jika pembuktian matematis dapat ditunjukkan, maka secara tipikal algoritma Greedy tersebut menjadi metode pilihan untuk kelas masalah yang diberikan.

Jika algoritma Greedy tidak berhasil menemukan solusi optimum untuk suatu masalah, maka sebagai alternatifnya, kita dapat menggunakan metode exhaustive search terhadap semua kemungkinan solusi untuk menemukan solusi optimum. Algoritma Greedy biasanya lebih cepat dibandingkan exhaustive search karena ia tidak mempertimbangkan seluruh alternatif solusi.

III. PEMBAHASAN

Dari dasar teori diatas kita dapat mengetahui bahwa algoritma greedy membentuk solusi langkah per langkah (step by step). Terdapat banyak pilihan yang perlu dieksplorasi pada setiap langkah solusi. Oleh karena itu, pada setiap langkah harus dibuat keputusan yang terbaik dalam menentukan pilihan.

Kita juga mengetahui bahwa untuk menyelesaikan game ini maka snake yang kita kendalikan harus memakan makanannya dan tumbuh sampai semua bidang di layar terisi. Untuk game snake kali ini akan digunakan bidang 10x10 jadi untuk memenangkan game ini maka ular perlu memiliki badan sebanyak 10x10 yaitu 100 dan perlu memakan makanan sebanyak 99 makanan untuk bertambah panjang dari tubuh yang hanya memiliki 1 kotak menjadi 100 kotak. Selain itu, saat memainkan ular tidak boleh sampai keluar dari bidang 10x10 dan kepala ular tidak boleh sampai menabrak bagian tubuhnya yang lain.

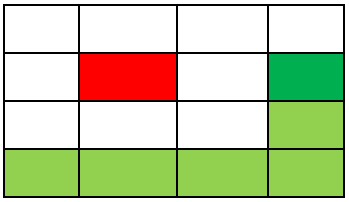


Gambar 3. Tampilan awal game snake yang dibuat dengan bidang 10x10

Dengan penggunaan algoritma greedy kita dapat membahas berbagai cara untuk mendapatkan solusi dalam penyelesaian game snake ini. Salah satu cara yang akan dibahas adalah greedy by shortest path to food.

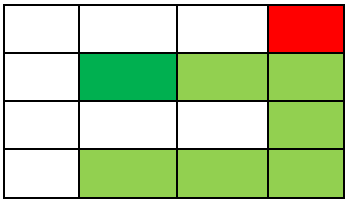
A. Greedy by Shortest Path to Food

Algoritma greedy ini akan membuat snake mencari path yang paling dekat untuk sampai ke makanan. Cara ini meminimalkan waktu untuk menyelesaikan game snake ini karena snake akan berusaha untuk selalu mencari path terdekat untuk mendapatkan makanannya. Ilustrasi untuk greedy by shortest path to food dapat dilihat dibawah ini.

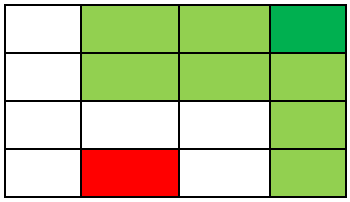


Pada ilustrasi ini digunakan game snake dengan bidang 4x4 dengan menyatakan bahwa kotak yang berwarna merah adalah makanan dan yang berwarna hijau adalah ular yang dimainkan dengan yang hijau tua sebagai kepala untuk digerakan dan yang warna hijau muda adalah badan dari ular itu yang bergerak mengikuti arah kepala ular bergerak.

Dari ilustrasi diatas dilihat bahwa makanan ular ada di kotak kolom kedua dan baris kedua, dalam makalah ini akan dipersingkat menjadi kotak(2,2) dengan 2 didepan sebagai baris dan 2 dibelakang sebagai kolom. Diketahui juga kepala ular berada di kotak(2,4). Dengan pengunaan algoritma greedy by shortest path to food maka akan dicari path terdekat untuk kepala ular menuju ke makanan ular yaitu dengan kepala ular berjalan 2 kali ke kiri melewati kotak(2,3) lalu ke kotak(2,2) dimana makanan berada.



Pada ilustrasi kedua ini dapat dilihat ular melakukan langkah 2 kali ke kiri dan memakan makanannya. Algoritma ini sangat meminimumkan waktu dari ular ke makanannya karena mengambil path terdekat. Namun, walaupun algoritma greedy ini sangat meminimumkan waktunya, yang kita ingin bahas adalah apakah algoritma greedy by shortest path to food ini dapat menyelesaikan game snake ini. Mari kita lanjutkan untuk ilustrasi diatas ini, dapat dilihat setelah memakan makanannya ular akan menambah jumlah badannya dan makanannya akan muncul kembali di dalam bidang dimana tidak kotak tersebut tidak terisi. Langkah yang diperlukan ular untuk memakan makanannya yaitu, ular perlu naik dan berjalan ke kanan sebanyak 2 kali melewati path kotak(1,2), kotak(1,3), dan kotak(1,4) untuk sampai ke makanannya.

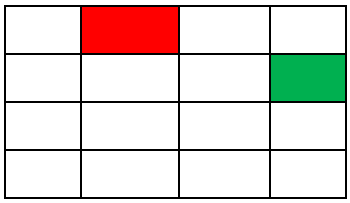


Dari ilustrasi diatas ini diketahui bahwa selanjutnya ular perlu berjalan menuju kotak(4,2) sedangkan kepala ular berada pada kotak(1,4). Dengan algoritma greedy by shortest path to food maka salah satu solusinya adalah untuk bergerak kebawah sebanyak 3 kali dan ke kiri sebanyak 2 kali. Namun, dapat dilihat dari ilustrasi tabel saat kepala ular berada di kotak(1,4), ular tidak dapat bergerak ke kiri maupun ke bawah karena akan menabrak bagian tubuhnya.

Maka dari itu dapat disimpulkan bahwa algoritma greedy by shortest path to food ini tidak pasti akan menciptakan solusi untuk setiap game snake yang dimainkan, walaupun algoritma greedy by shortest path to food ini sangatlah meminimumkan waktu dari ular ke makanannya namun algoritma ini hanya menentukan path dari kotak ular saat itu berada sampai ke makanannya dan tidak menentukan path selanjutnya apakah masih bisa menyelesaikan solusi atau tidak.

B. Greedy by Longest Path to Food

Algoritma greedy ini akan membuat snake mencari path yang paling panjang dan lama untuk sampai ke makanan. Cara ini memang tidak meminimalkan waktu untuk menyelesaikan game snake ini karena snake akan berusaha untuk selalu mencari path terpanjang untuk mendapatkan makanannya, namun dengan mencari path terpanjang maka badan ular akan mengikuti arah kepala ular mengitari tempat permainan snake sehingga ada kemungkinan badan ular akan membuka path setelah ular memakan makanannya. Ilustrasi untuk mencari path dengan algoritma greedy by longest path to food dapat dilihat dibawah ini.



Dari ilustrasi diatas dilihat bahwa makanan ular ada di kotak kotak(1,2) dan kepala ular berada di kotak(2,4). Dengan pengunaan algoritma greedy by longest path to food maka akan dicari path terpanjang dari kepala ular menuju ke makanan ular. Dengan penggunaan algoritma greedy by shortest path mungkin hanya memerlukan ular untuk naik 1 kali dan bergerak ke kiri 2 kali untuk mencapai makanan, namun dengan algoritma greedy by longest path to food akan digunakan langkah yang diilustrasikan dibawah ini.

14		2	1
13	12	3	
10	11	4	5
9	8	7	6

Ilustrasi diatas menyatakan angka yang merupakan langkah dari ular menuju ke makanan dengan angka 1 sebagai langkah pertama ular dan bergerak sesuai angka yang berada pada ilustrasi tersebut. Algoritma greedy by longest path to food ini berhasil untuk mendapatkan path dari ular ke makanannya, walaupun menggunakan waktu yang cukup lama karena perlu mencari path terpanjang. Namun, apakah solusi algoritma ini dapat menyelesaikan game snake ini sampai selesai, hal ini dapat dilihat pada ilustrasi berikut.

10	1	2	3
9			4
8	7	6	5

Pada ilustrasi diatas dapat dilihat bahwa kepala ular berada pada kotak(2,2) dan makanan berada pada kotak(1,4) dengan algoritma greedy by longest path to food, path terpanjangnya adalah dengan bergerak keatas 1 kali dan ke kanan 2 kali, hal ini untuk menghindari tabrakan dengan tubuh ularnya. Dapat dilihat bahwa path ini akan sama dengan algoritma greedy by shortest path to food yang diilustrasikan pada bagian algoritma tersebut, sehingga dapat diketahui bahwa path yang dilalui ini akan gagal untuk mencapai solusi akhir karena setelah memakan makanan pada kotak(1,4) maka ular tidak dapat bergerak karena akan menabrak tubuhnya sendiri.

Maka dari itu dapat disimpulkan bahwa algoritma greedy by longest path to food ini tidak pasti akan menciptakan solusi untuk setiap game snake yang dimainkan. Algoritma ini sama seperti algoritma greedy sebelumnya yang hanya menentukan path dari kotak ular saat itu berada sampai ke makanannya dan tidak menentukan path selanjutnya apakah masih bisa menyelesaikan solusi atau tidak karena telah mengambil path sebelumnya.

C. Greedy by Longest Path in field

Dari algoritma-algoritma greedy sebelumnya diketahui bahwa masalahnya adalah path yang di dapat itu hanya menentukan path dari kepala ular saat itu sampai ke makanannya dan tidak menentukan apakah setelah menggunakan path itu, ular dapat menentukan path lainnya untuk memakan makanannya yang di random di tempat lain.

Maka dari itu kita perlu menentukan path yang setelah path tersebut didapatkan ular dapat memakan makanannya dan selanjutnya mempunyai path yang memiliki solusi untuk menentukan jalan ke makanan selanjutnya. Jadi diperlukan solusi untuk menghentikan ular kita dalam menabrak tubuhnya setelah memakan makanannya dan dapat bergerak ke makanan lain setelah memakan makanan.

Solusinya adalah untuk menentukan sebuah path yang bergerak dari kepala ular dan kembali lagi ke tempat kepala ular tersebut berada setelah mengelilingi seluruh bidang yang ada di game snake ini. Algoritma greedy by longest path in field ini akan membuat path dimana ular tersebut bergerak melewati seluruh kotak yang ada di field ini dan kembali ke posisi awal ular tersebut. Hal ini dikarenakan ular tidak akan menabrak tubuhnya sendiri karena jalur yang dilalui ular akan sama terus sampai game tersebut dapat diselesaikan.

Ilustrasi pencarian path dengan algoritma greedy by longest path in field dapat dilihat dibawah ini.

8	9	14	15
7	10	13	
6	11	12	1
5	4	3	2

Pada ilustrasi diatas ini menyatakan angka sebagai urutan langkah, path diatas ini adalah salah satu path yang dapat digunakan untuk membuat ular bergerak dan kembali lagi ke posisi awalnya. Algoritma ini tidak meminimalkan waktu untuk penyelesaian game karena justru ular akan melewati seluruh kotak yang berada pada bidang, namun karena makanan pasti berada di salah satu kotak pada bidang maka ular pasti akan memakan makanannya saat bergerak mengikuti path.

Algoritma greedy by longest path in field ini akan cukup lama untuk menyelesaikan game ini karena walaupun makanan ada di dekatnya, ular akan tetap mengikuti path yang telah didapatkan pada saat awal game kita mengetahui posisi ular berada. Ular akan menggunakan path ini terus menerus sampai game snake dapat diselesaikan, jadi waktu penyelesaiannya tergantung bagaimana penempatan makanan di dalam kotak ini, semakin banyak makanan yang dimakan selama satu kali perjalanan path maka semakin cepat juga game snake ini dapat diselesaikan

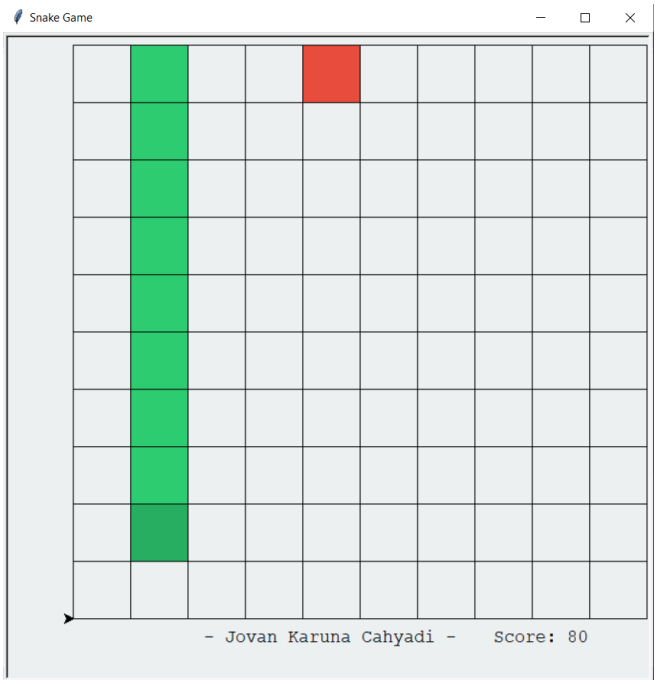
Maka dari itu dapat disimpulkan bahwa algoritma greedy by longest path in field ini pasti akan menciptakan solusi untuk setiap game snake yang dimainkan. Algoritma ini akan menentukan path yang akan diambil ular saat permainan dimulai setelah mengetahui posisi pertama ular tersebut berada, kemudian ular akan menggunakan path tersebut untuk berjalan, jika ular sudah selesai menggunakan path tersebut dan kembali lagi ke posisi awal namun game belum selesai, maka ular akan menggunakan path tersebut kembali sampai game snake ini diselesaikan yang didapat ketika tubuh ular sudah sepanjang kotak yang dimiliki di bidang.

IV. IMPLEMENTASI PROGRAM

Untuk melakukan testing apakah penerapan algoritma greedy by longest pathin field ini dapat menyelesaikan game snake, maka saya membuat game snake sendiri dengan menggunakan bahasa pemrograman python dan dengan bantuan turtle sebagai GUI.

Game snake yang saya buat ini menggunakan bidang kotak 10 x 10 sesuai dengan yang ditulis pada pendahuluan dan selebihnya game snake ini sama seperti game snake pada umumnya. Namun, untuk game snake ini, ular akan bergerak sesuai dengan path yang telah didapat dan tidak dapat digerakkan oleh pemain.

Pada game snake yang dibuat ini juga dapat dilihat kotak berwarna hijau tua sebagai kepala ular dan yang berwarna hijau muda adalah tubuh ular tersebut yang mengikuti kepala ular, serta kotak yang berwarna merah sebagai makanan ular.

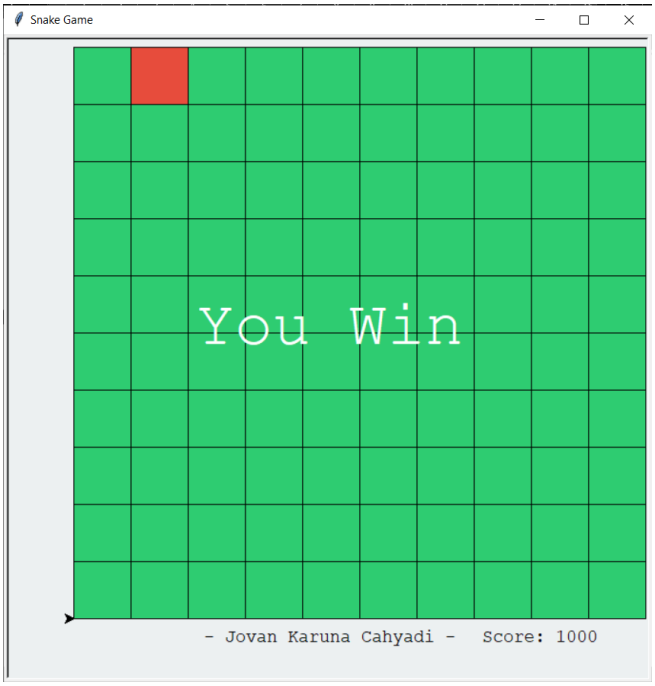


Gambar 3. Tampilan game snake yang dibuat

Untuk melakukan implementasi greedy by longest path in field maka perlu dicari salah satu solusi dari algoritma tersebut yang dapat dilakukan pada bidang 10 x 10, ilustrasi solusi dapat dilihat pada tabel dibawah ini.

5	6	23	24	41	42	59	60	77	78
4	7	22	25	40	43	58	61	76	79
3	8	21	26	39	44	57	62	75	80
2	9	20	27	38	45	56	63	74	81
1	10	19	28	37	46	55	64	73	82
100	11	18	29	36	47	54	65	72	83
99	12	17	30	35	48	53	66	71	84
98	13	16	31	34	49	52	67	70	85
97	14	15	32	33	50	51	68	69	86
96	95	94	93	92	91	90	89	88	87

Dari ilustrasi tabel diatas, dapat dilihat salah satu solusi yang didapat dengan penggunaan algoritma greedy by longest path in field dimana posisi awal ular ada di kotak(6,1) dan posisi makanan ada berada pada kotak(8,8) dan angka yang berada di tabel merepresentasikan urutan langkah yang perlu diambil dari langkah 1 sampai langkah 100. Langkah-langkah ini akan disimpan dalam sebuah array yang disebut path, lalu ular akan akan mengikuti array path tersebut dengan bergerak sesuai langkah yang ada di array, jika sudah sampai akhir array path maka ular akan bergerak sesuai dengan langkah pertama yang berada di array path tersebut, hal ini tidak akan berhenti sampai ular mempunyai tubuh sepanjang kotak yang berada di bidang ini atau 10x10 yaitu 100 yang sudah termasuk dengan kepala ular atau dapat dilihat dari scorenya yaitu 1000 dengan setiap kali memakan makanan akan menaikkan 10 point, jadi perlu memakan 100 makanan untuk menyelesaikan game ini.



Gambar 4. Tampilan game setelah game snake diselesaikan

Pada gambar diatas dapat dilihat bahwa implementasi solusi algoritma greedy by longest path in field dapat menyelesaikan permainan snake 10 x 10 yang saya buat.

V. KESIMPULAN

Kesimpulan yang didapatkan adalah algoritma greedy dapat menyelesaikan permainan game snake yaitu dengan mencari longest path in field dari permainan snake, jadi ular tidak akan menabrak tubuhnya sampai panjang ular sebanyak kotak di field.

Algoritma greedy ini tidak memberikan solusi optimum untuk setiap langkahnya karena tidak mencari path terdekat untuk memakan makanannya melainkan path yang kembali lagi ke posisi awalnya, sehingga untuk memakan makanannya tergantung dengan penempatan makanan yang berada pada game snake ini. Untuk game yang dibuat ini penempatan makanannya di random di tempat yang masih kosong.

LINK VIDEO YOUTUBE

Sebagai pendukung makalah, dibuatlah video yang akan diupload di platform YouTube. Link video tersebut dapat dilihat di https://youtu.be/xcXO6ZvbOMc.

LINK GITHUB

Source code dapat dilihat di link berikut https://github.com/JovanKaruna/SnakeGameSolver.

UCAPAN TERIMA KASIH

Pertama-tama, penulis mengucapkan syukur kepada Tuhan Yang Maha atas segala nikmat yang telah diberikan sehingga penulis dapat menyelesaian makalah ini. Penulis juga ingin mengucapkan terimakasih kepada:

1. Bapak Dr. Ir. Rinaldi Munir atas bimbingan dan pengajaran mata kuliah Strategi Algoritma yang telah mengajar pada K-03.
2. Keluarga yang turut membantu dan mendukung penulis dalam proses pendidikan selama ini.
3. Semua orang yang mempunyai peran besar dalam hidup penulis.

REFERENCES

[1] http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2019-2020/Algoritma-Greedy-(2020).pdf diakses pada 2 Mei 2020 pukul 16.35.

[2] https://towardsdatascience.com/slitherin-solving-the-classic-game-of-snake-with-ai-part-1-domain-specific-solvers-d1f5a5ccd635 diakses pada 2 Mei 2020 pukul 19.23.

[3] http://www.tahupedia.com/content/show/119/Sejarah-dari-Game-Snake-dan-Pada-Saat-Ditamatkan diakses pada 2 Mei 2020 pukul 19.30.

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.

Jakarta, 4 Mei 2020

Jovan Karuna Cahyadi 13518024