Penerapan Algoritma *Depth First Search* (DFS) Dalam Pencarian Solusi Permainan *Connect-Dot*

Go Dillon Audris – 13521062
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13521062@std.stei.itb.ac.id

***Abstraksi*-Connect-Dot merupakan salah satu permainan logika yang mudah dipahami oleh kalangan umum dan melatih cara kerja otak. *Connect-Dot* merupakan suatu permainan di mana pemain diminta untuk menghubungkan suatu titik akhir dengan titik awal dalam satu garis. Satu garis tersebut juga harus melewati semua titik yang ada pada permainan (selain titik awal dan titik akhir). Meskipun sederhana, terkadang terdapat susunan titik yang unik pada permainan, yang menyebabkan pemain harus berpikir terlebih dahulu. Selain itu, terdapat pula susunan titik pada permainan yang tidak memungkinkan adanya garis yang menghubungkan semua titik. Akibatnya, pencarian solusi permainan dengan algoritma *bruteforce* umumnya memakan waktu dan belum tentu memberikan hasil. Selain dengan memanfaatkan algoritma *brute-force*, sebenarnya algoritma pencarian seperti *Depth First Search* juga dapat dimanfaatkan untuk memberikan solusi terhadap permainan ini.**

Keywords*-algoritma, DFS, *Depth First Search*, *Connect-Dot

I. PENDAHULUAN

Permainan merupakan salah satu kegiatan interaktif dan menyenangkan yang dapat dilakukan oleh semua orang untuk menjernihkan pikiran ataupun sekadar mencari kesenangan. Beberapa orang bahkan menjadikan permainan (terutama yang berjenis *video-game*) sebagai pekerjaan untuk mencari nafkah. Tak dapat dipungkiri bahwa permainan sudah menjadi bagian hidup dari banyak orang dan merupakan sarana untuk melepas kelelahan dan menyegarkan pikiran sehabis bekerja atau melakukan kegiatan yang berat.

Permainan sudah ada sejak zaman dulu, yang lebih sering dikenal dengan nama permainan tradisional. Permainan tradisional berkembang dari budaya komunitas yang ada. Di Indonesia, permainan tradisional merupakan salah satu warisan budaya yang diharapkan untuk diteruskan oleh generasi berikutnya. Permainan tradisional dianggap memiliki nilai-nilai luhur dan kebaikan yang dapat dipahami dan dijadikan teladan oleh yang memainkannya. Permainan tradisional juga dianggap memiliki nilai seni atau bahkan nilai magis [1]. Beberapa permainan tradisional yang diketahui oleh masyarakat luas adalah petak umpet, bentengan, congklak, gobak sodor, enggrang, panjat pinang, dan lain-lain. Permainan-permainan tersebut tentunya sangat familiar di telinga oleh banyak orang dan menjadi bagian dari masa kecil mereka. Terdapat pula permainan tradisional dari negara lain seperti *kabbadi* (dari negara-negara Asia Selatan), *charades* (dari Perancis), *tangram* (dari China), dan masih banyak lagi.

Di tahun 1950-an, seiring berkembangnya internet dan komputer, permainan *video-game* muncul dan menyebar. *Video-game* pertama di dunia adalah permainan *tic-tac-toe* yang dikembangkan oleh profesor A. S. Douglas pada tahun 1952 untuk tesisnya di Universitas Cambridge [2]. Pada dua dekade berikutnya, perkembangan permainan *video-game* terus meningkat dan industri *game* semakin marak di pasar global. Sekarang, sudah tidak terhitung berapa banyak *video-game* yang ada di dunia. Tema atau genre permainan *video-game* juga bermacam-macam, mulai dari sekedar *puzzle* biasa hingga tema *role-playing-game* (RPG) yang populer. Beberapa *video-game* yang populer adalah *Elden Ring*, *Genshin Impact*, *DOTA*, *World of Warcraft*, dan lain-lain.

Beberapa permainan mengutamakan fisik, dan memang bermanfaat untuk melatih kekuatan fisik tersebut. Sebut saja permainan seperti panjat pinang, gobak sodor, sepak bola, dan permainan olahraga lainnya. Beberapa permainan yang lain lebih melatih cara kerja dan pola pikir otak agar pemain dapat memenangkan permainan sesuai dengan aturan yang berlaku. Permainan logika masuk ke dalam jenis permainan yang melatih cara pikir otak ini. Permainan logika merupakan permainan yang aturan-aturan bermainnya didasarkan pada logika sederhana yang dapat digunakan untuk menyelesaikan permainan. Permainan logika dipercaya untuk meningkatkan kemampuan otak serta melatih cara pemecahan masalah sederhana. Karena manfaat tersebut, permainan logika sering dikenalkan mulai dari usia dini. Contoh permainan logika adalah Sudoku, *One-Line*, *Minesweeper*, labirin, dan *Connect-Dot*.

Permainan *Connect-Dot* merupakan salah satu permainan logika dengan aturan yang cukup sederhana. Permainan terdiri atas susunan titik-titik. Satu titik berperan sebagai titik awal, dan satu lagi berperan sebagai titik akhir. Permainan selesai ketika pemain berhasil menghubungkan semua titik dari titik awal ke titik akhir hanya dengan satu tarikan garis saja. Susunan titik permainan memiliki banyak variasi, dan ada pula susunan titik yang tidak memiliki solusi penyelesaian. Sering kali ketika pertama kali memainkan *Connect-Dot*, pemain menyelesaikan permainan dengan mencoba semua kemungkinan jalur yang mungkin. Hal ini disebut sebagai algoritma *bruteforce*. Algoritma ini umumnya memakan waktu yang lama untuk menemukan solusi, terlebih jika jumlah titik pada susunan semakin banyak.

Sebenarnya, terdapat algoritma yang lebih mangkus untuk menemukan solusi permainan *Connect-Dot*. Permainan *Connect-Dot* dapat direpresentasikan ke dalam suatu graf dimana setiap simpulnya menyatakan bentuk garis yang telah terbentuk pada susunan titik. Dengan representasi graf ini, maka algoritma traversal graf dapat dimanfaatkan untuk mengunjungi simpul graf dan menemukan solusi penyelesaian.

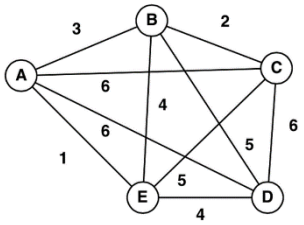
II. LANDASAN TEORI

A. Traversal Graf

Traversal graf merupakan suatu proses pencarian atau kunjungan terhadap simpul-simpul dalam suatu graf secara sistematis. Hal ini umumnya dilakukan untuk memecahkan masalah yang berkaitan dengan pencarian di dalam graf. Masalah tersebut dapat berupa:

- menemukan semua simpul yang dapat dikunjungi di dalam graf,
- menemukan suatu jalur untuk mengunjungi semua simpul di dalam graf, dan
- menemukan jalur terbaik di dalam graf (seperti jalur terpendek untuk perencanaan rute).

Dalam aplikasi di dunia nyata, traversal graf digunakan untuk memecahkan berbagai masalah yang dapat dipetakan menjadi suatu graf. Beberapa masalah tersebut misalnya pencarian rute, penjelajahan *web*, permainan *8-Puzzle*. *missionary and cannibal*, dan lain-lain. Secara umum, setiap masalah tersebut memiliki status-status yang merepresentasikan simpul dalam graf. Traversal graf menjadi proses pencarian status yang sesuai dengan kebutuhan untuk memecahkan masalah.



Gambar 2.1. Ilustrasi graf
(Sumber: https://math.stackexchange.com/questions/1019086/how-to-justify-the-statement-that-a-graph-is-connected)

B. Algoritma Traversal Graf

Algoritma traversal graf merujuk kepada berbagai algoritma yang dimanfaatkan untuk melakukan traversal atau mengunjungi simpul dalam graf untuk mencari solusi penyelesaian masalah, Setiap algoritma memiliki karakteristik dan cara tersendiri untuk melakukan traversal graf atau menentukan simpul mana yang dikunjungi, sehingga memiliki kelebihan dan kelemahannya masing-masing.

Secara garis besar, dalam melakukan traversal graf, simpul-simpul pada graf dibagi menjadi beberapa jenis:

- simpul hidup:
  merupakan simpul pada graf yang dapat dikunjungi oleh algoritma traversal graf. Simpul hidup dibangkitkan dengan cara menemukan semua simpul yang bertetangga dengan simpul ekspan dan yang belum dikunjungi.
- simpul ekspan:
  merupakan simpul yang sedang dikunjungi oleh algoritma traversal graf. Ketika mengunjungi simpul ekspan, algoritma traversal graf umumnya akan melakukan pemrosesan atau validasi apakah simpul ekspan merupakan simpul tujuan atau *goal*.
- simpul tujuan atau *goal*:
  merupakan simpul ekspan yang merepresentasikan status solusi penyelesaian masalah. Status yang direpresentasikan oleh simpul ini akan menjadi bagian dari ruang solusi masalah.

Berdasarkan informasi mengenai permasalahan yang ingin dicari penyelesaiannya, algoritma traversal graf dapat dibagi menjadi dua jenis:

1. Algoritma tanpa informasi (*uninformed / blind search*):
   Algoritma tanpa informasi merupakan jenis algoritma traversal graf dimana algoritma tidak memiliki informasi tambahan mengenai solusi penyelesaian yang ingin ditemukan. Akibatnya, algoritma tidak mampu untuk menerka apakah simpul ekspan sekarang mendekatkan algoritma terhadap simpul tujuan atau tidak [3]. Simpul-simpul yang akan dikunjungi hanya berdasarkan kondisi traversal sejauh tahap tersebut. Beberapa contoh algoritma traversal graf yang tidak memiliki informasi tambahan adalah:
   a. *Breadth First Search* (BFS):
      Breadth First Search merupakan algoritma traversal yang mengunjungi simpul-simpul yang bertetangga dengan simpul ekspan terlebih dahulu. Selanjutnya, simpul yang bertetangga dengan simpul pertama akan dikunjungi terlebih dahulu sebelum lanjut mengunjungi simpul yang bertetangga dengan simpul kedua. Solusi dari algoritma BFS pasti optimal jika biaya solusi sebanding dengan banyak sisi dari simpul awal ke simpul tujuan. [3]
   b. *Depth First Search* (DFS):
      Depth First Search merupakan algoritma traversal yang mengunjungi salah satu simpul yang bertetangga dengan simpul ekspan. Algoritma kemudian diulang mulai dari simpul baru tersebut. Algoritma DFS memanfaatkan proses *backtracking* agar dapat mengunjungi simpul lain yang belum dikunjungi. Solusi dari algoritma ini belum tentu optimal akibat panjang jalur yang ditempuh oleh algoritma DFS bisa tidak terbatas. [3]
   c. *Depth Limited Search* (DLS):
      Depth Limited Search merupakan algoritma traversal variasi dari algoritma DFS. Proses kunjungan simpul pada DLS sama dengan DFS, hanya saja terdapat suatu batasan kedalaman yang dapat dikunjungi oleh algoritma DLS. Hal ini untuk mencegah algoritma DLS terus menerus masuk ke kedalaman yang baru. Algoritma DLS tidak menjamin solusi akan ditemukan dan solusi tersebut, walaupun ada, belum tentu optimal. [4]
   d. *Iterative Deepening Search* (IDS):
      Iterative Deepening Search merupakan algoritma traversal yang mirip dengan algoritma DLS. Hanya saja, algoritma IDS akan dimulai dengan batasan kedalaman 0, dan selama simpul solusi belum ditemukan, maka batasan kedalaman akan terus ditingkatkan. Solusi algoritma IDS pasti optimal. [4]
   e. *Uniform Cost Search* (UCS):
      Uniform Cost Search merupakan algoritma traversal yang banyak digunakan untuk penentuan rute terpendek. Pada keempat algoritma sebelumnya, simpul ekspan dipilih secara FIFO (*first in first out*), di mana simpul ekspan berikutnya mengikuti urutan pembangkitan simpul hidup. Pada algoritma UCS, pemilihan simpul ekspan dilakukan dengan memilih satu simpul hidup yang memiliki biaya atau *cost* terkecil. Biaya ini umumnya berupa jarak dari simpul awal ke simpul hidup. Solusi dari algoritma UCS pasti optimal. [5]

2. Algoritma dengan informasi (*informed search*):
   Algoritma dengan informasi merupakan jenis algoritma traversal graf di mana terdapat informasi tambahan mengenai solusi masalah yang ingin ditemukan, Informasi tambahan ini memungkinkan algoritma dengan informasi untuk memilih simpul ekspan secara lebih cerdas sehingga mendekatkan algoritma dengan simpul tujuan. Informasi tambahan ini umumnya berupa nilai heuristik yang ditentukan lewat pengalaman atau keahlian dalam bidang masalah [3]. Beberapa contoh algoritma traversal graf dengan informasi tambahan adalah:
   a. *Greedy Best First Search* (GBFS):
      Greedy Best First Search merupakan algoritma traversal graf yang memilih simpul ekspan dengan cara yang sama seperti algoritma UCS, yakni simpul hidup dengan biaya terkecil. Hanya saja, biaya terkecil tiap simpul adalah informasi tambahan berupa nilai heuristik dari simpul tersebut ke simpul tujuan. Algoritma GBFS belum tentu menghasilkan solusi optimal. [5]
   b. A* (*A-Star*) :
      A* merupakan algoritma traversal graf yang bisa dibilang merupakan gabungan antara algoritma GBFS dengan UCS. Algoritma A* dibuat dengan maksud mengurangi jumlah iterasi kunjungan simpul (seperti algoritma GBFS) namun dengan tetap menjaga optimalitas dari algoritma UCS. Mekanisme pemilihan simpul pada algoritma A* sama seperti algoritma GBFS dan UCS. Sedangkan biaya atau *cost* tiap simpul dihitung dari penjumlahan biaya dari simpul awal ke simpul tersebut dengan nilai heuristik dari simpul tersebut ke simpul tujuan. Algoritma A* selalu memberikan hasil yang optimal ketika nilai heuristik yang digunakan *admissible* (tidak melebihi atau *over-estimate* nilai jarak sebenarnya). [6]

C. Algoritma Depth First Search (DFS)

Seperti yang telah dibahas pada bagian (B), algoritma *Depth First Search* (DFS) merupakan salah satu algoritma traversal graf tanpa informasi tambahan (*uninformed*). Pemilihan simpul ekspan dilakukan dengan memilih simpul yang bertetangga dengan simpul ekspan sebelumnya. Proses algoritma DFS kemudian dilakukan ulang terhadap simpul tersebut (umumnya secara rekursif). Ketika proses rekursif telah selesai dilakukan pada simpul tersebut, algoritma DFS kemudian memilih simpul lain yang juga bertetangga dengan simpul ekspan sebelumnya dan mengulangi proses rekursif pada simpul tersebut. Agar simpul lain tersebut dapat dikunjungi, maka algoritma DFS umumnya dilengkapi dengan suatu proses *backtracking* agar tetap mampu mengelola simpul-simpul yang belum dikunjungi dari simpul ekspan. Selain dengan memanfaatkan proses rekursif, algoritma DFS juga dapat memanfaatkan struktur data seperti *stack* untuk menyimpan semua simpul hidup yang belum dikunjungi.

Secara garis besar, algoritma DFS perlu melakukan serangkaian tahap berikut untuk melakukan traversal graf:

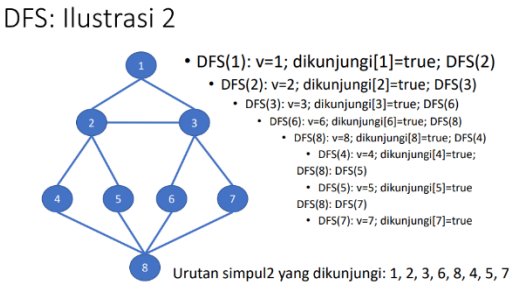
1. Pilih suatu simpul asal atau akar v dan kunjungi simpul tersebut.
2. Pilih simpul w yang bertetangga dengan simpul v.
3. Ulangi kembali algoritma DFS dengan simpul w sebagai simpul akar yang baru.
4. Ketika algoritma DFS mencapai simpul u sehingga tidak ada lagi simpul yang bertetangga dengan simpul u yang belum dikunjungi, maka lakukan proses *backtracking* ke simpul terakhir yang dikunjungi sebelum simpul u sehingga didapatkan simpul bertetangga yang belum dikunjungi.
5. Proses pencarian atau traversal graf berakhir ketika tidak ada lagi simpul yang dapat dikunjugi pada graf.

Untuk melakukan evaluasi terhadap algoritma DFS, terdapat beberapa terminologi yang perlu diketahui. Beberapa terminologi tersebut adalah:

- *branching factor* (b) :
  Jumlah rata-rata simpul hidup yang perlu dibangkitkan setelah mengunjungi suatu simpul ekspan.
- *depth* (d):
  Kedalaman minimum tempat simpul solusi optimal berada. Kedalaman dihitung mulai dari simpul awal atau akar.
- *maximum depth* (m):
  Maksimum kedalaman dari semua simpul yang ada dalam ruang status permasalahan. Kedalaman maksimum ini bisa menyentuh tak hingga.

Evaluasi terhadap algoritma DFS dilakukan dengan melihat beberapa aspek yaitu:

1. *Completeness*:
   Mengacu pada kemampuan algoritma DFS untuk menemukan solusi pada setiap jenis permasalahan yang memanfaatkan DFS. Dalam aspek ini, algoritma DFS merupakan algoritma yang *complete* atau pasti dapat menemukan solusi dari suatu permasalahan dengan catatan bahwa b (*branching factor*) terbatas dan adanya penanganan kasus yang baik. [4]
2. *Optimality*:
   Mengacu pada kemampuan algoritma DFS untuk memberikan solusi yang optimal terhadap permasalahan. Dalam hal ini, algoritma DFS tidak selalu memberikan hasil yang optimal. [4]
3. Kompleksitas Waktu:
   Algoritma DFS memiliki kompleksitas waktu $O(b^m)$. [4]
4. Kompleksitas Ruang:
   Algoritma DFS memiliki kompleksitas ruang O(bm). [4]

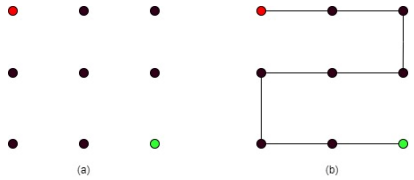


Gambar 2.2. Ilustrasi proses traversal graf dengan algoritma DFS
(Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag1.pdf)

D. Permainan Connect-Dot

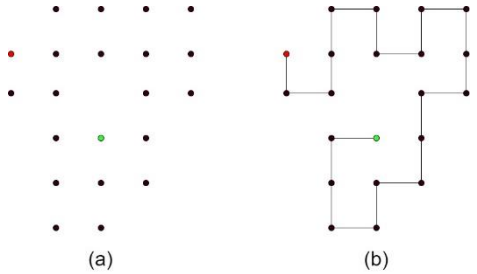
Permainan *Connect-Dot* merupakan suatu permainan logika sederhana di mana pemain dihadapkan dengan suatu susunan titik-titik. Dari kumpulan titik tersebut, satu titik berperan sebagai titik awal, dan satu titik lain berperan sebagai titik akhir. Tujuan dari permainan ini adalah menemukan satu garis pada permainan yang dimulai dari titik awal dan berakhir di titik akhir dan garis tersebut harus melewati semua titik yang ada pada susunan. Batasan pada permainan ini adalah garis yang dibuat tidak boleh menghubungkan dua titik yang terletak

secara diagonal, garis hanya boleh bergerak ke atas, bawah, kiri dan kanan. Garis juga tidak diperkenankan melalui titik yang sama lebih dari 1 kali.

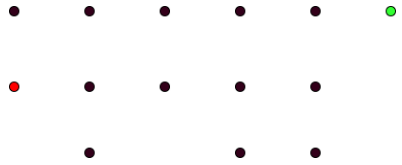


Gambar 2.3. Susunan titik permainan *Connect-Dot* dan salah satu solusinya
(Sumber: Dokumen pribadi)

Gambar 2.3 (a) menunjukkan contoh susunan titik sederhana untuk permainan *Connect-Dot* dan salah satu solusinya. Terlihat bahwa garis lurus ditarik dari titik berwarna merah (titik awal) ke titik berwarna hijau (titik akhir) dan melewati semua titik lain yang ada (titik-titik hitam). Susunan titik pada permainan *Connect-Dot* juga dapat menjadi lebih rumit seperti pada gambar 2.4. Atau bahkan tidak memiliki solusi penyelesaian seperti pada gambar 2.5.



Gambar 2.4. Susunan titik permainan *Connect-Dot* dan satu-satunya solusinya
(Sumber: Dokumen pribadi)



Gambar 2.5. Susunan titik permainan *Connect-Dot* yang tidak memiliki solusi
(Sumber: Dokumen pribadi)

Meskipun demikian, umumnya susunan titik yang tidak memiliki solusi penyelesaian tidak pernah dijadikan sebagai salah satu susunan yang diminta untuk diselesaikan oleh pemain (karena jelas tidak dapat diselesaikan). Permainan *Connect-Dot* sendiri sudah sangat populer dan muncul dalam berbagai representasi dan variasi.

Gambar 2.6. Variasi permainan *Connect-Dot* pada permainan *Flow*
(Sumber: https://cs.stackexchange.com/questions/86674/algorithms-for-solving-flow-game)

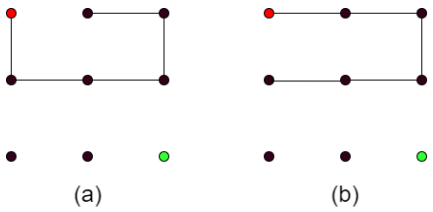
Variasi lain dari permainan *Connect-Dot* adalah adanya lebih dari 1 pasang titik awal dan titik akhir (seperti pada gambar 2.6). Akibatnya akan ada lebih dari 1 garis yang harus diciptakan untuk menghubungkan setiap pasang titik awal dan akhir. Pada gambar 2.6 juga, representasi titik berubah menjadi kotak atau petak yang harus dilewati oleh garis. Terlepas dari itu, batasan terhadap garis yang dibuat tetap sama seperti permainan *Connect-Dot* yang biasa. Oleh karena itu, sebagai batasan implementasi dari makalah ini, permainan *Connect-Dot* hanya memiliki sepasang titik awal dan titik akhir.

III. ANALISIS DAN PENYELESAIAN MASALAH

A. Pemetaan Permainan Connect-Dot Menjadi Elemen Graf

Untuk menggunakan algoritma traversal graf seperti *Depth First Search* (DFS) guna menemukan solusi permainan *Connect-Dot*, maka dibutuhkan suatu langkah atau cara untuk mengubah permainan tersebut menjadi suatu graf yang dapat diproses oleh algoritma DFS. Langkah transformasi permainan menjadi graf dilakukan dengan mendefinisikan representasi simpul serta sisi graf pada permainan. Seperti yang diketahui, bahwa simpul pada graf merupakan representasi dari entitas atau objek lain yang memiliki suatu hubungan, dan sisi pada graf menghubungkan dua simpul yang memiliki hubungan.

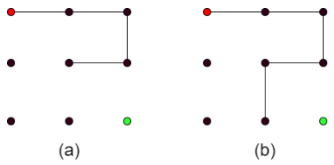
Pada permainan *Connect-Dot* , tentunya garis solusi tidak langsung didapatkan oleh pemain. Terdapat saat ketika garis solusi baru setengah jadi, atau hanya mengunjungi beberapa titik saja. Bisa pula dua garis pada saat yang berbeda tengah mengunjungi titik-titik yang sama, namun dengan arah orientasi atau keterhubungan antar titik yang berbeda. Kondisi garis pada permainan *Connect-Dot* pada setiap saat, yang meliputi titik-titik yang telah terhubung dan bagaimana keterhubungan antar setiap dua titik yang bersebelahan, mewakili status permainan *Connect-Dot* pada saat itu.



Gambar 3.1. Dua status berbeda pada permainan *Connect-Dot*
(Sumber: Dokumen pribadi)

Setiap status pada permainan *Connect-Dot* merepresentasikan sebuah simpul dalam graf. Dengan kata lain, simpul-simpul pada graf representasi Permainan *Connect-Dot* merupakan semua status garis yang mungkin diciptakan dari susunan titik yang diberikan oleh permainan.

Sementara itu, untuk mendefinisikan sisi graf, maka perlu didefinisikan hubungan antara dua status permainan (hubungan antara dua simpul). Dua status permainan dikatakan berhubungan jika satu status dapat dicapai dengan dengan menggerakkan "kepala" garis pada status yang lain ke salah satu titik yang valid untuk dihubungkan. Titik yang valid didefinisikan sebagai titik-titik yang dapat dikunjungi oleh garis dengan bergerak ke atas, bawah, kiri atau kanan.



Gambar 3.2. Keterhubungan antara dua status permainan
(Sumber: Dokumen pribadi)

Sebagai contoh, pada gambar 3.2, terlihat bahwa status kedua (bagian b) dapat dicapai dengan menghubungkan "kepala" garis pada status pertama (bagian a) yang berada di titik tengah dengan titik yang ada di bawahnya. Dengan demikian, terdapat hubungan antara status pertama dengan status kedua. Akan terdapat sisi pada graf representasi permainan *Connect-Dot* yang menghubungkan simpul status pertama dengan simpul status kedua. Sehingga secara umum, sisi-sisi pada graf akan terbentuk di antara dua simpul jika status pada salah satu simpul dapat dicapai dengan menggerakkan garis pada status di simpul lain menuju salah satu titik valid yang dapat dihubungkan oleh garis.

B. Pemetaan Graf Permainan Connect-Dot Menjadi Elemen Algoritma Depth First Search

Setelah graf representasi permainan *Connect-Dot* terbentuk, maka graf tersebut dapat digunakan dalam algoritma DFS untuk melakukan traversal graf dan pencarian solusi. Sebelum algoritma DFS dimanfaatkan dalam graf ini, elemen-elemen algoritma DFS harus didefinisikan terlebih dahulu agar algoritma dapat melakukan traversal graf dengan benar. Elemen algoritma DFS yang dimaksud meliputi simpul awal atau akar, simpul ekspan, simpul hidup, dan simpul tujuan atau *goal*.

Simpul awal atau akar merupakan simpul pertama yang akan dikunjungi dalam proses traversal graf. Jelas bahwa pada simpul awal atau akar, status permainan masih tidak memiliki garis sama sekali, namun "kepala" garis harus berada pada titik yang dipilih sebagai titik awal. Gambar 2.3 (a) dan Gambar 2.4 (a)_merupakan contoh dari simpul awal.

Simpul ekspan merujuk kepada simpul yang sedang dikunjungi oleh proses algoritma DFS. Setiap kali mengunjungi simpul ekspan, maka algoritma akan mengecek apakah simpul tersebut merupakan simpul tujuan atau bukan. Jika simpul tersebut adalah simpul tujuan, maka proses algoritma DFS dapat dihentikan karena solusi telah ditemukan. Jika simpul bukan simpul tujuan, maka algoritma akan melakukan ekspansi dan membangkitkan simpul-simpul baru.

Simpul hidup merujuk kepada simpul yang dibangkitkan oleh algoritma DFS dari suatu simpul ekspan. Simpul hidup belum dikunjungi oleh algoritma DFS, namun menjadi kandidat yang layak untuk dikunjungi dan dijadikan simpul ekspan. Dalam membangkitkan simpul hidup dari simpul ekspan, maka harus terdapat sisi pada graf antara simpul hidup dengan simpul ekspan. Dengan kata lain, simpul hidup yang dibangkitkan oleh algoritma DFS dari suatu simpul ekspan harus merepresentasikan suatu status yang dapat dicapai dari status yang direpresentasikan oleh simpul ekspan tersebut (dengan cara menghubungkan garis ke atas, bawah, kiri atau kanan). Jika merujuk pada Gambar 3.2, ketika simpul pada status pertama (bagian a) merupakan simpul ekspan, maka simpul pada status kedua (bagian b) merupakan salah satu simpul hidup yang dibangkitkan oleh algoritma DFS.

Simpul tujuan atau *goal* merupakan simpul solusi penyelesaian masalah. Status yang direpresentasikan oleh simpul ini adalah suatu garis dari titik awal sampai titik akhir yang melalui semua titik lain pada permainan dengan memperhatikan batasan-batasan garis yang telah ditemukan. Algoritma DFS hanya perlu untuk menemukan satu simpul solusi saja. Dengan demikian, ketika satu simpul solusi telah ditemukan, maka algoritma DFS dapat dihentikan.

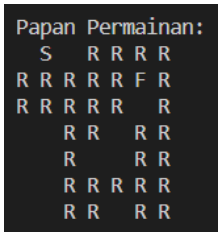
C. Langkah-langkah Algoritma Depth First Search Untuk Menyelesaikan Permainan Connect-Dot

Setelah pemetaan terhadap permainan *Connect-Dot* menjadi graf dan pemetaan graf tersebut menjadi elemen algoritma DFS telah dilakukan, maka dapat dibuat algoritma untuk menyelesaikan permainan *Connect-Dot* sebagai berikut:

- Praproses:
  1. Terima *file* berisi susunan titik pada permainan dan validasi *file* tersebut. Ciptakan suatu matriks susunan titik berdasarkan isi *file* tersebut. Matriks tersebut akan digunakan untuk merepresentasikan simpul atau status permainan.
- Algoritma DFS:
  1. Kunjungi terlebih dahulu titik awal dalam permainan. Dalam hal ini, berarti algoritma DFS sedang mengunjungi simpul akar pada graf. Simpul akar sedang menjadi simpul ekspan.
  2. Lakukan ekspansi terhadap simpul ekspan. Untuk setiap arah dalam urutan atas, kanan, bawah, kiri, jika memungkinkan, lakukan tahap berikut:
     a. Kunjungi titik yang berkorespondensi dengan arah yang dimaksud.
     b. Periksa apakah semua titik sudah dikunjungi atau belum (Ketika semua sudah dikunjungi maka garis solusi permainan sudah terbentuk). Jika sudah, maka tampilkan hasil solusi berupa arah-arah yang harus dilalui oleh garis solusi untuk melewati semua titik. Lanjut ke langkah 3.
     c. Jika semua titik belum dikunjungi, namun titik yang sedang dikunjungi sekarang merupakan titik akhir, maka keluar dari tahapan pemrosesan arah ini. Hal ini karena garis yang diberikan oleh simpul status melanggar batasan, di mana garis seharusnya tidak boleh terhubung ke titik akhir sebelum semua titik lainnya terhubung dahulu. Lanjutkan ke arah selanjutnya.
     d. Selain itu, maka lakukan langkah 2 secara rekursif dengan simpul sekarang menjadi simpul ekspan.
  3. Jika telah didapatkan solusi terhadap susunan permainan *Connect-Dot* yang diberikan, maka tidak perlu dilakukan apapun. Namun, jika seluruh simpul yang mungkin telah dikunjungi oleh algoritma dan tidak adapun yang menjadi simpul tujuan, maka dapat disimpulkan bahwa susunan permainan *Connect-Dot* yang diberikan tidak memiliki penyelesaian.

IV. IMPLEMENTASI DAN PENGUJIAN

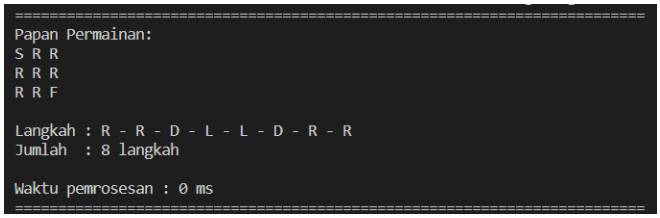
Dengan menerapkan langkah-langkah algoritma penyelesaian permainan *Connect-Dot* yang telah dirumuskan pada bagian III C, telah berhasil dibuat suatu program dalam bahasa Java yang mampu memberikan solusi terhadap permainan *Connect-Dot*. Terdapat beberapa kasus uji dan hasil untuk melakukan pengujian terhadap program. Namun sebelumnya, terdapat beberapa hal yang perlu untuk diketahui agar hasil uji dapat dipahami dengan baik.



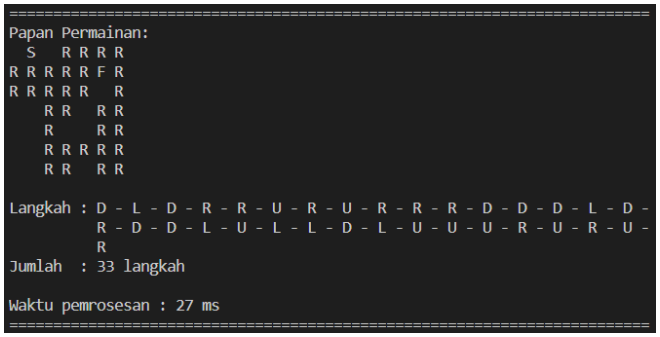
Gambar 4.1. Tampilan susunan titik pada permainan *Connect-Dot*
(Sumber: Dokumen pribadi)

Gambar 4.1 memberikan suatu tampilan susunan titik pada permainan *Connect-Dot*. Untuk membedakan antara titik awal, titik akhir, dan titik-titik lainnya, maka titik direpresentasikan dengan suatu huruf. Huruf S melambangkan titik mulai, huruf F melambangkan titik akhir, dan huruf R melambangkan titik-titik lain yang perlu dikunjungi oleh garis untuk menyelesaikan permainan.

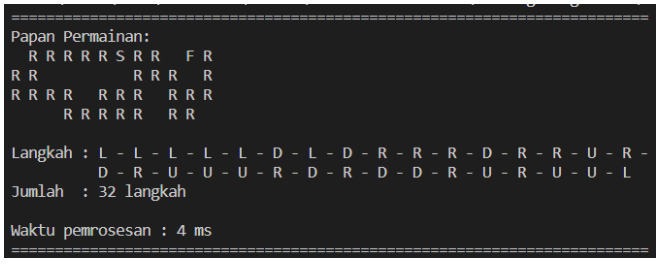
Berikut adalah beberapa kasus uji terhadap berbagai susunan titik permainan *Connect-Dot*. Untuk setiap kasus uji, program akan menampilkan urutan langkah yang harus dilalui oleh garis untuk menghubungkan semua titik, atau memberikan pesan bahwa solusi penyelesaian permainan tidak ada.



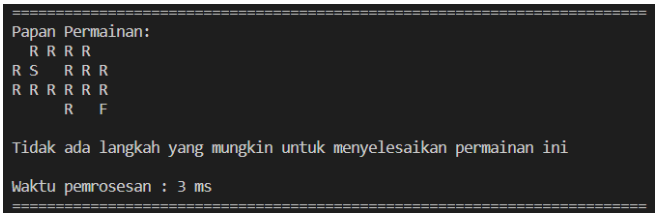
Gambar 4.2. Solusi penyelesaian permainan *Connect-Dot* untuk susunan titik yang sederhana
(Sumber: Dokumen pribadi)



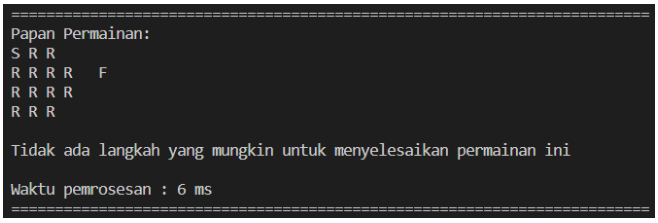
Gambar 4.3. Solusi penyelesaian permainan *Connect-Dot* untuk susunan titik yang kompleks
(Sumber: Dokumen pribadi)



Gambar 4.4. Solusi penyelesaian permainan *Connect-Dot* untuk susunan titik yang kompleks lainnya
(Sumber: Dokumen pribadi)



Gambar 4.5. Tidak ditemukannya solusi penyelesaian permainan *Connect-Dot* untuk susunan titik tertentu
(Sumber: Dokumen pribadi)



Gambar 4.6. Tidak ditemukannya solusi penyelesaian permainan *Connect-Dot* untuk susunan titik lainnya
(Sumber: Dokumen pribadi)

Berdasarkan hasil pengujian terhadap berbagai kasus uji, dapat ditarik keputusan bahwa algoritma yang dibuat telah berhasil untuk menemukan solusi penyelesaian permainan *Connect-Dot* dengan efektif dan tepat. Program berhasil memberikan langkah solusi yang benar ketika solusi memang ada dan memberikan pesan ketika solusi untuk susunan titik tertentu memang tidak ada. Susunan titik yang dijadikan juga cukup bervariasi dan diharapkan telah mewakili jenis kasus lainnya.

Adapun implementasi program telah disusun dan diarsipkan dalam suatu repositori *github*. Untuk akses lebih lanjut serta uji coba program, dapat membuka repositori yang tersedia pada laman berikut :

https://github.com/GoDillonAudris512/ConnectDot_Solver

V. KESIMPULAN

Permainan *Connect-Dot* merupakan salah satu permainan logika sederhana dengan aturan yang mudah dipahami oleh banyak orang. Permainan dilakukan dengan mencari suatu garis yang mampu menghubungkan setiap titik yang ada pada permainan. Umumnya, sebagian besar orang akan mencoba semua kemungkinan garis yang ada untuk menyelesaikan permainan. Namun, algoritma traversal graf seperti *Depth First Search* ternyata dapat dimanfaatkan dan telah dibuktikan untuk menemukan solusi permainan dengan lebih efektif dan optimal. Keberhasilan dibuktikan dengan terciptanya program yang mampu menerapkan langkah algoritma DFS dan memberikan solusi yang tepat untuk berbagai kasus uji permainan. Dengan adanya makalah ini, diharapkan bahwa penyelesaian berbagai masalah (terlebih selain permainan ini) dapat dipandang dari sudut yang berbeda sehingga menghasilkan strategi algoritma yang lebih efisien dan optimal dalam proses komputasi berbagai penyelesaian masalah.

VI. UCAPAN TERIMA KASIH

Puji Syukur kepada Tuhan Yang Maha Esa karena berkat dan kasih karunia-Nya penulis dapat menyelesaikan makalah ini dengan baik dan tanpa kendala yang berarti. Terima kasih juga penulis sampaikan kepada orang tua penulis yang telah mendukung penulis dalam perkuliahan dan dalam penulisan

makalah ini. Tak lupa penulis mengucapkan terima kasih kepada Dr. Nur Ulfa Maulidevi, S.T., M. Sc. selaku dosen pengampu mata kuliah IF2211 Strategi Algoritma 2022/2023 Kelas 02 yang telah memberikan ilmunya kepada penulis sehingga makalah ini dapat ditulis dengan baik. Penulis berharap bahwa pembahasan pada makalah ini tidak berhenti dan dapat dilanjutkan demi kemajuan bangsa dan negara.

REFERENSI

[1] Ayu. 2021. "Olahraga Tradisional: Pengertian, Tujuan, Hingga Sejarah", https://organisasi.co.id/olahraga-tradisional-pengertian-tujuan-hingga-sejarah/. Diakses pada 21 Mei 2023 pukul 12.42.

[2] Team, Editorial. 2021. "Sejarah Video Game: Lika-liku Perjalanan *Video Game* dari Masa ke Masa", https://divedigital.id/sejarah-video-game/#:~:text=Sejarah%20Video%20Game%3A%20Lika-liku%20Perjalanan%20Video%20Game%20dari,Game%203D%20...%206%20Era%20Modern%20Game%20. Diakses pada 21 Mei 2023 pukul 14.11.

[3] Munir, Rinaldi. 2021. "*Breadth / Depth First Search* (BFS/DFS) (Bagian 1)", https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag1.pdf. Diakses pada 21 Mei 2023 pukul 16.28.

[4] Munir, Rinaldi. 2021. "*Breadth / Depth First Search* (BFS/DFS) (Bagian 2)", https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag2.pdf. Diakses pada 21 Mei 2023 pukul 16.32.

[5] Munir, Rinaldi. 2021. "Penentuan Rute (*Route/Path Planning*) (Bagian 1)", https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Route-Planning-Bagian1-2021.pdf. Diakses pada 21 Mei 2023 pukul 16.47.

[6] Munir, Rinaldi. 2021. "Penentuan Rute (*Route/Path Planning*) (Bagian 2)", https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Route-Planning-Bagian2-2021.pdf. Diakses pada 21 Mei 2023 pukul 16.49.

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

Go Dillon Audris
13521062