Penerapan Algoritma Graf untuk Pengembangan Sistem *Gameplay* pada Gim *Idle*

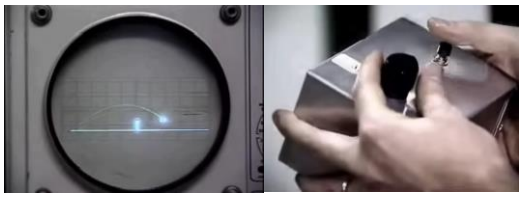
Mayla Yaffa Ludmilla 13523050
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13523050@std.stei.itb.ac.id, [2]maylayaffa@gmail.com

***Abstract*— Gim *idle* atau *incremental game* telah menjadi salah satu genre gim yang populer, dengan karakteristik unik dimana pemain dapat memperoleh kemajuan bahkan ketika tidak aktif bermain. Makalah ini mengeksplorasi penerapan algoritma graf dalam pengembangan sistem *gameplay* pada gim *idle*. Pembahasan dimulai dengan tinjauan historis perkembangan gim dan gim *idle*, dilanjutkan dengan pembahasan konsep dasar teori graf dan algoritma graf seperti Dijkstra dan t*opological sort*. Hasil pembahasan menunjukkan bahwa algoritma graf dapat digunakan secara efektif untuk mengatur sistem progresi pemain, rekomendasi peningkatan infrastruktur, dan jalur pembukaan fitur baru dalam gim *idle*. Implementasi dilakukan menggunakan bahasa pemrograman Python untuk membuat gim *idle* sederhana berbasis CLI dengan sistem pengelolaan pertanian.**

***Kata kunci*—game, *idle*, graf**

I. PENDAHULUAN

Seiring dengan perkembangan teknologi dan komputer sejak pertengahan abad ke-20, industri gim telah menjadi salah satu wujud nyata inovasi digital yang paling berpengaruh. Pada awalnya, komputer adalah sebuah perangkat yang sangat besar yang hanya bisa menyelesaikan persoalan aritmatika sederhana. Pada tahun 1958, video gim yang diciptakan murni untuk tujuan hiburan, Tennis for Two, dikenalkan ke masyarakat. Tennis for Two terdiri dari sebuah monitor yang memungkinkan dua orang untuk 'memukul' sebuah spot cahaya bolak-balik di antara mereka dan alat untuk mengatur pergerakan bola.



Gambar 1.1 Tennis for Two
Sumber: https://youtu.be/6PG2mdU_i8k

Perkembangan besar terjadi pada 1970-an dengan kelahiran industri gim video komersial, ditandai oleh peluncuran *Pong* oleh Atari pada 1972. Selain itu, mesin arcade seperti *Pac-Man* dan *Space Invaders* menjadi fenomena budaya pada akhir 1970-an dan awal 1980-an. Namun, pada 1983, industri gim mengalami krisis akibat pasar yang jenuh dan penurunan kualitas gim. Nintendo berhasil membangkitkan industri ini dengan meluncurkan Nintendo Entertainment System (NES) pada 1985.

Di tahun 1990-an, teknologi 3D dan grafis yang lebih canggih merevolusi pengalaman bermain, dipelopori oleh konsol seperti PlayStation dan Nintendo 64. Saat ini, video gim diakui sebagai bentuk seni, menjadikannya media global yang menjangkau jutaan pemain di seluruh dunia. Gim tidak hanya menjadi sarana hiburan, tetapi juga memiliki peran penting dalam pendidikan, simulasi pelatihan, dan budaya populer.

Perkembangan gim mengakibatkan munculnya berbagai genre gim. Salah satu genre yang mendapatkan popularitas adalah gim *idle* atau *incremental game*. Konsep yang unik dari gim *idle* adalah pemain dapat memperoleh kemajuan dalam permainan bahkan ketika tidak aktif bermain. Muncul pertama kali pada tahun 2002 dengan gim bertajuk Progress Quest yang merupakan projek satir, gim *idle* terus berkembang seiring berjalannya waktu. Beberapa contoh gim baru dari genre ini adalah Cats & Soup yang memungkinkan pemain berperan dalam mengelola penjualan sup oleh kucing-kucing, Idle Research yang mengajak pemain menjalankan fasilitas penelitian dengan sistem otomatisasi yang kompleks, dan Clicker Heroes, salah satu pelopor genre gim *idle* yang *gameplay*-nya berfokus di mengalahkan monster dengan mekanisme klik.

Konsep graf adalah hal yang banyak diimplementasikan dalam pengembangan gim *idle*. Penerapan algoritma graf dapat digunakan untuk merancang sistem upgrade yang kompleks, jalur progresi pemain, manajemen sumber daya, dan interaksi antar elemen permainan. Dengan memanfaatkan struktur data graf, pengembang dapat menciptakan *gameplay* yang lebih dinamis dan memberikan pengalaman bermain yang lebih seru untuk para pemain.

II. DASAR TEORI

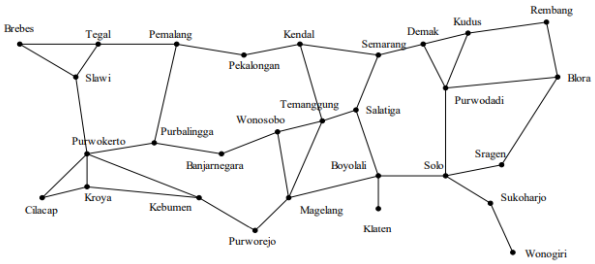
A. Teori Graf

Graf adalah struktur disrit yang terdiri dari simpul (vertex) dan sisi (edge) yang menghubungkan simpul-simpul tersebut. Setiap sisi memiliki satu atau dua simpul yang terhubung dengannya. Simpul ini disebut dengan *endpoint*.

Graf G = (V,E) disusun oleh V, himpunan tak-kosong yang terdiri dari simpul-simpul, dan E, himpunan yang boleh kosong dan terdiri dari sisi-sisi. Dengan kata lain,

sebuah harus memiliki simpul tetapi tidak harus memiliki sisi.

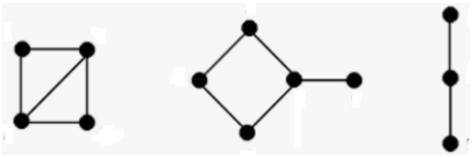
Graf biasa digunakan untuk merepresentasikan beberapa objek distrik yang memiliki hubungan satu sama lain. Sebagai contoh, graf di bawah ini merepresentasikan peta jaringan jalan raya yang menghubungakan sebagian kota di Provinsi Jawa Tengah.



Gambar 2.1 Contoh Graf
Sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf

Ada beberapa jenis-jenis graf. Berdasarkan ada tidaknya gelang atau sisi ganda di dalam graf, ada 2 jenis graf:

1. Graf Sederhana
   Graf sederhana adalah graf yang tidak memiliki gelang ataupun sisi ganda.



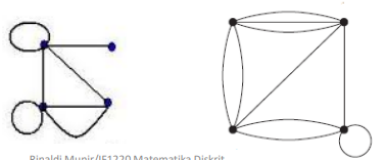
Gambar 2.2 Graf Sederhana
Sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf

2. Graf Tak-Sederhana
   Graf tak-sederhana adalah graf yang memiliki sisi ganda atau gelang.
   Graf tak-sederhana dibagi lagi menjadi dua yaitu:

- Graf Ganda
  Graf yang memiliki sisi ganda disebut graf ganda.

Gambar 2.3 Graf Tak-Sederhana Ganda
Sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf

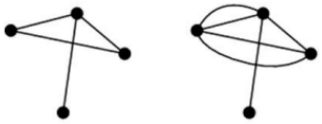
- Graf Semu
  Graf yang memiliki sisi gelang disebut graf semu.



Gambar 2.4 Graf Tak-Sederhana Semu
Sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf

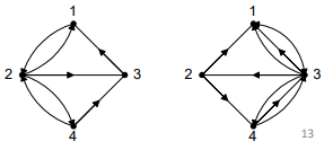
Berdasarkan orientasi arah pada sisi, graf dibagi menjadi dua jenis:

1. Graf Tak-Berarah
   Graf tak-berarah adalah graf yang sisinya tidak memiliki orientasi arah.



Gambar 2.5 Graf Tak Berarah
Sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf

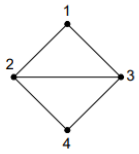
2. Graf Berarah
   Graf berarah adalah graf yang setiap sisinya memiliki orientasi arah.



Gambar 2.6 Graf Berarah
Sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf

Ada beberapa terminologi yang digunakan ketika kita sedang membahas graf, di antaranya:

1. Ketetanggaan
   Dua buah simpul disebut bertetangga apabila keduanya terhubung langsung. Contohnya, di graf G1, simpul 1 bertetangga dengan simpul 2 dan 3 tetapi tidak bertetangga dengan simpul 4.

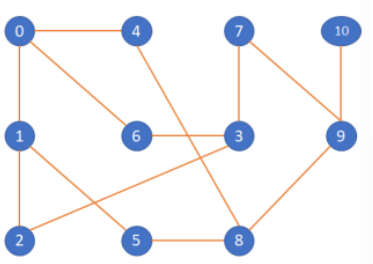


G_1

Gambar 2.7 Graf G1
Sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf

2. Lintasan
   Lintasan sepanjang n dari simpul awal v_0 menuju simpul tujuan v_n dalam graf G merupakan sebuah

rangkaian yang terdiri dari simpul-simpul dan sisi-sisi secara bergantian dengan pola $v_0, e_1, v_1, e_2, v_2, \ldots, v_{n-1}, e_n, v_n$. sedemikian sehingga $e_1 = (v_0, v_1), e_1 = (v_1, v_2), \ldots, e_1 = (v_{n-1}, v_n)$ adalah sisi-sisi dari graf G.



Gambar 2.8 Graf Sederhana G
Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf

Pada graf di atas, : lintasan 0, 6, 3, 7, 9, 10 adalah lintasan dari simpul 0 ke 10 yang melalui sisi (0, 6), (6,3), (3,7), (7, 9), (9, 10).

Jika graf tersebut mengandung sisi ganda, maka setiap sisi e_i perlu dicantumkan dalam lintasan untuk membedakan lintasan satu dengan lainnya. Namun, jika graf merupakan graf sederhana (tidak memiliki sisi ganda), maka penulisan sisi e_i tidak diperlukan. Panjang lintasan adalah jumlah sisi dalam lintasan tersebut. Panjang lintasan 0, 6, 3, 7, 9, 10 pada G adalah 5.

3. Siklus atau sirkuit
   Siklus atau sirkuit adalah lintasan yang berawal dan perakhir pada simpul yang sama. Panjang sikluas adalah jumlah sisi dalam sirkuit tersebut. Pada graf G di gambar 2.8, lintasan 0, 4, 8, 5, 1, 0 adalah sebuah siklus yang memiliki panjang 5.

B. Algoritma Graf

1. Algoritma Djikstra
   Untuk sebuah graf berarah, algoritma Dijkstra adalah salah satu algoritma yang dapat digunakan untuk menemukan bobot jarak terpendek yang dapat ditempuh. Algoritma Dijkstra dapat dilakukan dengan langkah langkah berikut:
   - Pilih simpul awal untuk memulai proses.
   - Berikan nilai bobot atau jarak untuk setiap simpul. Atur nilai 0 untuk jarak dari simpul awal ke dirinya sendiri, sementara simpul lainnya diberikan nilai tak hingga karena jaraknya belum diketahui.
   - Dari simpul awal, periksa semua simpul tetangganya yang belum dilewati. Hitung jarak dari simpul keberangkatan ke setiap simpul tetangga. Jika jarak baru lebih kecil dari jarak yang telah tercatat sebelumnya, perbarui dengan nilai jarak yang lebih kecil tersebut.
   - Setelah semua simpul tetangga selesai diperiksa, tandai simpul keberangkatan sebagai simpul yang telat dilewati. Simpul yang sudah dilewati tidak akan diperiksa lagi, dan nilai jarak terakhir yang disimpan adalah jarak terpendek ke simpul tersebut.
   - Dari simpul-simpul yang belum dilewati, pilih simpul dengan jarak terkecil sebagai simpul keberangkatan berikutnya. Ulangi proses ini sampai semua simpul telah diperiksa.

2. *Topological Sort*
   Topological Sort atau pengurutan topologi adalah teknik pengurutan simpul-simpul pada graf berarah tanpa siklus (*Directed Acyclic Graph*/DAG). Dalam pengurutan ini, setiap simpul u yang memiliki sisi menuju simpul v akan muncul sebelum simpul v dalam urutan. Dengan kata lain, *topological sort* memberikan urutan linear dari simpul-simpul yang merepresentasikan ketergantungan antar elemen pada graf. Hasil dari topological sort tidak unik.

   Salah satu pendekatan yang digunakan untuk membuat algoritma *topological sort* adalah *Depth First Search* (DFS) yang mengecek seluruh simpul yang bertetangga di graf sejauh mungkin sebelum kembali ke simpul sebelumnya dan mengecek tetangga simpul lainnya, jika ada. Berikut adalah langkah langkah menggunakan pendekatan DFS untuk algoritma *topological sort:*
   - Buat *stack* untuk menyimpan hasil
   - Buat *array* untuk menandai simpul yang sudah dikunjungi agar tidak diproses lebih dari satu kali
   - Lakukan DFS pada simpul yang belum dikunjungi, lalu tandai simpul yang telah dikunjugi. Untuk setiap tetangga dari simpul ini, jika tetangga belum dikunjungi, lakukan DFS secara rekursif pada tetangga tersebut.
   - Setelah semua tetangga simpul telah diproses (DFS selesai), tambahkan simpul ini ke *stack*.
   - Karena simpul terakhir yang masuk ke stack adalah simpul yang selesai paling akhir, urutan yang benar akan didapatkan dengan mencetak isi stack dalam urutan terbalik.

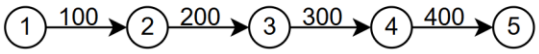
III. PEMBAHASAN

Ada beberapa penerapan konsep graf dalam pengembangan gim *idle*, di antaranya:

1. Level
   Dalam sebagian besar gim, ada status level yang dapat ditingkatkan oleh pemain seiring dengan berjalannya gim. Biasanya, seorang pemain dapat ‘naik’ tingkat atau meningkatkan level mereka ketika mereka sudah memenuhi syarat yang telah ditentukan sebelumnya. Salah satu syarat yang umum digunakan adalah sistem poin. Misalnya, jika pemain, sudah mencapai jumlah poin tersebut,

maka pemain tersebut dapat meningkatkan levelnya.

Jika digambarkan dalam bentuk graf, maka level dapat dibuat menjadi sebuah graf berbobot dan berarah. Simpul dari graf adalah aras yang dapat dicapai oleh pemain. Sedangkan, sisi graf menunjukan hubungan antar-aras. Bobot sisi graf menyatakan poin yang dibutuhkan oleh pemain untuk meningkatkan aras mereka.



Gambar 3.1 Contoh Graf Aras
Sumber: Dokumen Penulis

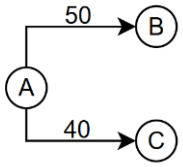
2. Rekomendasi Peningkatan Infrastruktur

Di beberapa gim *idle* seperti Cats & Soup yang memiliki tujuan untuk menghasilkan mata uang agar dapat meningkatkan nilai jual produk dan menukarnya dengan produk lain, sumber daya yang dimiliki dapat ditingkatkan arasnya untuk menghasilkan uang lebih cepat. Seringkali, ada beberapa pilihan sumber daya yang dapat ditingkatkan oleh pemain. Oleh karena itu, beberapa gim menyediakan rekomendasi sumber daya yang mana yang perlu ditingkatkan terlebih dahulu untuk dapat menghasilkan uang dengan lebih cepat.

Gambar 3.2 Contoh Rekomendasi di Gim *Idle*
Sumber: Dokumen Penulis

Jika digambarkan dalam bentuk graf, permasalahan peningkatan infrastruktur ini dapat dibuat dalam bentuk graf berbobot dan berarah. Simpul dari graf melambangkan aksi meningkatkan suatu sumber daya, sedangkan bobot sisi dari graf melambangkan keuntungan yang akan didapatkan oleh pemain apabila memilih untuk meingkatkan infrastruktur di node tujuan.

Sebagai contoh, ada 2 infrastruktur yang levelnya dapat ditingkatkan, yaitu infrastruktur B dan infrastruktur C. Biaya untuk meningkatkan infrastruktur B adalah 1000 koin dan infrastruktur B akan menghasilkan 20 koin/detik. Di sisi lain, biaya untuk meningkatkan infrastruktur C adalah 2000 koin dan infrastruktur C akan menghasilkan 50 koin/detik. Sekilas, meningkatkan infrastruktur B terlihat lebih baik dilakukan terlebih dulu karena biayanya lebih murah. Akan tetapi, jika menghitung waktu yang diperlukan untuk mengembalikan modal meningkatkan infrastruktur, meningkatkan infrastruktur C lebih menguntungkan. Di gambar 3.3, bobot pada graf diisi berdasarkan waktu yang dibutuhkan untuk mengembalikan modal meningkatkan infrastruktur.



Gambar 3.3 Contoh Graf Meningkatkan Infrastruktur
Sumber: Dokumen Penulis

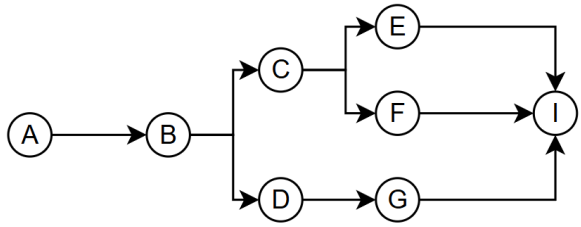
Algoritma Dijkstra dapat digunakan untuk menyelesaikan permasalahan ini dengan mencari jalur paling efektif untuk meningkatkan keuntungan. Dalam kasus ini, akan dipilih jalur A-C sebagai solusi optimal karena memiliki bobot lebih kecil (40) dibandingkan jalur A-B (50). Ini membuktikan bahwa keputusan peningkatan infrastruktur tidak hanya bergantung pada biaya awal, tetapi juga pada efisiensi pengembalian modal yang direpresentasikan sebagai bobot dalam graf.

3. Jalur Pembelian Infrastruktur

Banyak gim *idle* modern yang memiliki fitur-fitur baru yang hanya dapat diakses setelah pemain memenuhi syarat tertentu, seperti jumlah uang atau aras tertentu. Fitur ini dapat berupa infrastruktur ataupun skill yang dapat didapatkan oleh pemain.

Dalam graf berarah tanpa siklus, simpul mewakili fitur atau level baru yang dapat diakses. Sedangkan, sisi menunjukkan ketergantungan antar fitur. Misalnya, fitur B harus dibuka/dibeli sebelum fitur A dapat diakses.

Topological Sort memastikan bahwa fitur-fitur ini diakses secara berurutan berdasarkan ketergantungan yang telah ditentukan, sehingga pemain mendapatkan pengalaman bermain yang terstruktur. Misalnya, dari contoh graf di gambar 3.4 yang melambangkan infrastruktur yang tersedia, dapat dilihat bahwa sebelum membeli infrastruktur B, harus membeli infrastruktur A terlebih dahulu. Oleh karena itu, salah satu urutan berdasarkan *topological sort* adalah A-B-C-D-E-F-G-I. Perlu diingat bahwa hasil *topological sort* tidak unik, jadi ada kemungkinan urutan lain.



Gambar 3.4 Contoh Graf Jalur Peningkatan Infrastruktur
Sumber: Dokumen Penulis

IV. IMPLEMENTASI

Implementasi dibuat menggunakan bahasa python. Penulis membuat sebuah gim *idle* sederhana berbasis *command line interface* (CLI) yang memiliki konsep pengelolaan pertanian. Pada awal permainan, pemain memiliki kandang sapi yang menghasilkan susu dan sepetak tanah yang ditanami serta menghasilkan padi. Susu dan padi akan dihasilkan terus menerus tanpa memerlukan interaksi langsung dengan pemain, sesuai dengan konsep gim *idle*.

Pemain dapat melakukan beberapa kegiatan dalam gim ini. Pertama, melihat *inventory* barang dan infrastruktur yang dimiliki. Kedua, mengakses Shop, tempat untuk jual-beli barang menggunakan koin. Pemain dapat menjual barang yang mereka miliki untuk ditukar dengan koin atau membeli infrastruktur yang tersedia. Terakhir, pemain dapat membuat produk berdasarkan infrastruktur yang telah dimiliki. Setiap infrastruktur memiliki jenis produk yang berbeda-beda. Produk-produk ini dapat dijual di Shop dan ditukar dengan koin. Permainan akan berakhir ketika pemain berhasil membuat produk *mango ice cream*.

Penerapan graf dalam permainan ini adalah dalam implementasi infrastruktur yang ada di dalam permainan. Infrastruktur yang ada harus di-unlock dalam urutan tertentu. Apabila pemain berusaha membeli infrastruktur yang ada sebelum infrastruktur sebelumnya terbeli, akan dimunculkan pesan bahwa pemain harus membeli infrastruktur sebelumnya.

Dalam implementasi, pertama-tama dibuat class untuk *node* (simpul), *edge* (sisi), dan graf. Simpul graf memiliki properti id, nama, dan *unlocked*. *Unlocked* akan diisi oleh nilai boolean yang menyatakan apakah infrastruktur/node tersebut telah di-*unlock* atau belum.

```
class Node:
    def __init__(self, id, name, unlocked):
        self.id = id
        self.name = name
        self.unlocked = unlocked
        return None
```

Gambar 4.1 Class Node
Sumber: Dokumen Penulis

```
class Edge:
    def __init__(self, id, node1, node2):
        self.id = id
        self.node1 = node1
        self.node2 = node2
        return None
```

Gambar 4.2 Class Edge
Sumber: Dokumen Penulis

```
class Graph :
    def __init__(self, list_nodes, list_edges):
        self.nodes = list_nodes
        self.edges = list_edges

    def get_node_by_name(self, node_name):
        for node in self.nodes:
            if node.name.lower() == node_name.lower():
                return node
        return None
```

Gambar 4.3 Class Graph
Sumber: Dokumen Penulis

Setelah itu, sesuai dengan spesifikasi sebelumnya, graf infrastruktur diinisialisasi. Ada 4 infrastruktur yang tersedia, yaitu Bakery, Garden, Smoothies Shop, dan Ice Cream Shop. Infrastruktur harus dibeli dengan urutan Bakery, Garden, Smoothies Shop, dan Ice Cream Shop.

```
def initialize_graph():
    bakery = Node(1, "Bakery", False)
    garden = Node(2, "Garden", False)
    smoothies = Node(3, "Smoothies Shop", False)
    ice = Node(4, "Ice Cream Shop", False)

    edge1 = Edge(1, bakery, garden)
    edge2 = Edge(2, garden, smoothies)
    edge3 = Edge(3, smoothies, ice)

    list_nodes = [bakery, garden, smoothies, ice]
    list_edges = [edge1, edge2, edge3]

    graph = Graph(list_nodes, list_edges)

    return graph
```

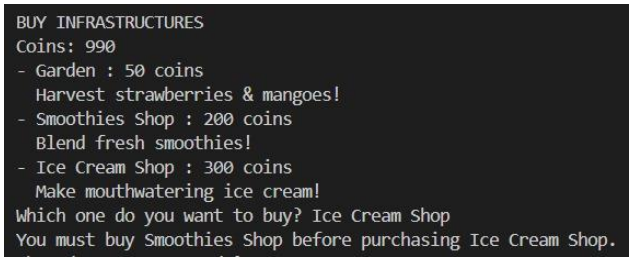
Gambar 4.4 Inisialisasi Graf
Sumber: Dokumen Penulis

Ketika kode dijalankan, akan muncul pesan yang menyambut pemain dan memberikan gambaran umum tentang hal-hal yang bisa dilakukan.

Gambar 4.5 *Display* Menyambut Pemain
Sumber: Dokumen Penulis

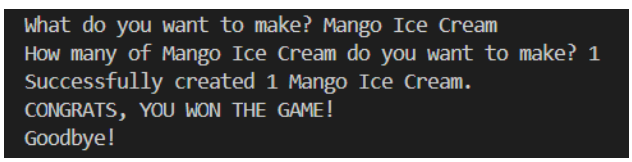
Sesuai spesifikasi, jika pemain berusaha membeli infrastruktur yang ada sebelum infrastruktur sebelumnya terbeli, dimunculkan pesan bahwa pemain harus membeli infrastruktur sebelumnya. Sebagai contoh, di gambar 4.6, pemain hanya memiliki Bakery, tetapi ingin membeli Ice Cream Shop. Oleh karena itu, ditampilkan pesan bahwa pemain harus membeli Smoothies Shop sebelum membeli Ice Cream Shop seperti di gambar 4.7.

Gambar 4.6 Contoh *Inventory* Pemain
Sumber: Dokumen Penulis



Gambar 4.7 Contoh Pesan
Sumber: Dokumen Penulis

Selanjutnya, setelah pemain berhasil membuat *mango ice cream*, permainan selesai.



Gambar 4.8 Pesan Akhir Gim
Sumber: Dokumen Penulis

Kode lengkap untuk implementasi gim *idle* sederhana ini terdapat di GitHub penulis yang dapat dilihat di bagian V makalah.

V. Appendix

Link GitHub implementasi makalah: github.com/maymilla/mini-idle-game
Link video implementasi makalah: www.youtube.com/watch?v=dHLmgleM1-o

VI. Kesimpulan dan Saran

Makalah ini membahas bagaimana konsep graf seperti algoritma Dijkstra dan *topological sort* bisa diterapkan dalam pengembangan *gameplay* untuk gim *idle*. Penerapan dari graf dapat dilihat di sistem kenaikan level, rekomentasi peningkatan infrastruktur, dan jalur peningkatan infrastruktur. Setelah itu, implementasi dengan Python menunjukkan bagaimana konsep graf diterapkan di sebuah gim *idle* sederhana dengan membantu memastikan pemain hanya bisa membuka fitur baru setelah memenuhi syarat tertentu.

VII. Ucapan Terima Kasih

Penulis menyampaikan ucapan terima kasih kepada:
1. Tuhan Yang Maha Esa, yang atas berkat-Nya penulis dapat menyelesaikan makalah ini,
2. Kedua orang tua penulis yang selalu memberikan dukungan kepada penulis,
3. Bapak Rila Mandala selaku dosen pengampu mata kuliah Matematika Diskrit kelas K2 yang telah membimbing penulis dan memberikan banyak ilmu yang bermanfaat selama perkuliahan.

Akhir kata, penulis mengharapkan makalah ini dapat bermanfaat bagi pihak yang membacanya..

Referensi

[1] Rosen, K.H. (2012). Discrete Mathematics and Application to Computer Science 7 th Edition. Mc Graw-Hill.
[2] https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf diakses 3 Januari 2025 pukul 15.30
[3] https://mti.binus.ac.id/2017/11/28/algoritma-dijkstra/ diakses 4 Januari 2025 pukul 08.43
[4] https://www.geeksforgeeks.org/topological-sorting/ diakses 4 Januari 2025 pukul 08.55
[5] https://blog.nationalmuseum.ch/en/2020/01/the-history-of-video-games/ diakses 4 Januari 2025 pukul 9.21
[6] https://mrmine.com/blog/the-evolution-and-origins-of-idle-clicker-and-incremental-games/ diakses 5 Januari 2025 pukul 12.18

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 7 Januari 2025

Mayla Yaffa Ludmilla 13523050