Penerapan Algoritma Prim yang Dimodifikasi untuk Menentukan Rute Penelusuran Titik Lokasi pada Peta Video Game Open-World

Aurelius Marcel Candra 13519198[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13519198@std.stei.itb.ac.id

***Abstrak*—Eksplorasi didefinisikan sebagai usaha untuk menemukan sesuatu yang baru, misalnya bahan tambang, menapaki wilayah yang tidak dikenali, atau menambah catatan baru mengenai suatu hal ke dalam katalog. Eksplorasi atau penjelajahan, baik secara nyata maupun dalam video game, erat kaitannya dengan sebuah peta sebagai penuntun. Pada umumnya, dalam peta ditunjukkan wilayah daratan, perairan, dan batas alam yang dilengkapi dengan lokasi-lokasi penting sebagai penanda. Untuk menjangkau keselurahan lokasi secara efisien, tidak dapat dilakukan secara acak atau sekedar menghubungkan titik-titik terdekat. Penentuan rute yang efisien dapat dilakukan dengan memanfaatkan salah satu algoritma dalam teori pohon, yaitu algoritma Prim, yang dimodifikasi.**

***Kata kunci*— algoritma, pembentukan pohon, pemindaian, rute peta.**

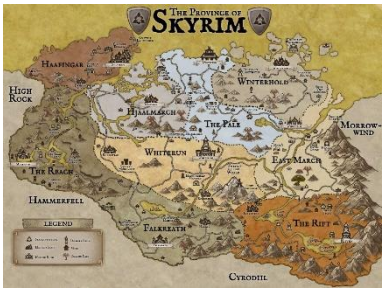
I. PENDAHULUAN

Sebuah peta sejak zaman dahulu sangat bermanfaat bagi manusia untuk keperluan navigasi dan eksplorasi, terutama untuk wilayah yang cakupannya sangat luas. Peta yang baik memudahkan manusia untuk mencapai suatu lokasi tujuan tanpa tersesat. Peta yang baik biasanya tidak hanya berskala tepat dan menampilkan bentuk wilayah daratan, perairan, dan jalan, melainkan juga ditambahkan dengan penanda-penanda (*landmark*) sebagai acuan nyata dalam perjalanan.

Gambar 1: Peta abad eksplorasi
Sumber: *https://blogs.loc.gov/loc/2020/01/free-to-use-and-reuse-maps-of-discovery-and-exploration/*

Pada zaman modern sekarang ini, perjalanan sudah sangat dimudahkan dengan adanya peta berbasis teknologi GPS (*Global Positioning System*). Abad eksplorasi juga sudah sangat lampau sehingga usaha manusia untuk melakukan penjelajahan area baru sudah tidak lagi memungkinkan. Sekarang, eksplorasi tidak perlu jauh-jauh atau susah-susah lagi dilakukan manusia, dengan adanya *video game open-world* yang terus berkembang. Open-world adalah genre atau mekanisme dari video game yang memungkinkan pemain bereksplorasi dalam dunia virtual. Pemain diberi kesempatan untuk bereksplorasi sambil menjalankan suatu objektif sehingga setiap pemain mungkin saja mengambil rute yang beragam. Dengan demikian, tentunya video game open-world dilengkapi dengan fitur peta untuk menuntut pemain.

Perkembangan video game open-world dapat dikatakan ambisius dibuktikan dengan perkembangan ukuran dan kompleksitas bentuk dunia virtual yang terus bertambah dari tahun ke tahun. Dapat ditelusuri kembali, bahwa awal mula konsep open-world dimulai dari tahun 1976 dengan permainan berbasis teks berjudul *Colossal Cave Adventure*. Pada pertengahan tahun 80-an dimulai konsep video game open-world dalam tiga dimensi, seperti *Elite* dan *Mercenary*, lalu disusul dengan konsep *racing game*, seperti *Vette*. Perkembangan open-world setelah tahun 90-an didominasi dengan video game buatan Nintendo yang memutakhirkan grafis dalam tiga dimensi. Melewati milenium kedua, banyak dilahirkan serial video game open-world dengan ukuran yang bertambah luas, di antaranya *Grand Theft Auto*, *Saints Row*, *Just Cause*, *Fallout*, *Far Cry*, *Assassin's Creed*, *Need for Speed*, dan masih banyak lagi. Hingga akhir 2020 ini, judul-judul video game open-world terus bertambah dengan karakteristiknya masing-masing, seperti *Minecraft*, *No Man's Sky*, *Metal Gear Solid*, *Watch Dogs*, *Witcher*, *Horizon Zero Dawn*, dan terakhir *Cyberpunk 2077* yang akan rilis Desember ini.



Gambar 2: Model peta video game open-world, The Elder Scrolls V: Skyrim
Sumber: *https://www.reddit.com/r/inkarnate/comments/d6486y/skyrim/*

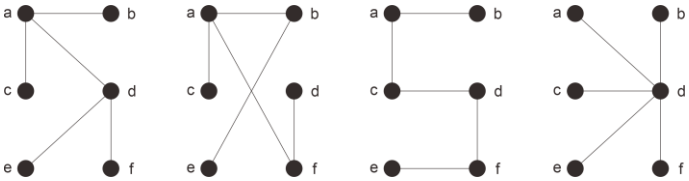
Fitur yang sering diimplementasikan dalam video game open-world adalah teleportasi ke lokasi-lokasi strategis tertentu yang sudah pernah dijelajahi oleh pemain. Teleportasi adalah konsep memindahkan posisi pemain dari satu lokasi ke lokasi lain secara instan. Lokasi-lokasi yang dapat dituju oleh pemain biasanya muncul pada peta sebagai penanda atau landmark. Fitur teleportasi kadang menjadi pisau bermata dua, di satu sisi membantu menghemat waktu pemain untuk kembali ke suatu tempat, tetapi di sisi lain fitur teleportasi "membunuh" mekanisme eksplorasi dalam video game tersebut.

Dalam makalah ini, dibahas mengenai cara-cara atau algoritma untuk menentukan rute eksplorasi dari sebuah peta video game open-world dengan titik-titik sebagai destinasi teleportasi (*waypoint*) yang tersebar di dalamnya. Dari beberapa alternatif yang dibahas, akan ditentukan cara atau algoritma yang dinilai paling efisien untuk diterapkan.

II. Landasan Teori

A. Pohon

Pohon didefinisikan sebagai graf tidak berarah yang terhubung dan tidak terdapat sirkuit di dalamnya. Secara formal, pohon adalah graf G = (V, E), V (*vertices*) adalah himpunan titik-titik pada graf pohon dan E (*edges*) adalah himpunan sisi-sisi yang menghubungkan titik-titik dalam graf pohon tersebut.



Gambar 3: Contoh-contoh graf pohon
Sumber: dokumen penulis

B. Sifat-sifat Pohon

Terdapat teorema yang menyatakan sifat-sifat yang dimiliki oleh graf pohon, yang sekaligus menjadi definisi lain dari graf pohon. Teorema tersebut menyebutkan, misalkan G = (V, E) adalah graf tak-berarah sederhana dan jumlah simpulnya n, maka, pernyataan-pernyataan berikut adalah ekuivalen:

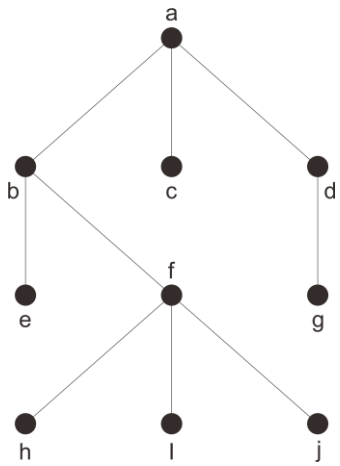
1. G adalah pohon.
2. Setiap pasang simpul di dalam G terhubung dengan lintasan tunggal.
3. G terhubung dan memiliki m = n – 1 buah sisi.
4. G tidak mengandung sirkuit dan memiki m = n – 1 buah sisi.
5. G tidak mengandung sirkuit dan penambahan satu sisi pada graf akan membuat hanya satu sirkuit.
6. G terhubung dan semua sisinya adalah jembatan (jembatan diartikan sebagai sisi yang jika dihilangkan akan menyebabkan graf terpecah menjadi dua komponen).

C. Pohon Berakar

Pohon berakar adalah pohon yang simpulnya diperlalukan sebagai akar dan sisi-sisinya diberi arah sehingga menjadi graf berarah. Akar mempunyai derajat masuk 0 dan simpul-simpul lainnya memiliki derajat masuk 1. Simpul yang memiliki derajat keluar berjumlah 0 disebut daun. Setiap simpul dalam pohon dapat dicapai dari akar dengan sebuah lintasan tunggal. Sesuai konvensi, arah pada sisi dalam graf pohon dapat dibuang karena setiap simpulnya dapat dicapai dari simpul akar. Sembarang pohon tidak berakar dapat dibuat menjadi pohon berakar dengan memilih satu simpul sebagai akar. Hasilnya tidak selalu unik, dikarenakan pemilihan simpul yang berbeda akan menghasilkan pohon berakar yang berbeda juga.

D. Terminologi Pohon Berakar

Ada beberapa terminologi yang dipakai untuk pohon berakar:



Gambar 4: Diagram pohon sebagai ilustrasi untuk terminologi
Sumber: dokumen penulis

1. Anak (*child*) dan orang tua (*parent*)
   Jika sisi-sisi pada graf pohon diberik arah, anak merupakan simpul yang menjadi ujung dari sebuah sisi, sedangkan orang tua adalah simpul yang menjadi pangkal sisi.
   Contoh: Simpul b, c, dan d adalah anak (child / children) dari simpul a. Sedangkan simpul a adalah orang tua (parent) dari simpul b, c, dan d.
2. Lintasan (*path*)
   Lintasan adalah urutan simpul-simpul yang dilalui dari titik awal hingga tujuan. Panjang lintasan adalah jumlah sisi yang dilalui dalam suatu lintasan tersebut.
   Contoh: Lintasan dari simpul a ke simpul g adalah a, d, dan g dengan panjang lintasan 3.
3. Saudara Kandung (*sibling*)
   Simpul yang memiliki orang tua yang sama disebut saudara kandung satu dengan lainnya.
   Contoh: relasi simpul b, c, dan d.
4. Upapohon (*subtree*)
   Upapohon secara formal memiliki definisi terdapat graf G = (V, E) dan G' = (V', E') di mana himpunan V' adalah subset dari himpunan V dan himpunan E' adalah subset dari himpunan E.
   Contoh: Simpul-simpul beserta sisi-sisi antara f,h,i, dan j adalah upapohon.
5. Derajat (*degree*)
   Derajat adalah jumlah anak pada simpul tersebut. Derajat maksimum dari semua simpul merupakan derajat pohon itu sendiri.
   Contoh: Simpul a memiliki derajat 3, simpul d memiliki derajat 2, dan simpul g memiliki derajat 0.
6. Daun (*leaf*)

Daun adalah simpul yang berderajat nol.
Contoh: Simpul c, e, g, h, i, dan j.

7. Simpul dalam (*internal nodes*)
   Simpul dalam adalah simpul yang memiliki anak.
   Contoh: Simpul b, d, e, dan f.
8. Aras (*level*)
   Nilai aras dari akar adalah 0. Nilai aras selain akar dijumlahkan dengan panjang lintasan dari akar ke simpul tersebut.
   Contoh: Aras dari simpul g bernilai 2, sedangkan aras dari simpul j bernilai 3.
9. Tinggi (*height*)
   Tinggi didefinisikan sebagai aras maksimum dari keseluruhan graf pohon. Dengan kata lain, tinggi suatu pohon adalah panjang maksimum lintasan dari akar ke sebuah daun.
   Contoh: Graf pohon pada gambar X memiliki tinggi dengan nilai 3.

E. Pohon Merentang dan Pohon Merentang Minimum

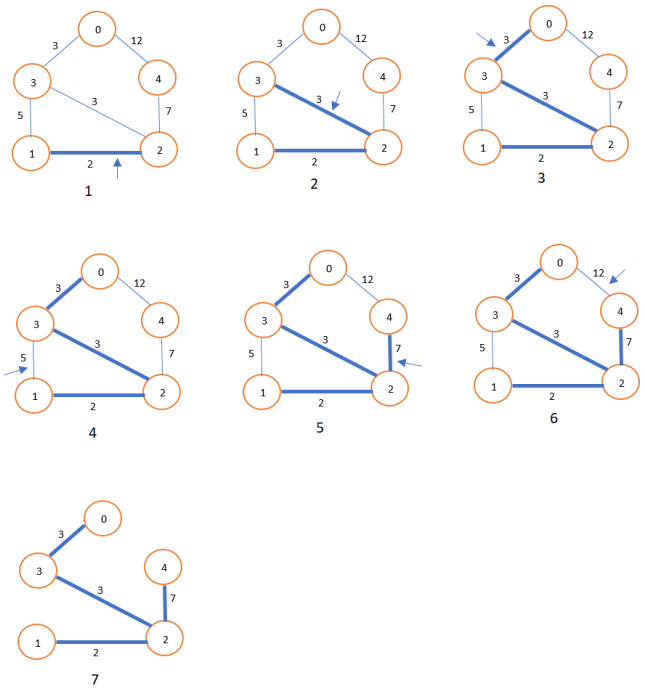
Pohon merentang dapat dibentuk dari graf sembarang. Misalkan terdapat graf G = (V, E) yang merupakan graf tidak berarah terhubung. Graf G dapat diubah menjadi pohon T = (V', E') dengan mengeliminasi seluruh sirkut yang ada dengan cara memutus salah satu sisi sebagai elemen dari sirkuit tersebut. Langkah ini dilakukan hingga tidak ada lagi sirkuit yang tersisa sehingga terbentuk graf pohon T, yaitu pohon merentang. Pohon merentang yang dihasilkan dari proses ini tidak selalu unik dan memungkinkan untuk dihasilkan pohon merentang yang beragam dengan metode atau algoritma yang beragam juga.

Pohon merentang disebut minimum jika graf G sebagai basisnya adalah graf yang berbobot, yaitu terdapat nilai untuk setiap sisi yang menghubungkan tiap simpul, dan sedemikian sehingga pada pohon merentang yang dibentuk memiliki jumlah bobot yang minimum dari himpunan pohon merentang yang mungkin dibentuk. Bobot pohon merentang T dari graf G didefinisikan sebagai jumlah bobot semua sisi di T. Terdapat dua algoritma untuk membangun pohon merentang minimum, yaitu algoritma Prim dan algoritma Kruskal. Keduanya dikategorikan ke dalam algoritma "*greedy*".

F. Algoritma Prim

Algoritma Prim adalah salah satu alternatif untuk membentuk pohon merentang minimum. Inti dari algoritma ini adalah dari sebuah titik awal (acuan) dicari sisi yang memiliki bobot minimum, kemudian pindah ke titik baru dan dilakukan hal yang sama. Langkah ini dilakukan berulang-ulang hingga seluruh titik terhubung dan terbentuk pohon merentang minimum. Langkah algoritma Prim lebih lengkap sebagai berikut:

1. Ambil sisi dari graf G yang berbobot minimum, masukkan ke dalam T.
2. Pilih sisi (u, v) yang mempunyai bobot minimum dan bersisian dengan simpul di T, tetapi (u, v) tidak membentuk sirkuit di T. Tambahkan (u, v) ke dalam T.
3. Ulangi langkah 2 sebanyak (n –2) kali, dengan n adalah banyaknya titik atau simpul dari graf G.



Gambar 5: Simulasi algoritma Prim
Sumber: *https://www.dotnetlovers.com/article/230/minimum-spanning-tree-mst-using-kruskals-algorithm*

III. PEMBAHASAN

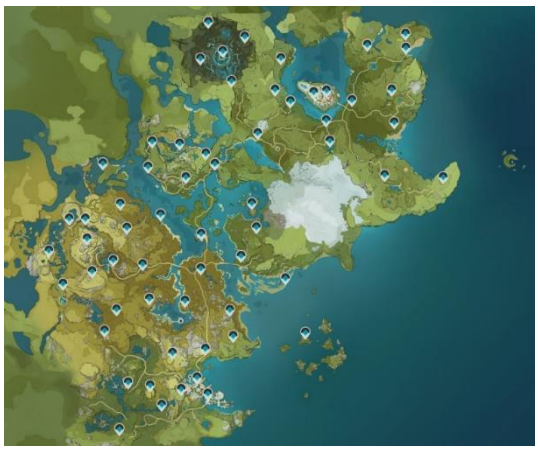
A. Batasan dan Asumsi

Terdapat batasan dan asumsi yang digunakan untuk menguji hasil modifikasi algoritma Prim, yaiti sebagai berikut:

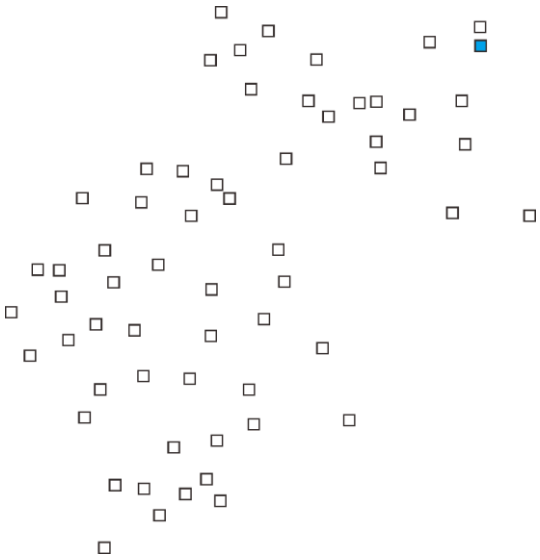
1. Rute eksplorasi diartikan sebagai rute efisien sebatas untuk menjangkau semua titik-titik destinasi teleportasi (*waypoint*), bukan rute untuk eksplorasi keseluruhan dunia virtual video game tersebut.
2. Diasumsikan semua titik-titik destinasi dapat dijangkau dari titik manapun, dengan kata lain tidak ada pembatas atau halangan, misalnya perairan atau gunung, yang mencegah akses dari satu titik ke titik lainnya.
3. Term teleportasi pada analisis algoritma diartikan sebagai pemain dapat kembali ke titik manapun, terkhusus pada titik lokasi yang bercabang, untuk kemudian menelusuri cabang lain yang belum dijelajahi sebelumnya sehingga tidak perlu lagi kembali ke titik akar untuk menelusuri cabang baru.

B. Model Peta

Peta yang digunakan berasal dari video game open-world online, Genshin Impact (dirilis pada September 2020) yang memiliki pesebaran titik-titik (tanda lokasi berwaran hitam-biru) sebagai berikut.



Gambar 6: Model peta yang digunakan
Sumber: *https://progameguides.com/genshin-impact/genshin-impact-how-to-unlock-and-use-fast-travel/*



Gambar 7: Model peta yang disederhanakan, kode warna biru sebagai titik awal (akar keseluruhan pohon)
Sumber: dokumen penulis

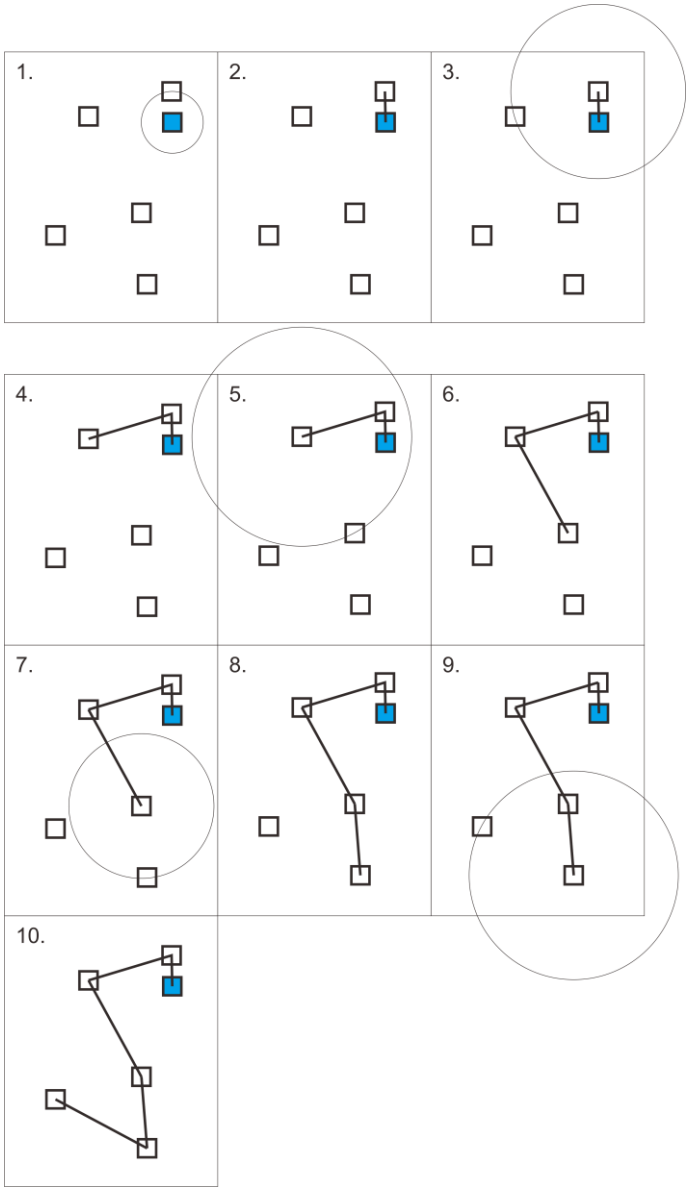
C. Modifikasi Algoritma Prim

Algoritma Prim dimodifikasi sedemikian sehingga tidak lagi diperlukan data akurat yang menunjukkan nilai jarak antartitik dalam sebuah peta. Hal ini dilakukan dengan memanfaatkan mekanisme kerja jangka untuk membentuk lingkaran dengan variasi radius sehingga menyerupai pemindaian melingkar (*radial scanning*) layaknya radar. Dalam pengujian modifikasi algoritma Prim ini, dimanfaatkan aplikasi grafis berbasis vektor, yaitu CorelDraw untuk menerapkan mekanisme pemindaian melingkar tadi. Langkah pemindaian yang dilakukan cukup sederhana, yaitu menentukan suatu titik sebagai pangkal, kemudian titik tersebut dijadikan sebagai titik pusat lingkaran, dan lingkaran yang dibentuk dapat ditambah radiusnya hingga beririsan dengan titik baru. Basis algoritma Prim yang tidak dihilangkan adalah adanya titik acuan awal dan mekanisme pencarian titik baru selalu berasal dari himpunan titik yang sudah dicapai sebelumnya. Dari percobaan modifikasi terhadap algoritma Prim ini, diperoleh tiga ide algoritma baru.

D. Metode Algoritma "Linear"

Langkah-langkah:

1. Dari titik awal, dilakukan pemindaian melingkar hingga ditemukan titik terdekat.
2. Jadikan titik terdekat tersebut sebagai pangkal baru.
3. Ulangi langkah 1-2 hingga semua titik pada peta tersambung.



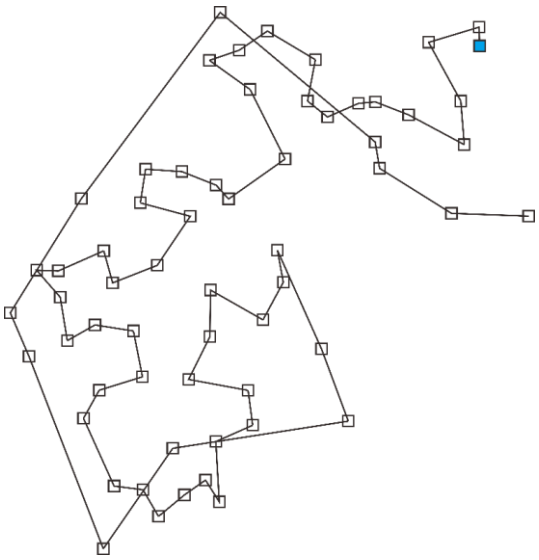
Gambar 8: Simulasi algoritma "linear"
Sumber: dokumen penulis

Dengan algoritma ini, dibentuk sebuah pohon yang condong (*skewed tree*) dan hanya memiliki 1 buah daun. Algoritma ini dapat dengan jelas dinilai tidak efisien dikarenakan pemain tidak dapat memanfaatkan fitur teleportasi dalam video game tersebut. Penilaian ini diambil berdasarkan alasan:

a) Pohon yang dibentuk berbentuk linear.
b) Pohon tidak memiliki percabangan.
c) Kecenderungan adanya titik lokasi yang relatif dekat, tetapi menjadi jauh karena "terdistraksi" titik lain.

Sedangkan dari sisi kelebihan, algoritma ini memiliki kelebihan:

a) Tidak perlu melakukan perbandingan.
b) Tidak perlu melakukan pencatatan informasi apapun.

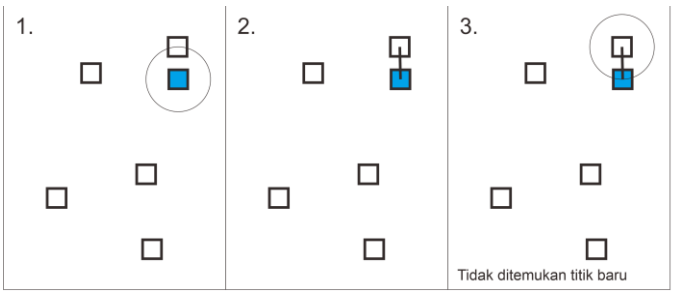


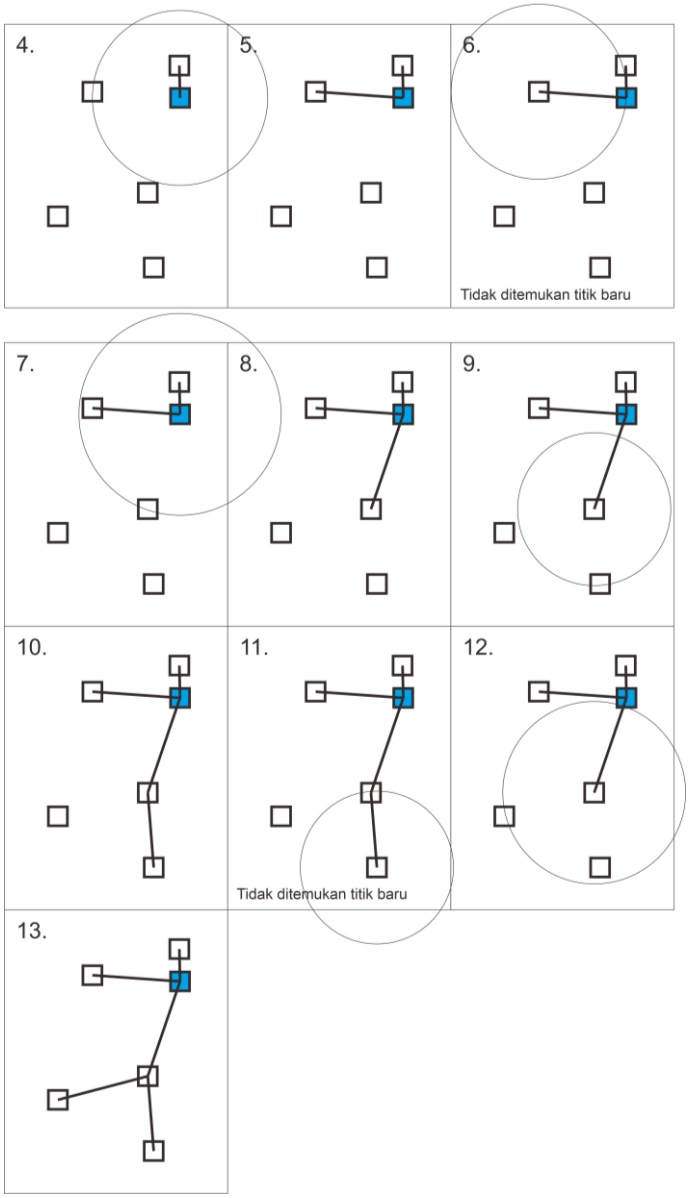
Gambar 9: Diagram pohon hasil algoritma "linear"
Sumber: dokumen penulis

E. Metode Algoritma "Tulang Daun"

Langkah-langkah:

1. Dari titik awal, dilakukan pemindaian melingkar hingga ditemukan 1 titik terdekat.
2. Jadikan titik terdekat (child) tersebut sebagai pangkal baru, kemudian dilakukan pemindaian melingkar dengan radius lebih pendek daripada radius titik baru (child) dengan titik sebelumnya (parent).
3. Jika ditemukan titik baru dan ulangi langkah 2.
4. Jika tidak ditemukan titik baru, kembali ke titik sebelumnya (parent), kemudian dilakukan pemindaian melingkar dengan radius lebih penjang daripada radius titik parent ke child sebelumnya.
5. Ulangi hingga semua titik pada peta tersambung.





Gambar 10: Simulasi algoritma "tulang daun"
Sumber: dokumen penulis

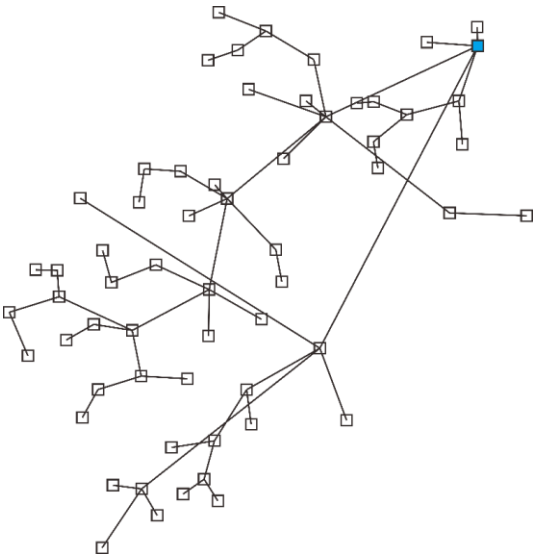
Dengan algoritma ini, berhasil dibentuk sebuah diagram pohon dengan jumlah daun sebanyak 32. Jika divisualisasikan secara tertata, diagram pohon yang dibentuk menyerupai ruas-ruas tulang daun, yaitu semakin jauh jarak (semakin tinggi level) sebuah sisi ke pangkal (akar), semakin pendek panjang dari sisi tersebut. Algoritma ini lebih baik daripada algoritma pertama yang membentuk pohon linear, tetapi dinilai masih kurang efisien dengan alasan:

a) Terdapat titik lokasi yang relatif dekat, tetapi menjadi jauh karena keterbatasan radius pemindaian.
b) Pemain memanfaatkan fitur teleportasi menjadi terlalu sering, yang padahal dimungkinkan akses titik waypoint baru tanpa perlu kembali ke titik sebelumnya (parent).

Sedangkan dari sisi kelebihan, algoritma ini memiliki kelebihan:

a) Tidak perlu melakukan perbandingan.

b) Tidak perlu melakukan pencatatan informasi apapun.

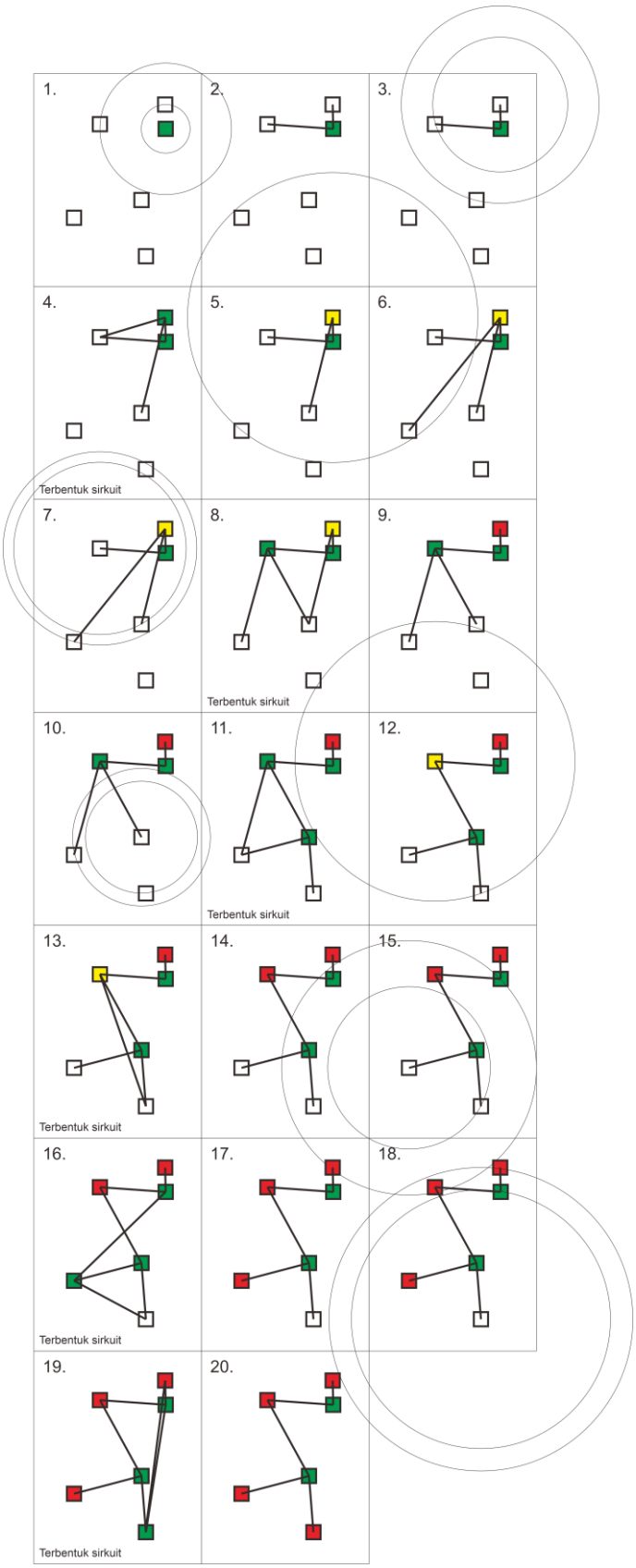


Gambar 11: Diagram pohon hasil algoritma “tulang daun”
Sumber: dokumen penulis

F. Metode Algoritma “Pohon Biner”

Langkah-langkah:

1. Dari titik awal pertama, dilakukan pemindaian melingkar hingga diperoleh dua titik terdekat sebagai node baru.
2. Jadikan titik baru (child) tersebut sebagai pangkal baru, kemudian dilakukan pemindaian melingkar hingga dibentuk dua sisi baru (mungkin ditemukan titik baru) dan beri kode, misalnya hijau, pada titik baru sebagai pangkal pemindaian tersebut.
3. Jika ujung dari sisi yang terbentuk merupakan titik dengan kode hijau dan sudah bertetangga dengan 3 titik lainnya, eliminasi sisi tersebut dan dilakukan pencarian sisi yang baru.
4. Selama terbentuk sirkuit:
    a. Eliminasi satu sisi terpanjang dan catat sisi tersebut supaya tidak ada titik lain yang dapat membentuk sisi serupa.
    b. Jika titik yang merupakan pangkal dari sisi yang dieliminasi masih berkode hijau, ganti kode menjadi misalnya kuning (sebagai tanda sudah mengalami eliminasi sisi satu kali), kemudian jadikan titik (yang berkode kuning) tersebut sebagai pangkal baru dan lakukan pencarian sisi terdekat yang baru.
    c. Jika titik yang merupakan pangkal dari sisi yang dieliminasi sudah berkode kuning, ganti kode menjadi misalnya merah (sebagai tanda sudah mengalami eliminasi dua kali), dan tidak lagi dilakukan pencarian sisi baru dari titik tersebut.
5. Pindah ke titik baru yang belum pernah dijadikan pangkal dan ulangi dari langkah ke 2.



Gambar 12: Ssimulasi algoritma “pohon biner”
Sumber: dokumen penulis

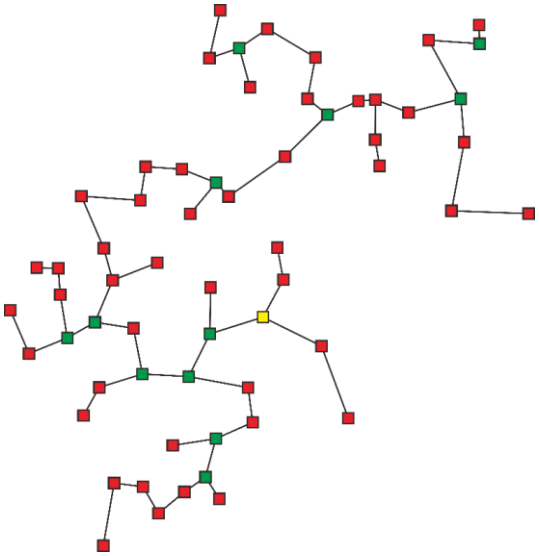
Dengan algoritma ini, berhasil dibentuk sebuah diagram pohon dengan jumlah daun sebanyak 16. Jika dilihat

berdasarkan hasil yang diperoleh (dalam bentuk visual), algoritma ini paling efisien dibandingkan algoritma-algoritma hasil modifikasi sebelumnya. Hal ini didasari dengan pertimbangan:

a) Jumlah daun yang medium (tidak terlalu banyak dan tidak terlalu sedikit.
b) Pemain dapat memanfaatkan fitur teleportasi dalam jumlah yang wajar dan efektif.

Tetapi, jika diperhatikan dari segi algoritmanya, terdapat kekurangan signifikan dibandingkan algoritma-algoritma sebelumnya, seperti:

a) Perlu melakukan pencatatan terhadap kode dari titik dan sisi yang sudah dieliminasi.
b) Perlu dilakukan *tracking* sisi yang dibentuk sehingga dapat memprediksi kemungkinan terbentuknya sirkuit.
c) Lebih kompleks dengan adanya pengulangan (*loop*).



Gambar 13: Diagram pohon hasil algoritma "pohon biner"
Sumber: dokumen penulis

IV. Simpulan

Teori pohon dapat diaplikasikan dalam kehidupan sehari-hari untuk menangani permasalahan, baik sederhana hingga kompleks. Melalui makalah ini, ditunjukkan bahwa teori pohon dengan modifikasi algoritma Prim dapat membantu menemukan rute eksplorasi suatu peta dalam video game open-world yang dilengkapi titik-titik destinasi penting.

Berdasarkan hasil analisis ketiga algoritma hasil modifikasi yang diajukan, algoritma ketiga ("pohon biner") merupakan algoritma yang menghasilkan diagram pohon paling efisien, dihitung dari jumlah daun yang dimiliki graf pohon yang dihasilkan. Pemain dapat memanfaatkan fitur teleportasi sambal memungkinkan untuk melakukan eksplorasi keseluruhan peta. Walaupun hasilnya efisien, dari segi komputasi, algoritma ini lebih kompleks dan cenderung akan memakan banyak memori karena memerlukan pencatatan banyak elemen dalam peta yang diujikan, seperti sisi yang pernah dieliminasi dan kode untuk tiap titik lokasi.

V. Ucapan Terima Kasih

Puji syukur kepada Tuhan Yang Maha Esa atas berkat dan rahmat-Nya sehingga makalah ini dapat diselesaikan dengan baik dan tepat waktu. Penulis menyampaikan terima kasih kepada Dra. Harlili M.Sc, selaku dosen mata kuliah IF 2120 Matematika Diskrit untuk K2 semester I 2020/2021, atas bimbingannya selama satu semester dan Dr. Ir. Rinaldi Munir, M.T., atas penyediaan website yang berisi sumber materi dan daftar makalah sebagai sumber inspirasi. Penulis juga menyampaikan terima kasih kepada sumber-sumber referensi sehingga dapat mendukung isi dan informasi dari makalah ini.

Referensi

[1] Booker, Logan (2008). "Pandemic Working On New 'Open World / Sandbox' IP". Kotaku.
[2] "The complete history of open-world games (part 2)". Computer and Video Games. (2008).
[3] Munir, Rinaldi. 2016. Matematika Diskrit Edisi Revisi keenam. Bandung: Informatika Bandung.
[4] https://www.geeksforgeeks.org/prims-minimum-spanning-tree-mst-greedy-algo-5/ diakses pada 4 Desember 2020.
[5] https://www.tutorialspoint.com/graph_theory/graph_theory_trees.htm diakses pada 4 Desember 2020.
[6] https://arstechnica.com/gaming/2017/03/youre-now-free-to-move-about-vice-city-a-history-of-open-world-gaming/?comments=1 diakses pada 5 Desember 2020.

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, 5 Desember 2020

Aurelius Marcel Candra
13519198