Simulasi Jalur Fast Break Optimal Berbasis A* Algorithm dengan Dukungan Passing pada Grid Lapangan Basket 2D

Raka Daffa Iftikhaar - 13523018[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]raka.daffa2005@gmail.com, 13523018@std.stei.itb.ac.id

Abstrak - *Fast break* merupakan salah satu komponen krusial dalam permainan basket. *Fast break* bisa menjadi peran penting dalam menentukan arah pertandingan terhadap kedua tim. Karena pentingnya *fast break*, harus ada sumber evaluasi terkait berbagai tindakan yang telah dilakukan pada saat berada pada kondisi ini. Makalah ini menjadi solusi terkait hal tersebut dengan simulasi *fast break* yang optimal menggunakan pendekatan algoritma A*. Simulasi ini membuktikan bahwa algoritma A* dapat digunakan untuk menggambarkan gerakan yang efisien saat *fast break* dan dapat dijadikan dasar evaluasi untuk tindakan yang diambil.

Kata Kunci - Algoritma, A*, *Fast Break*, Evaluasi Basket.

I. Pendahuluan

Olahraga basket merupakan salah satu olahraga yang digemari oleh banyak orang. Olahraga ini dimainkan oleh dua tim dengan maksimal lima orang per tim. Berdasarkan standar FIBA (*Federation Internationale de Basketball*), permainan basket dimainkan pada sebuah lapangan persegi panjang dengan panjang 28 meter dan panjang 15 meter. Popularitas olahraga ini bisa dibilang cukup tinggi. [3] Berdasarkan beberapa website yang telah penulis kunjungi, olahraga basket selalu menjadi tujuh besar olahraga paling populer di dunia. Kepopuleran ini dikarenakan skema kompetitif yang dimiliki oleh olahraga basket, dimulai dari peraturan sampai tata cara bermain.

Membahas tentang tata cara bermain, terdapat satu strategi unik yang digunakan pada skema kompetitif ataupun permainan kasual. Strategi itu bernama *fast break*. Apa itu *fast break*? Untuk memahami *fast break* secara utuh, kita perlu mengerti terkait dasar permainan ini. Seperti olahraga kompetitif pada umumnya, pemenang didapatkan dari tim dengan poin tertinggi. Sebuah poin di olahraga basket didapatkan ketika sebuah tim memasukkan bola ke ring milik lawannya. Berdasarkan hal tersebut, semua anggota tim penyerang akan berperan dalam melakukan *attacking* atau penyerangan ke ring milik tim bertahan. Hal unik terjadi ketika bola yang coba dimasukkan oleh tim penyerang ke ring tim bertahan gagal dimasukkan dan bola tersebut didapatkan oleh tim bertahan. Tim bertahan akan melakukan sebuah skema menyerang balik yang dinamakan *fast break*. Kenapa dinamakan *fast break*? Karena tim bertahan akan melakukan transisi bola yang sangat cepat dari ring milik mereka ke ring milik tim penyerang dalam sebuah lapangan dengan panjang 28 meter.

Transisi cepat tersebut pastinya tidak mudah, dimana orang yang mendapat bola pertama harus memiliki kecermatan dalam memilih, diantara memberikan bola ke teman satu tim atau membawanya sendiri. Hal itu dikarenakan tim penyerang pasti akan mencoba untuk menghalangi dan mengejar bola tersebut. Oleh karena itu, pada makalah kali ini penulis akan mensimulasikan *fast break* sederhana optimal dengan pendekatan algoritma A* dengan beberapa batasan yang akan dibahas pada bagian metode penelitian. Walaupun demikian, diharapkan simulasi ini bisa digunakan sebagai dasar analisis dan evaluasi terhadap pengambilan keputusan yang sudah dilakukan oleh seorang pemain dalam olahraga basket.

II. Metode Penelitian

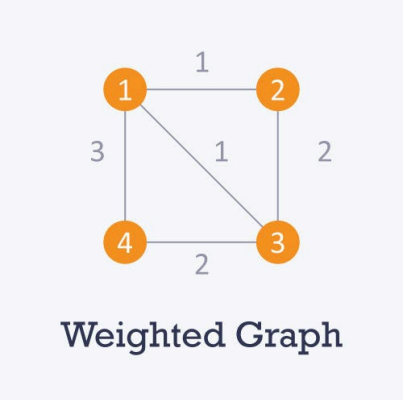
Penelitian ini dibuat dengan menggunakan metode pengembangan sistem simulasi. Metode ini dipilih karena tujuan utama dari penelitian ini adalah merancang sebuah program fungsional dan fleksibel yang dapat mensimulasikan dan mencari jalur *fast break* paling efektif berdasarkan posisi pemain dalam lapangan basket yang akan direpresentasikan dalam sebuah grid dua dimensi dengan pendekatan algoritma A*. Simulasi yang dikembangkan dalam penelitian ini memiliki sejumlah batasan tertentu untuk meningkatkan efisiensi program.

Proses pengembangan program akan mengikuti tahapan yang terstruktur, dimulai dari analisis kebutuhan program, perancangan program, implementasi, dan pengujian. Penulis juga melakukan studi kasus dengan didasarkan pada data posisi pemain yang diekstraksi dari skenario-skenario *fast break* yang didokumentasikan dalam video pertandingan bola basket. Dengan demikian, metode ini memastikan bahwa output dari program tidak hanya berupa analisis teoritis, tapi juga sebuah produk sistem yang fungsional dan teruji dengan data yang merepresentasikan kondisi nyata.

III. Dasar Teori

*A. Algoritma Pencarian Jalur A**

Algoritma A* adalah salah satu algoritma pencarian jalur. Jalur yang dicari adalah jalur terpendek antara suatu titik awal menuju ke titik akhir. Algoritma ini sering digunakan dalam menemukan jalur terpendek dalam sebuah peta. [1] Algoritma A* berjalan menggunakan basis graf berbobot. Graf berbobot adalah graf yang menggunakan angka untuk mewakili biaya yang dimiliki oleh suatu node jika node lain ingin mengunjunginya. Dikarenakan biaya tersebut, algoritma A* akan mencari jalur dengan biaya terendah dari titik awal ke tujuan akhir dengan detail khusus.



Gambar 1. Contoh Graf Berbobot
Sumber: https://sethuram52001-medium-com.translate.goog/data-structures-weighted-graphs/

[1] Pada dasarnya, A* juga dibuat berdasarkan algoritma lain yaitu BFS atau *best-first search*. Algoritma A* terkenal karena ciri khas hasil perhitungan yang optimal dan lengkap. Optimal dihitung dari segi biaya dan lengkap dari segi hasil yang ditemukan. Hal tersebut dikarenakan terdapat hal unik untuk menghitung biaya dari sebuah node ke node lainnya. Bila kita biasanya menghitung biaya dari sebuah node A ke node C melalui B dengan perhitungan biaya A - B dan B - C saja, pada algoritma A* diperlukan juga sebuah perhitungan heuristik.

Apa heuristik itu? Heuristik adalah sebuah metode atau strategi yang digunakan untuk memperkirakan solusi terbaik secara efisien, terutama dalam penyelesaian masalah yang kompleks. Bila diibaratkan, heuristik seperti tebak-tebakan bagaimana suatu hal bisa mencapai tujuan secara efisien tanpa memikirkan aturan-aturan yang berlaku. Heuristik dalam algoritma A* dibentuk dalam sebuah fungsi. Fungsi ini akan menghitung perkiraan biaya dari node awal ke node tujuan dengan metode yang beragam, tergantung keinginan pembuat program.

Kembali ke perhitungan algoritma A*, algoritma ini dihitung dengan menjumlahkan biaya asli dari node awal ke node n (jalan yang sudah ditempuh) ditambah dengan estimasi biaya dari node n ke node tujuan (heuristik). Bila dituliskan secara formal, rumus perhitungan biaya dari A* adalah sebagai berikut.

$$f(n) = g(n) + h(n)$$

$f(n)$ = Estimasi biaya total dari node awal ke node akhir.
$g(n)$ = Biaya asli dari node awal ke node n.
$h(n)$ = Estimasi biaya dari n ke node tujuan (heuristik).

Terdapat hal penting dalam perhitungan algoritma ini, dimana fungsi heuristik yang digunakan harus lebih kecil atau sama dengan biaya sebenarnya ke node tujuan. Pendekatan dengan dua komponen yang dihitung ini memang membuat algoritma A* menjadi algoritma yang optimal, namun terdapat kekurangan pada algoritma ini yaitu kompleksitas waktu dan kompleksitas ruangnya yang cukup besar karena algoritma ini perlu banyak ruang untuk menyimpan semua kemungkinan jalur dan sangat banyak waktu yang digunakan untuk menemukan hasilnya.

B. Penjelasan Detail Fast Break

Sejatinya, dasar tentang *fast break* telah dijelaskan pada bagian pendahuluan, namun untuk memahami *fast break* lebih dalam, diperlukan pembahasan terkhusus. *Fast break* memiliki empat fase utama, yaitu fase inisialisasi, fase transisi, fase serangan, dan fase penyelesaian. Fase inisialisasi adalah bagaimana sebuah peristiwa *fast break* dimulai, yaitu pada saat tim penyerang gagal memasukkan bola ke dalam ring tim bertahan dan tim bertahan mendapatkan bola tersebut. Fase kedua dimulai saat tim yang awalnya bertahan menjadi balik menyerang dengan ritme penyerangan yang sangat cepat. Hal ini untuk memanfaatkan posisi pemain lawan yang sedang berada di posisi depan dan tidak ada yang menjaga ring mereka.

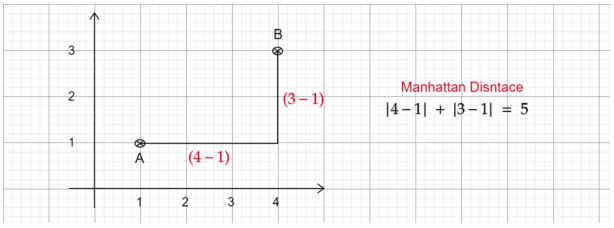
Selanjutnya akan masuk ke fase ketiga yaitu fase serangan. Pada fase ini, tim yang mendapat bola untuk menyerang akan berusaha secepat mungkin menuju ring lawan dan tim yang kehilangan bola akan berusaha semaksimal mungkin untuk merebut bola itu kembali. Ketika pemain yang membawa bola sudah dekat dengan ring lawan, fase terakhir akan dimulai, yaitu fase penyelesaian. Pada fase ini, biasanya pemain akan langsung memasukkan bola ke ring dengan *lay-up*, tapi tidak jarang pemain melakukan atraksi unik seperti mengopernya ke temannya ataupun melakukan gerakan-gerakan unik.

Gambar 2. Contoh *Fast Break*
Sumber: https://en.wikipedia.org/wiki/Fast_break

Simulasi terkait *fast break* ini akan dilakukan dalam grid dua dimensi. Secara singkat, nantinya akan ada grid lapangan dan penanda bagi tim yang menyerang dan bertahan. Simulasi *passing* bola juga akan dilakukan. Terkait simulasi ini akan dijelaskan lebih detail pada bagian analisis.

C. Manhattan Distance

Manhattan distance atau jarak manhattan adalah cara untuk menghitung jarak antara dua titik di dalam sebuah grid. Perbedaan *manhattan distance* dengan *euclidean distance* adalah *manhattan distance* menghitung jarak dari dua titik bila titik tersebut hanya bisa bergerak dalam empat gerakan saja, yaitu atas, bawah, kanan, dan kiri sesuai dengan spesifikasi grid yang akan digunakan dalam simulasi kali ini. Nama unik perhitungan ini diambil dari taxi di jalan Manhattan yang hanya bisa bergerak horizontal atau vertikal saja.



Gambar 3. Perhitungan *Manhattan Distance*
Sumber:https://datascienceparichay.com/article/manhattan-distance-python/

IV. Analisis

Proses analisis pada upaya simulasi kali ini dibagi menjadi beberapa bagian. Bagian tersebut adalah pembuatan grid lapangan beserta pemainnya, inisialisasi *state*, transisi antar *state*, logika *passing* dan A*, urutan aksi, dan visualisasi.

A. Representasi Grid

Pembuatan grid merupakan salah satu komponen utama pada simulasi ini. Grid merupakan representasi lapangan basket dalam simulasi kali ini. Tidak hanya merepresentasikan lapangan basket, grid juga merepresentasikan pemain dari dua tim, ring basket, serta bola basket. Secara lengkap, berikut contoh representasi grid yang telah penulis buat.

```
[".", ".", ".", "D", ".", ".", ".", ".", ".", ".", "."]
["A", "A", "D", ".", ".", ".", ".", ".", ".", ".", "."]
["R", "A", "D", ".", ".", ".", ".", ".", ".", ".", "R"]
["B", ".", "D", ".", ".", ".", ".", ".", ".", ".", "."]
[".", "D", "A", ".", ".", ".", ".", ".", ".", ".", "."]
```

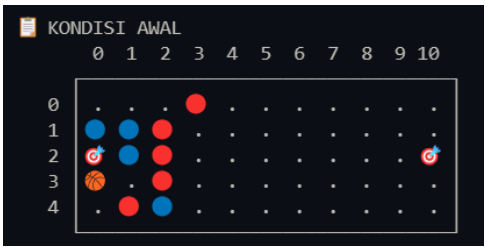
Berdasarkan grid tersebut, terdapat beberapa simbol huruf. R berperan sebagai ring dimana R di sebelah kiri berperan sebagai ring tim yang menyerang saat *fast break* dan ring di sebelah kanan berperan sebagai ring tujuan dari tim penyerang. Huruf A yang berarti *attacking* menjadi representasi tim yang menyerang sedangkan huruf D yang berarti *defending* menjadi representasi tim yang bertahan. Huruf B pada grid tersebut merepresentasikan bola basket pada saat bola baru jatuh dari ring dan ditangkap oleh salah satu pemain *attacking*.

Grid dibuat dalam sebuah ruang dua dimensi dengan ukuran 5x11. Hal tersebut untuk meniru secara nyata ukuran lapangan basket yang sebenarnya. Bagian kosong pada lapangan untuk pemain bergerak direpresentasikan dengan sebuah titik.

B. Inisialisasi State

Inisialisasi *state* adalah bagian dimana tahapan awal pada simulasi ini. Semua kebutuhan yang diperlukan pada simulasi akan diinisialisasi pada awal program. Grid yang telah dibahas pada bagian sebelumnya juga termasuk inisialisasi *state* awal. Terdapat mekanisme *deep copy* grid awal agar grid yang diproses tidak merusak data asli dari grid yang telah dibuat. Selain itu jumlah pergerakan awal dan jumlah maksimal *state* yang boleh dilakukan juga diinisialisasi pada bagian ini.

Tidak hanya terfokus di grid lapangan, inisialisasi awal ini juga digunakan untuk mengetahui posisi awal setiap pemain, lalu posisi bola, dan posisi ring basket. Semua kebutuhan terkait algoritma A* juga diinisialisasi seperti cost atau biaya yang telah dihitung. Semua itu dikumpulkan menjadi satu dalam sebuah visualisasi *state* yang menarik seperti berikut.



Gambar 4. Kondisi Awal

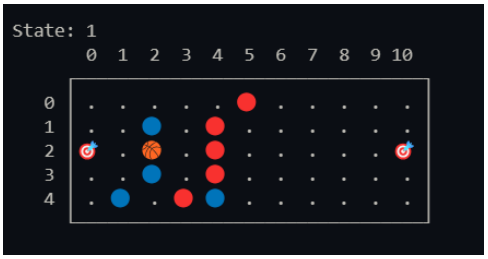
Berdasarkan gambar tersebut, bisa dilihat bahwa representasi grid awal berupa huruf diubah menjadi lebih menarik pada saat inisialisasi. Huruf R diubah menjadi papan target, huruf B diubah menjadi bola basket, huruf A diubah menjadi lingkaran biru, dan huruf D diubah menjadi lingkaran merah.

C. Transisi antar State

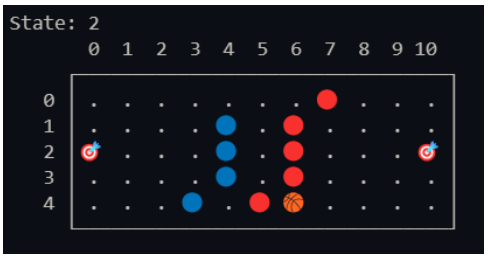
Transisi yang dilakukan pada simulasi ini dilakukan menggunakan sebuah fungsi untuk mengubah grid lama menjadi grid baru setiap tick simulasi. Setiap tick, proses yang dilakukan adalah membuat sebuah grid kosong kemudian menyalin elemen statik pada grid seperti ring basket. Informasi pada setiap tick akan disimpan agar memudahkan pemrosesan untuk tick berikutnya.

Mekanisme pergerakan pemain dilakukan dengan membuat setiap pemain penyerang bergerak dua langkah ke arah ring tujuan di sebelah kanan. Hal ini merepresentasikan tim penyerang dalam *fast break* yang berlari menuju ring lawan dengan cepat. Proses ini memastikan juga pada simulasi bahwa tidak ada dua pemain yang menempati posisi yang sama, semua pemain

tersebar di lapangan. Selain itu, pergerakan juga divalidasi agar hanya bisa bergerak ke sel yang kosong. Berikut contoh transisi yang dilakukan.



Gambar 5. State pertama



Gambar 6. State kedua

Pada dua gambar tersebut, terlihat perbedaan yang cukup signifikan pada *state* awal dan akhir, dimana terlihat baik itu pemain menyerang, bertahan, dan bola sama-sama bergerak ke arah ring. Setiap *state* ini akan muncul satu per satu setiap dua detik untuk menciptakan simulasi nyata dari permainan basket itu sendiri. Terkait teknis *passing* dan pergerakan bola akan dibahas pada bagian selanjutnya.

*D. Logika Passing dan A**

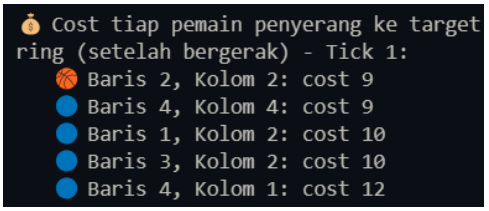
Passing adalah proses mengoper bola ke teman satu tim yang dirasa posisinya lebih kosong atau lebih leluasa untuk mengendalikan bola dan melindungi bola dari rebutan lawan. Logika *passing* atau operan pada simulasi ini menggunakan pendekatan algoritma A*. Hal tersebut dilakukan dengan menghitung *cost* atau biaya dari tiap pemain menyerang ke ring tujuan pada setiap *state*. Seperti yang sudah dijelaskan diatas, algoritma A* menggunakan rumus sebagai berikut.

$$f(n) = g(n) + h(n)$$

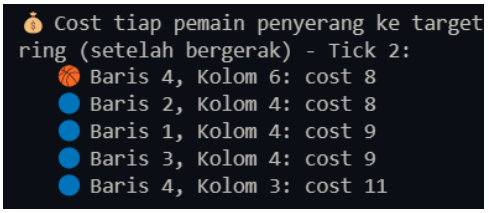
$f(n)$ = Estimasi biaya total dari node awal ke node akhir.
$g(n)$ = Biaya asli dari node awal ke node n.
$h(n)$ = Estimasi biaya dari n ke node tujuan (heuristik).

Pada program ini, biaya asli yang digunakan adalah jarak yang sudah ditempuh pemain dari posisi awalnya sampai ke suatu titik tertentu atau titik n. Heuristik yang digunakan pada program ini adalah *manhattan distance* dari setiap pemain menyerang ke ring tujuan ditambah dengan jumlah pemain yang menghalanginya. Hal ini dikarenakan untuk mensimulasikan kondisi aktual dimana pemain yang dihalangi oleh banyak pemain lawan akan lebih sulit untuk membawa bola ke ring daripada pemain yang tidak dihalangi oleh lawan.

Berdasarkan hal tersebut, logika *passing* yang berjalan adalah setelah semua pemain baik itu penyerang atau bertahan bergerak ke arah kanan sebesar dua satuan setiap *state*-nya, program akan menghitung semua biaya atau *cost* pemain menyerang. Kemudian, program akan menentukan sebuah pilihan yaitu diantara seorang pemain tetap membawa bola atau pemain tersebut mengoper bola ke rekannya. Pemain akan tetap membawa bola tersebut bila melihat bahwa tidak ada *cost* atau biaya yang lebih baik dari dia, sementara jika ada pemain dengan biaya yang lebih baik atau sama dengan dia maka bola akan di oper ke pemain tersebut. Hal tersebut dapat dijelaskan dengan gambar berikut.



Gambar 7. Biaya tiap pemain pada state pertama



Gambar 8. Biaya tiap pemain pada state kedua

Gambar di atas adalah penjelasan biaya atau *cost* dari state pertama dan kedua yang sudah diperlihatkan di bagian sebelumnya. Pada gambar tersebut, terlihat bahwa di state pertama pemain yang membawa bola adalah pemain pada baris dua (pemain 2). Hal tersebut berbeda dengan state kedua dimana pemain yang membawa bola adalah pemain pada baris empat (pemain 4). Hal tersebut dikarenakan pemain 2 melihat bahwa ada rekan setimnya yang memiliki *cost* sama dengan dia dan pemain 2 memutuskan untuk mengoper bola tersebut. Hal ini memang tidak mempercepat bola untuk sampai ke ring karena biaya antar dua pemain ini sama untuk mencapai ring, namun dalam representasi dunia nyata, hal ini dilakukan untuk mengecoh pemain lawan supaya tidak terfokus ke satu pemain saja. Kecohan ini dilakukan dengan mempertimbangkan resiko juga sehingga bola di oper menuju orang yang memiliki biaya menuju ring minimal sama dengan yang sedang membawa bola sekarang.

Terlihat juga biaya pada kedua state untuk setiap pemain berbeda, dimana pada state kedua biaya untuk setiap pemain menurun sebanyak dua. Hal tersebut dikarenakan pada state tersebut para pemain sudah bergerak dua satuan ke kanan yang membuat jarak dari setiap pemain ke ring tujuan lebih dekat.

Untuk membuktikan perhitungan A* yang dilakukan pada program ini, mari ambil satu contoh yaitu pemain pada baris dua di state pertama. Pemain tersebut berada pada baris dua dan kolom dua pada state pertama. Jika kita menghitung heuristiknya, maka terlihat bahwa jarak antara ring dengan pemain adalah tujuh satuan, lalu terdapat satu pemain musuh yang menghalangi jadi total dari fungsi heuristiknya adalah delapan. Pemain ini telah bergerak satu kali dari kondisi awal yang membuat jarak dari node awal ke node sekarang adalah satu. Oleh karena itu, *cost* atau biaya dari pemain ini pada state pertama adalah sembilan. Perhitungan yang sama bisa dilakukan juga pada state kedua, perbedaannya hanya pada jarak asli dari awal (g(n)) yang menjadi dua dan heuristik (h(n)) yang menjadi enam karena sudah bergerak dua satuan mendekati ring.

E. Urutan Aksi

Urutan aksi untuk simulasi ini sebenarnya cukup sederhana. Program dimulai dari inisialisasi seperti yang sudah dijelaskan di atas. Tahap setelah inisialisasi adalah melakukan iterasi state atau tick. Pada setiap state atau tick, program akan selalu menggerakan pemain baik itu pemain bertahan ataupun pemain menyerang sebanyak dua langkah. Setelah selesai menggerakan, program akan menentukan posisi terbaru dari setiap pemain. Karena adanya pergerakan pemain dan posisi baru, program juga menghitung ulang *cost* atau biaya dari setiap pemain untuk setiap state. Logika *passing* seperti yang sudah dijelaskan pada bagian sebelumnya juga akan dilakukan jika syarat-syaratnya dipenuhi.

Hal tersebut akan terus dilakukan sampai bola basket menyentuh ring. Kasus ini ditandai dengan gambar target di sebelah kanan yang sudah berubah menjadi piala. Hal ini dikarenakan dalam dunia nyata, *fast break* selesai saat pemain yang membawa bola sudah dekat dengan ring dan memasukkan bola tersebut ke dalam ring. Berikut visualisasi dari program ketika simulasi sudah selesai dilakukan.

Gambar 9. Hasil Akhir Simulasi

Setiap tick atau state pada simulasi mewakili satu satuan waktu permainan. Untuk menciptakan kesan simulasi nyata, antar tick akan diberikan jeda waktu sehingga pergerakan pemain dan pergerakan bola dapat diamati secara seksama oleh pengguna. Dengan urutan aksi yang terstruktur ini, simulasi dapat menggambarkan proses *fast break* secara logis dan mendekati kondisi permainan basket di dunia nyata.

F. Kompleksitas Waktu dan Ruang

Waktu yang dibutuhkan program dapat dijelaskan dengan melihat komponen-komponen utama di setiap langkah simulasi atau tick. Algoritma A*, yang digunakan untuk menentukan jalur terbaik bola basket menuju ring memiliki kompleksitas waktu sekitar O(V x log V + E) dalam kasus terburuk, dimana V adalah jumlah posisi di grid (node) dan E adalah jumlah hubungan antar posisi (edge). Karena grid berukuran 5x11, maka maksimal ada 55 posisi dan sekitar 220 hubungan antar posisi. Dengan demikian, pemanggilan A* per pemain memiliki waktu sekitar O(220).

Untuk gerakan pemain, program hanya mempertimbangkan dua langkah menuju ring. Oleh karena itu, proses ini hanya membutuhkan waktu tetap, yaitu O(1) per pemain. Pekerjaan tersulit pada saat mencari operan terbaik. Hal tersebut dikarenakan program harus membuat salinan grid sementara dan menjalankan A* untuk setiap pemain penyerang. Waktu yang dibutuhkan sekitar O(55 + 5x220) atau O(1155) per tick. Secara keseluruhan, waktu komputasi per tick adalah sekitar O(1.266). Karena ukuran grid dan jumlah pemain tetap, waktu ini dianggap konstan atau O(1) per tick.

Terkait kompleksitas ruang, penggunaan memori utama pada program ini berasal dari grid lapangan dan struktur data yang digunakan oleh algoritma A*. Grid utama yang berukuran 5x11 membutuhkan 55 unit ruang memori (O(55)). Selain itu, saat perhitungan A* dilakukan, program membuat salinan grid sementara yang juga memakan ruang sebesar O(55). Salinan yang dibuat dalam satu tick adalah lima, oleh karena itu, total penggunaan memorinya adalah O(275).

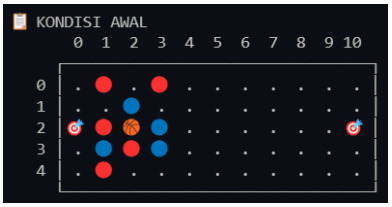
Program juga menyimpan posisi-posisi yang sudah bergerak, sedang ditempati, atau sudah diproses dalam tick tertentu menggunakan dictionary dan set, dengan total penggunaan memori sekitar O(20). Data status permainan membutuhkan biaya tambahan O(5). Jika dijumlahkan, total penggunaan memori untuk seluruh simulasi adalah O(55 + 55 + 275 + 20 + 5) = O(410). Ukuran grid yang tetap menjadikan kompleksitas ruangnya konstan atau O(1).

G. Perbandingan Program dalam Permainan Basket

[2] Perbandingan ini penulis lakukan dengan sebuah klip *fast break* antara tim Davidson melawan tim North Carolina. Berikut kondisi awal dari *fast break* antara kedua tim ini.

Gambar 10. Kondisi Awal *Fast Break*

Berdasarkan gambar tersebut, representasi grid yang saya buat adalah seperti ini.



Gambar 11. Representasi Grid Kondisi Awal

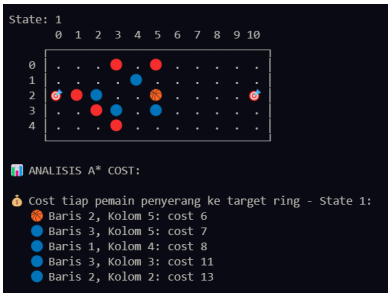
Kemudian klip video ini penulis bagi menjadi tiga bagian karena total waktu klip adalah tiga detik.

Gambar 12. Bagian 1

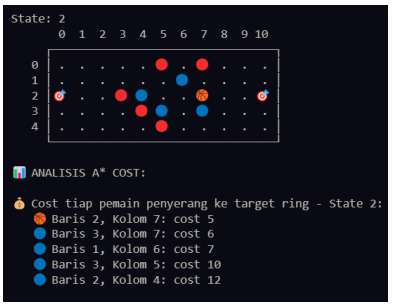
Gambar 13. Bagian 2

Gambar 14. Bagian 3

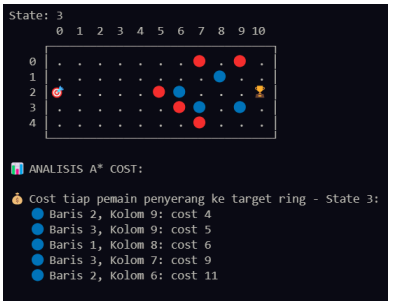
Kemudian, ini hasil simulasi yang diberikan oleh program. Berdasarkan gambar, bisa dilihat bahwa adanya kemiripan yang cukup signifikan dari state asli di olahraga basket dengan hasil dari program.



Gambar 15. State/Tick Pertama



Gambar 16. State/Tick Kedua



Gambar 17. State/Tick Ketiga

V. Kesimpulan

Berdasarkan simulasi dan analisis yang telah dilakukan, dapat disimpulkan bahwa algoritma A* bisa digunakan untuk melakukan simulasi jalur *fast break* optimal dengan dukungan passing pada grid lapangan basket dua dimensi. Dengan mendefinisikan state berupa posisi pemain, bola, dan ring dalam bentuk grid, program ini mampu memodelkan pergerakan bola dan pemain secara efisien.

Penggunaan algoritma A* memungkinkan simulasi menentukan jalur terpendek menuju ring lawan dengan mempertimbangkan posisi pemain bertahan dan jarak ke ring lawan sebagai hambatan. Logika *passing* juga dirancang berdasarkan evaluasi biaya minimum antar pemain yang menyerang. Analisis kompleksitas juga menunjukkan bahwa program berjalan dengan kompleksitas waktu dan ruang yang konstan. Hal ini bisa menjadi bahan evaluasi jangka panjang bagi seorang pemain basket ataupun sebuah tim basket.

Penulis berharap sebuah program lanjutan yang lebih mendetail baik itu terkait analisis ataupun fitur-fitur yang lebih matang bisa dikembangkan suatu hari. Dengan demikian, A* bisa menjadi sebuah metode yang berguna bagi pelatih, analisis olahraga, dan juga sebagai salah satu metode evaluasi.

VI. Lampiran

Link Repository Kode:
https://github.com/rakdaf08/makalahStima

Link Video Demonstrasi:
https://youtu.be/P3RpHQtzZzM

Link Video *Fast Break*:
https://bit.ly/dncFastBreak

VII. Ucapan Terima Kasih

Segala puji dan syukur penulis panjatkan kepada Tuhan Yang Maha Esa karena dengan nikmat dan anugerahnya penulis bisa menyelesaikan makalah ini. Penulis juga menyampaikan terima kasih kepada Ibu Dr. Nur Ulfa Maulidevi, S.T, M.Sc. sebagai dosen pengampu IF2211 Strategi Algoritma kelas K01 atas bimbingan dan pemberian ilmu sebagai dasar penulisan makalah ini. Tidak lupa juga penulis mengucapkan terimakasih kepada Bapak Dr. Ir. Rinaldi Munir T. dan Bapak Monterico Adrian, S.T., M.T. sebagai dosen pengajar mata kuliah ini. Penulis berterima kasih kepada ibu penulis karena tanpa dukungan dari beliau, penulis belum tentu bisa menyelesaikan makalah ini. Terakhir, penulis juga mengucapkan terima kasih kepada teman-teman penulis serta pihak-pihak lain yang telah mendorong penulis dalam menyelesaikan makalah ini.

Referensi

[1] "Algoritma A* (A Star): Pengertian, Cara Kerja, dan Kegunaannya." *Trivusi*, 20 Januari 2023, https://www.trivusi.web.id/2023/01/algoritma-a-star.html. Diakses pada 23 Juni 2025.

[2] Phipps, Jez. "How To Properly Run a Fastbreak in Basketball | JP Productions." *YouTube*, 16 Desember 2019, https://www.youtube.com/watch?v=kPtQ1IU5zB0. Diakses pada 24 Juni 2025.

[3] Putri, Fayrisya Maliha Riyawati Soehadi. "10 Most Popular Sports in the World." *Tempo English*, 28 Oktober 2024, https://en.tempo.co/read/1933986/10-most-popular-sports-in-the-world. Diakses pada 22 Juni 2025.

PERNYATAAN

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

Bandung, 24 Juni 2025

Raka Daffa Iftikhaar - 13523018