Penentuan Jalur Scouting pada Permainan The Program Menggunakan Uniform Cost Search, Greedy Best First Search, dan A*

Haikal Assyauqi - 13522052
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13522052@std.stei.itb.ac.id

***Abstract*— Dalam dunia game, efisiensi sangat penting untuk meningkatkan pengalaman bermain dan kemampuan pengguna. Pada makalah ini, akan meneliti sebuah game bernama "The Program", sebuah game *American football* yang memiliki sistem scouting rute yang menyerupai struktur graf. Rute scouting ini akan dianalisis menggunakan 3 algoritma yaitu Uniform Cost Search, Greedy Best First Search, dan A-Star. Setiap algoritma diterapkan untuk menemukan jalur paling efisien dalam berbagai skenario permainan, dengan mempertimbangkan faktor seperti jumlah dan kualitas pemain yang bisa dipantau.**

Keywords—The Program, Uniform Cost Search, Greedy Best First Search, A*

I. Pendahuluan

"The Program" merupakan sebuah permainan manajerial di bidang *American football* tingkat kuliah, permainan ini bertujuan untuk membuat sebuah tim menuju konferensi terbaik dalam liga NCAA yakni *Big 10* dan menjuarai *National championship.*

Secara umum, seperti kuliah pada umumnya, setiap pemain akan melewati 4 tingkat perkuliahan di antaranya, *Freshmen* (Tingkat 1), *Sophomore* (Tingkat 2), *Junior* (Tingkat 3), dan *Senior* (Tingkat 4), ketika seorang pemain telah mencapai tingkat 4 perkuliahan, maka seorang pemain tidak diperbolehkan lagi untuk membela kampus, hal ini menyebabkan sebuah tim akan kehilangan pemain setiap tahunnya, oleh karena itu dalam liga NCAA terdapat sistem *scouting* untuk menyeleksi pemain SMA ke dalam tim kampus, dalam permainan "The Program" sistem *scouting* dibuat sebagai graf yang harus dikunjungi tim *scouting* jika ingin memantau dan mengontrak seorang pemain

Dalam graf *scouting* pada permainan "The Program", terdapat beberapa aspek yang ditampilkan pada setiap kota di antaranya pemain normal, pemain top, harta karun, dan titik kosong, karena banyaknya aspek, pemain cenderung salah memilih strategi *scouting* yang berakibat pada buruknya *draft class* pada kampus tersebut.

Dengan menggunakan algoritma UCS, GBFS, dan A*, *scouting* yang dilakukan akan menjadi lebih efektif dan pemain yang didapatkan akan menjadi lebih berkualitas sehingga dapat menuju konferensi *Big 10.*

II. Landasan Teori

A. Path Planning

Path planning merupakan sebuah cara atau proses dalam menentukan jalur optimal dari titik awal ke titik tujuan dalam suatu graf,. Algoritma menggunakan aspek biasanya menggunakan biaya perjalanan sebagai aspek perhitungan. Bidang yang menggunakan *path planning* sebagai algoritma pemecahan masalah seperti robotika, permainan, navigasi, dan transportasi

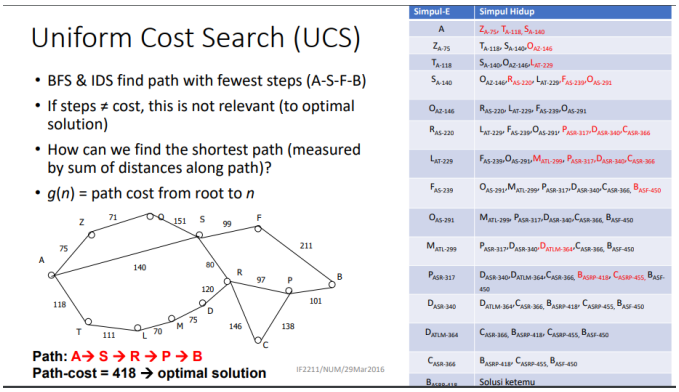
Beberapa pendekatan *path planning* antara lain *Uniform Cost Search, Greedy Best First Search,* dan *A-star,* setiap pendekatan memiliki kelebihan masing-masing seperti Solusi yang optimal maupun waktu yang singkat

B. Uniform Cost Search

Uniform Cost Search (UCS) adalah algoritma untuk menentukan jalur dengan biaya terendah dari titik awal ke tujuan dalam graf. UCS merupakan pengembangan dari BFS, dengan memperhatikan biaya, dalam UCS, biaya sering dilambangkan dengan g(n) yang merupakan jarak dari titik awal ke titik n, yang di mana jarak antar titik didefinisikan oleh *programmer* ataupun keadaan yang terlihat.

Prinsip dari UCS yaitu UCS memperioritaskan biaya paling sedikit terlebih dahulu, dan pada *code* UCS didefinisikan menggunakan *priority queue,* di mana yang akan diolah terlebih dahulu yaitu titik yang memiliki nilai paling rendah

Dengan menggunakan UCS terdapat kelebihan dan kekurangan, kelebihan menggunakan UCS yaitu *output* yang dihasilkan pasti optimal, karena program ini menguji seluruh biaya yang termurah dan belum tereksplorasi, namun kekurangannya antara lain kompleksitas waktu yang memakan waktu yang banyak karena hamper seluruh titik dikembangkan oleh algoritma ini dan hal itu juga berpengaruh kepada penggunaan memori yang sangat besar



Gambar 1. Ilustrasi UCS
Sumber: [1]

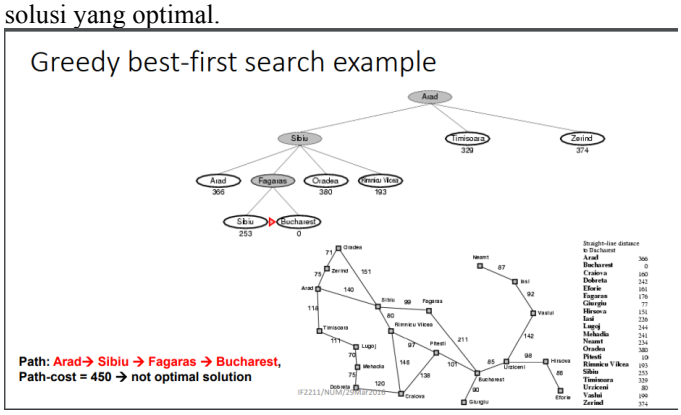
C. *Greedy Best First Search (GBFS)*

Greedy Best First Search merupakan algoritma yang menggunakan pendekatan heuristic dalam menentukan solusi, algoritma ini diyakini merupakan algoritma *path planning* dengan waktu paling cepat.

Nilai heuristic atau biasa ditulis h(n) pada GBFS merupakan nilai estimasi dari titik n ke titik tujuan, biasanya nilai heuristic diambil dari jarak titik n ke titik tujuan atau menggunakan Manhattan Distance, dalam menentukan biaya heuristic, kita harus memastikan nilainya admissible, admissible yaitu estimasi biaya simpul ke tujuan lebih kecil dari biaya actual terendah dari titik n ke tujuan, beberapa sifat yang dimiliki oleh nilai heuristic yang admissible yaitu, nilai heuristic tidak pernah melebihi biaya nyata, mengarah ke optimalitas, dan kosnisten.

Kelebihan yang dimiliki oleh GBFS yaitu GBFS memiliki kecepatan yang sangat cepat, dan GBFS sangat sederhana sehingga program ini baik unutk manajemen memori.

Kekurangan GBFS yaitu hasil ini tidak menjamin Solusi yang optimal, yang artinya hasil akhir berniali lebih besar dari solusi yang optimal.



Gambar 2. Ilustrasi GBFS
Sumber: [1]

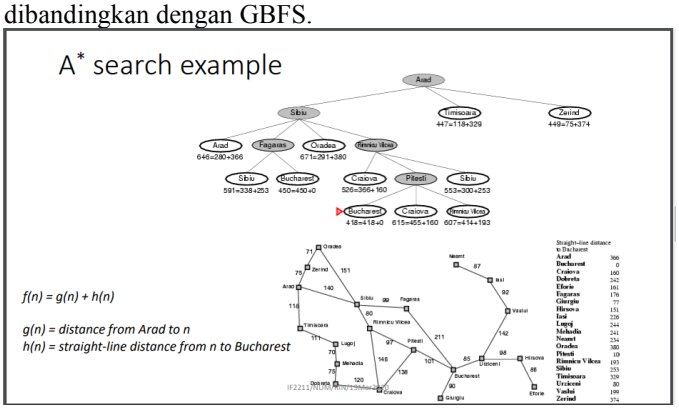
D. *A* Search*

A* *search* merupakan gabungan kelebihan dari algoritma *Uniform Cost Search* dan *Greedy Besr Fist Search* dengan menggabungkan biaya sebenarnya dari titik awal ke titik n dan biaya heuristic dari titik n ke titik akhir, karena menggabungkan 2 algoritma, *A* search* bisa dibilang dapat memenuhi hal yang kurang dari *Uniform Cost Search* dan *Greedy Best First Search* sehingga program ini cepat dan memiliki solusi yang optimal.

Prinsip dari A* yaitu melakukan evaluasi menggunakan g(n) dan h(n). Sehingga nilai prioritas menjadi f(n) = g(n) + h(n), yang mana nilai f(n) ini yang menentukan *node* yang akan diekspansi, dengan nilai f(n) terendah lah yang menjadi prioritas utama untuk diekspand.

Kelebihan dari algoritma ini adalah program ini dapat menemukan solusi yang optimal, efisiensi yang sangat baik, dan fleksibel.

Sementara kekurangannya yaitu, waktu yang lebih lama dibandingkan dengan GBFS.



Gambar 3. Ilustrasi A*
Sumber: [2]

E. *NCAA Football League*

NCAA Football League merupakan liga *American football* mahasiswa yang diadakan setiap tahunnya, mayoritas dari pemain professional NFL (National Football League) akan diambil dari *NCAA Football League* ini.

Secara garis besar, *NCAA Football League* dibagi menjadi 2 subdivisi yakni FBS (Football Bowl Subdivision) dan FCS (Football Championship Subdivision), namun secara pamor dan kualitas FBS jauh lebih baik dari FCS, oleh karena itu banyak permainan yang menggunakan FBS sebagai *storyline* dan latar kondisi.

Dalam *NCAA Football League* terdapat 10 konferensi yang memiliki kualitas berbeda-beda yaitu *Big Ten, Pacific 12, Big 12, SEC, ACC, Sun Belt, American Athletic Conference, Mountain West, Conference-USA,* dan *Mid-American Conference*, setiap tim dalam konferensi ini bersaing untuk memperoleh *College Football National Championship*, mayoritas tim yang meraih kesempatan untuk bersaing di *College Football National Championship* merupakan tim yang berasal dari 5 konferensi besar yakni *Big Ten, Pacific 12, Big 12, SEC,* dan *ACC*, oleh karena itu ada tim yang berpindah konferensi untuk mencoba peruntungan di konferensi yang lebih kuat dan bersaing di *College Football National Championship*. Sebagai contoh, salah satu tim *underdog* saat ini yakni Texas Christian University, usai mendominasi di *Mountain West,* pada 2012 TCU memilih untuk berpindah ke salah satu konferensi tebaik yakni *Big 12*

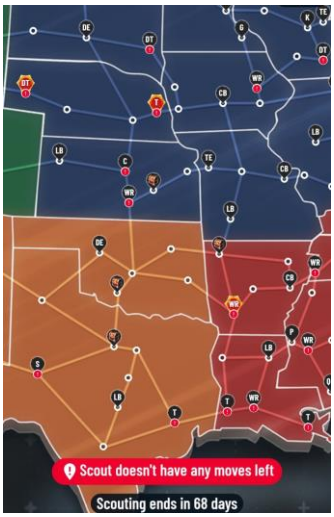
Selain *College Football National Championship,* terdapat pertandingan *bowl* yang mempertandingkan juara antar konferensi, untuk konferensi besar terdapat *bowl* seperti *Rose Bowl* dan *Sugar Bowl,* untuk konferensi lemah seperi *AAC, C-USA,* dan *Mountain West* membuat *bowl* seperti *Birmingham Bowl* dan *Armed Forces Bowl*

F. *The Program*

The Program merupakan sebuah permainan *American football* di liga *NCAA Football League*, pemain ditugaskan untuk memulai permainan di konferensi rendah hingga menuju konferensi tertinggi di *NCAA Footbal League* seperti *Big 10* maupun *Big 12,* banyak fitur yang bisa dikembangkan oleh pemain seperti *training* dimana pemain dapat melatih skuadnya untuk menjadi yang terbaik, seperti aturan dalam *NCAA Football League* seorang pemain dalam skuad hanya daoat bermain maksimal 4 tahun, oleh karena itu terdapat fitur lain yang penting dalam permainan ini yakni *Scouting.*

Gambar 4. Tampilan Menu Permainan *The Program*
Sumber: Dokunentasi Pribadi

Dalam permainan *The Program* ini,kita memiliki waktu 70 hari unutk memantau dan menagajak pemain bermain di kampus yang kita latih, dalam proses perekrutannya kita akan melewati berbagai kota yang ada di Amerika Serikat yang berupa *node,* di mana kita bisa berpindah dari satu kota ke kota lain 1 kali dalam 1 hari, tiap kotanya kita akan melihat berbagai *node* yang berbeda seperti *node* merah yakni *node* yang memiliki *top 5 class* yang mungkin kita dapatkan, *node* hitam yaitu pemain normal yang kualiatasnya belum pasti bisa jadi bagus, bisa jadi tidak, *node* harta karun yakni *node* yang berisi resource untuk melakukan perekrutan dan yang terakhir yaitu *node* kosong yang tidak memiliki apa-apa di dalamnya, pada *node* merah dan *node* hitam dapat dilihat juga bahwa terdapat tanda seru yang menandakan posisi yang wajib direkrut karena posisi tersebut akan ditinggalkan oleh pemain *Senior* yang akan segara tamat dan mengikuti *NFL draft.*



Gambar 5. *Scouting* dalam gim *The Program*
Sumber: Dokumentasi Pribadi

III. Implementasi dan Pengujian

Pada menu *scouting* di permainan *The Program* nilai g(n) antar satu *node* dengan *node* lain tidak diberikan, untuk mendapatkan hasil optimal yakni pemain yang didapat berkualitas dan dapat memenuhi slot kosong maka jarak dari satu *node* ke *node* lain akan menggunakan nilai prioritas, dari hasil pengamatan selama bermain maka penulis merumuskan nilai prioritas sebagai berikut:

1. Nilai 1 untuk pemain *top 5* dengan tanda seru.
2. Nilai 2 untuk pemain normal dengan tanda seru.
3. Nilai 3 untuk pemain *top 5.*
4. Nilai 4 untuk harta karun.
5. Nilai 5 untuk pemain normal.
6. Nilai 6 untuk titik kosong.

Node dengan tanda seru dijadikan sebagai *node* dengan prioritas tertinggi karena kita harus memenuhi persyaratan jumlah pemain minimum unutk musim depan, untuk pemain *top 5* dijadikan prioritas ketiga karna sebuah tim perlu *superstar* dalam tim, sementara pemain *normal* di posisi kelima karena hasil dari pemain normal ini sangat *random,* oleh karena itu, jika tidak ada kebutuhan mendesak seperti pemenuhan kuota pemain, pemain normal tidak terlalu diperlukan.

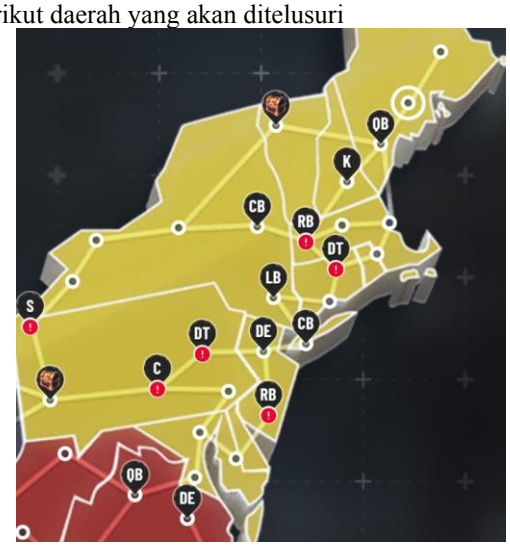
Sementara unutk nilai h(n) menggunakan jarak dari titik n ke titik akhir, namun hal yang unik dari *scouting* ini adalah titik akhir kita bebas di mana saja asal kota yang sudah dilewati mencapai 70 kota (sesuai dengan jumlah hari yang disediakan dalam *scouting),* hal ini juga berlaku dengan titik awal, namun pada makalah ini titik awal akan ditentukan karena jika tidak ditentukan akan menggunakan algoritma *brute force* yang mana tidak sesuai dengan bahasan makalah ini. Maka sebagai contoh untuk nilai h(n), jika hari dibatasi menjadi 7 hari dan jumlah kota yang dikunjungi sama dengan 2 maka nilai h(n) akan sama dengan 5.

Selanjutnya, kita akan uji coba admissible, jika kita amati nilai dari h(n) memiliki nilai maksimal = 1, dan pada g(n)

nilai minimal yang mungkin kita dapatkan yaitu 1 ketika *node* memiliki pemain *top 5* dengan tanda seru, sehingga nilai h(n) pasti selalu lebih kecil atau sama dengan g(n) sehingga nilai h(n) bisa dianggap admissible.

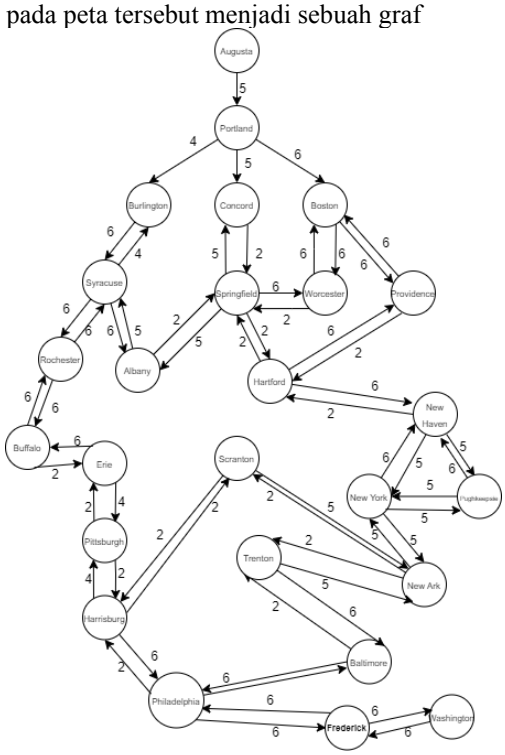
Setelah menentukan nilai g(n) dan h(n), saatnya kita mulai menelusuri jalur *scouting* yang tersedia untuk mendapatkan pemain yang berkualitas, untuk mempermudah pencarian kita akan memulai penjelajahan dari kota Augusta, hari pencarian dibatasi menjadi 7 hari, dan area penjelajahan dibatasi di area *East Coast* tekhusus *Atlantic*.

Berikut daerah yang akan ditelusuri



Gambar 6. Peta *Scouting* yang Ditelusuri
Sumber: Dokumentasi Pribadi

Agar memudahkan dalam menerima informasi akan kita ubah info pada peta tersebut menjadi sebuah graf



Gmabar 7. Graf *Scouting* yang Memiliki Nilai
Sumber: Dokumentasi Pribadi

Dari graf yang telah dibuat, kita dapat menentukan nilai g(n) nantinya, dan dengan info di atas kita dapat membuat list kota yang tersedia

```
city_list = [
    ['Augusta', 6, ['Portland']],
    ['Portland', 5, ['Burlington', 'Concord', 'Boston',
'Augusta']],
    ['Burlington', 4, ['Syracuse', 'Portland']],
    ['Syracuse', 6, ['Burlington', 'Albany',
'Rochester']],
    ['Rochester', 6, ['Buffalo', 'Syracuse']],
    ['Buffalo', 6, ['Rochester', 'Erie']],
    ['Erie', 2, ['Buffalo', 'Pittsburgh']],
    ['Pittsburgh', 4, ['Erie', 'Harisburg']],
    ['Harrisburg', 2, ['Pittsburgh', 'Philadelphia']],
    ['Philadelphia', 6, ['Frederick',
'Harisburg','Baltimore']],
    ['Concord', 5, ['Springfield', 'Portland']],
    ['Springfield', 2, ['Concord', 'Albany', 'Worcester',
'Hartford']],
    ['Worcester', 6, ['Springfield', 'Boston']],
    ['Boston', 6, ['Worcester', 'Providence',
'Portland']],
    ['Providence', 6, ['Boston', 'Hartford']],
    ['Hartford', 2, ['Providence', 'Springfield', 'New
Haven']],
    ['New Haven', 6, ['Hartford', 'New York',
'Poughkeepsie']],
    ['Poughkeepsie', 5, ['New Haven', 'New York']],
    ['New York', 5, ['Poughkeepsie', 'Newark', 'New
Haven']],
    ['Newark', 5, ['New York', 'Trenton', 'Scranton']],
    ['Scranton', 2, ['Newark', 'Harrisburg']],
    ['Trenton', 2, ['Newark', 'Baltimore']],
    ['Baltimore', 6, ['Frederick', 'Trenton']],
    ['Frederick', 6, ['Washington','Philadelphia']],
    ['Washington', 6, ['Frederick']],
    ['Albany', 5, ['Syracuse', 'Springfield']],
    ['Philadelphia', 6, ['Frederick', 'Harisburg']],
]
```

Pada program di atas kita dapat melihat nama kota, cost menuju kota, dan kota yang bisa dituju dari kota n, saatnya mulai menghiung mengunakan algoritma UCS, GBFS, dan A*

1. UCS
   Dengan algoritma UCS kita akan menghitung nilai penelusuran berdasarkan g(n)

```
if algo == 'UCS':
    while (len(list_path[0]) <= max_path):
```

```
        path_now = list_path[0].copy()
        city_now = list_path[0][len(list_path[0])-1]
        idx_city = get_city(city_now, city_list)
        already_expand.append(city_now)
        for i in range(len(city_list[idx_city][2])) :
            path = path_now.copy()
            if city_list[idx_city][2][i] not in
already_expand :
                val_saat_ini = gn[0]
                next_city = city_list[idx_city][2][i]
                path.append(next_city)
                idx_next_city = get_city(next_city,
city_list)
                val_next = city_list[idx_next_city][1] +
val_saat_ini # f(n) indeks ekspand + city indeks ekspand
                idx_list_path =
get_idx_list_path(val_next, gn)
                list_path.insert(idx_list_path,path)
                gn.insert(idx_list_path, val_saat_ini +
city_list[idx_next_city][1])
        list_path.pop(0)
        gn.pop(0)
    print(list_path)
    print(gn)
    print(list_path[0])
```

Dengan menggunakan UCS, maka penjelajahan yang disarankan adalah

```
['Augusta', 'Portland', 'Concord', 'Springfie
ld', 'Hartford', 'New Haven', 'New York', 'Po
ughkeepsie']
```

Gambar 8. Hasil Pencarian dari Algoritma UCS
Sumber: Dokumentasi Pribadi

Terlihat dari hasil pencarian menggunakan UCS disarankan untuk melalui kota Augusta → Portland → Concord → Springfield → Hartford → New Haven → New York → Poughkeepise, dari hasil tersebut jika melihat ke peta *scouting* kita akan mendapat 4 pemain normal dan 2 pemain normal *urgent* (memiliki tanda seru), hal ini sangat optimal karena hampir dalam tiap *step* terdapat pemain baru

2. GBFS
   GBFS menggunakan nilai h(n) sebagai prioritasnya

```
while (len(list_path[0]) <= max_path):
        path_now = list_path[0].copy()
        city_now = list_path[0][len(list_path[0])-1]
        idx_city = get_city(city_now, city_list)
        already_expand.append(city_now)
        for i in range(len(city_list[idx_city][2])) :
            path = path_now.copy()
            if city_list[idx_city][2][i] not in
already_expand :
                val_saat_ini = hn[0] # f(n) indeks
ekspand
                next_city = city_list[idx_city][2][i] #
city indeks ekspand
                path.append(next_city) # path indeks
ekspand + city indeks ekspand
                idx_next_city = get_city(next_city,
city_list)
                val_next = max_path - len(path) + 1 #
f(n) indeks ekspand + city indeks ekspand
                idx_list_path =
get_idx_list_path(val_next, hn) # mencari indeks untuk
memasukkan f(n) indeks ekspand + city indeks ekspand
                list_path.insert(idx_list_path,path)
                hn.insert(idx_list_path, val_next)
        idx_path = list_path.index(path_now)
        list_path.pop(idx_path)
        hn.pop(idx_path)
        print("Ini list path: ", list_path)
        print("Ini value: ", hn)
    print(list_path)
    print(list_path[0])
```

Dengan menggunakan GBFS, maka penjelajahan yang disarankan adalah

```
['Augusta', 'Portland', 'Burlington', 'Syracuse',
'Albany', 'Springfield', 'Worcester', 'Boston']
```

Gambar 9. Hasil Pencarian dari Algoritma GBFS
Sumber: Dokumentasi Pribadi

Terlihat dari hasil pencarian menggunakan GBFS, disarankan untuk melalui kota Augusta → Portland → Burlington → Syracuse →Albany → Springfield → Worcester → Boston, jika melihat ke peta maka kita akan mendapat 2 pemain normal, 1 pemain normal *urgent,* dan 1 harta karun, cukup bagus, namun hasil yang diberikan belum optimal.

3. A*
   A* menggabungkan antara g(n) dan h(n) dalam perhitungannya

```
while (len(list_path[0]) <= max_path):
        path_now = list_path[0].copy()
        city_now = list_path[0][len(list_path[0])-1]
        idx_city = get_city(city_now, city_list)
        already_expand.append(city_now)
        for i in range(len(city_list[idx_city][2])) :
            path = path_now.copy()
```

```
            if city_list[idx_city][2][i] not in
already_expand :
                next_city = city_list[idx_city][2][i
                path.append(next_city)
                idx_next_city = get_city(next_city,
city_list)
                gn_next = city_list[idx_next_city][1] +
gn[0]
                hn_next = max_path - len(path) + 1
                val_next = gn_next + hn_next
                idx_list_path =
get_idx_list_path(val_next, astar)
                list_path.insert(idx_list_path,path)
                astar.insert(idx_list_path, val_next)
                gn.insert(idx_list_path, gn_next)
                hn.insert(idx_list_path, hn_next)
        idx_path = list_path.index(path_now)
        list_path.pop(idx_path)
        hn.pop(idx_path)
        gn.pop(idx_path)
        astar.pop(idx_path)
        print("Ini list path: ", list_path)
        print("Ini value: ", astar)
    print(list_path)
    print(list_path[0])
```

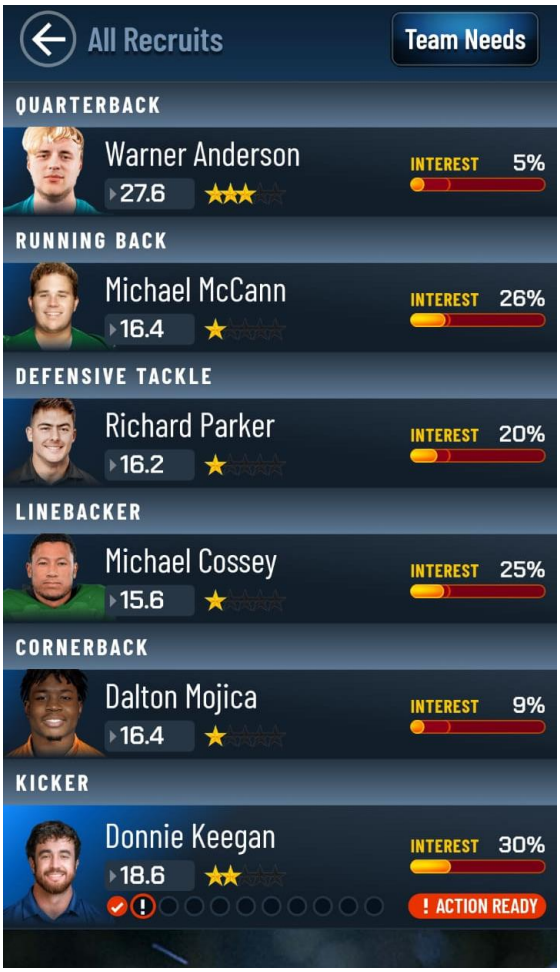
Dari hasil perhitungan, algoritma A* menyarankan unutk mengunjungi kota

```
['Augusta', 'Portland', 'Concord', 'Springfield', '
Hartford', 'New Haven', 'New York', 'Poughkeepsie']
```

Gambar 10. Hasil Pencarian dari Algoritma A*
Sumber: Dokumentasi Pribadi

Terlihat dari hasil pencarian menggunakan UCS disarankan untuk melalui kota Augusta → Portland → Concord → Springfield → Hartford → New Haven → New York → Poughkeepise, dari hasil tersebut jika melihat ke peta *scouting* kita akan mendapat 4 pemain normal dan 2 pemain normal *urgent* (memiliki tanda seru), yang mana hasil ini sama dengan UCS sehingga bisa dibilang optimal.

Usai mendapatkan hasil ini, kita akan mencoba menerapkannya untuk *scouting* pada permainan *The Program*



Gambar 11. Hasil *Scouting*
Sumber: Dokumentasi Pribadi

Terdapat 6 pemain yang bisa kita rekrut dari 7 hari melakukan *scouting,* rendahnya kualitas pemain dikarenakan tim saat ini bermain di konferensi lemah yakni *Mid-American Conference*, dengan menggunakan A* kita dapat lebih mudah dalam mencapai konferensi *Big 10.*

IV. Kesimpulan

Dari hasil penelitian, untuk mendapatkan solusi yang optimal kita bisa menggunakan UCS dan A*, sementara jika kita menggunakan GBFS hasil yang didapat hanya akan mencapai local maksimal, yang tidak akan menjadi efisien jika diterapkan ke dalam permainan *The Program.*

Meskipun UCS cukup efektif dalam menemukan solusi namun hal ini dikarenakan ruang lingkup yang kita ambil sangtalah kecil yakni 7 hari, 10x lebih kecil dibanding dengan total hari dalam 1 musim *The Program* yang mencapai 70 hari, hal ini dikarenakan UCS memakan waktu yang banyak dalam melakukan penelusuran pada graf berukuran besar, karena UCS menelusuri semua path yang memiliki nilai terkecil sehingga kompleksitasnya mencapai $O(n^{1+\lfloor C*/\epsilon \rfloor})$.

Sementara untuk GBFS sangat cepat, namun hasilnya seperti yang dilihat pada implementasi, haislnya jauh dari kata

optimal, dan jika kita memperbesar batas hari menjadi 70 hari, kita dapat mengubah h(n) nya menjadi 70 – jumlah kota yang telah dikunjungi.

Dan algoritma terbaik yang bisa digunakan adalah A* karena hasil yang diberikan optimal sama seperti UCS dan hamper sama cepatnya dengan GBFS, karena algoritma ini menggabungkan nilai g(n) dan h(n) dalam mempermudah perhitungan, dengan kompleksitasnya $O(n^m)$.

Dengan makalah ini, diharapkan pembaca dapat menerapkan algoritma ini dengan baik karena algoritma ini tidak hanya digunakan pada permainan *The Program* ini saja, bisa untuk navigasi seperti melakukan perjalanan pada objek wisata pada suatu daerah, pembaca bisa menggunakan jarak sebagai g(n) nya.

UCAPAN TERIMA KASIH

Penulis ingin mengucapkan terima kasih kepada semua pihak yang telah membantu dalam penyelesaian proyek ini. Pertama, penulis bersyukur kepada Allah SWT atas berkat dan rahmat-Nya. Terima kasih kepada Ibu Dr. Nur Ulfa Maulidevi, S.T, M.Sc., dosen mata kuliah Strategi Algoritma IF2211, yang telah memberikan bimbingan dan ilmu yang berharga. Penulis juga berterima kasih semua orang yang telah membantu menyelesaikan ini. Semua kontribusi dan dukungan sangat berarti bagi penulis. Terima kasih atas kerja sama dan dedikasinya.

REFERENCES

[1] R. Munir, 'IF2211 Strategi Algoritma - Semester II Tahun 2023/2024', https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Route-Planning-Bagian1-2021.pdf
Diakses 11 Juni 2024

[2] R. Munir, 'IF2211 Strategi Algoritma – Semester II Tahun 2023/2024', https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Route-Planning-Bagian2-2021.pdf
Diakses 11 Juni 2024

[3] NCAA, https://www.ncaa.com/history/football/fbs
Diakses 12 Juni 2024

[4] ESPN, 'College Football Conferences', https://www.espn.com/college-football/conferences
Diakses 12 Juni 2024

[5] ESPN, 'NCAA 2024: College football ralignment tracker', https://www.espn.com/college-football/story/_/id/40293423/cfb-conference-realignment-tracker-2024
Diakses 12 Juni 2024

[6] Moriarty, Moragn. 'Ranking Every College Football Conference Aftre the 2023-2024 Season', https://bleacherreport.com/articles/10104303-ranking-every-college-football-conference-after-the-2023-24-season
Diakses 12 Juni 2024

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, 12 Juni 2024

Haikal Assyauqi, 13522052