Aplikasi Graf dalam Pemilihan Jalur Operasi Militer Taktis: Studi Kasus Carrier Strike Group-2 USS Dwight D. Eisenhower (CVN-69) oleh US Navy

Eduardus Alvito Kristiadi - 13522004
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13522004@std.stei.itb.ac.id

***Abstract*—Angkatan Laut Amerika Serikat merupakan angkatan laut terkuat di dunia yang memiliki sejumlah *Carrier Strike Group* demi menjaga perairan dunia. Operasi militer dalam satu misi yang dijalankan oleh CSG selalu memiliki rute tertentu. Akan tetapi, dalam kasus tertentu, rute tersebut bisa jadi tidak dapat digunakan karena bebeberapa alasan. Dalam rangka memperhitungkan efisiensi bahan bakar serta suplai logistik di laut, diperlukan adanya suatu pemodelan matematis menggunakan algoritma A* serta prinsip graf berbobot dan berarah untuk mencari rute tercepat serta rute alternatif apabila rute utama tidak dapat digunakan.**

***Keywords*—Graf, Algoritma A*, *Carrier Strike Group*, *US Navy*.**

I. PENDAHULUAN

Amerika Serikat, sebuah negara adidaya yang memiliki pengaruh yang sangat dominan di seluruh dunia. Dengan angkatan bersenjata yang sangat berpengaruh di ranah internasional serta alutsista yang sangat mutakhir, Amerika Serikat merupakan salah satu negara dengan kekuatan militer terkuat di dunia. Berdasarkan *Global Firepower* 2023 *Military Strength Ranking*, Amerika Serikat berada di peringkat pertama, diikuti oleh militer Rusia dan Tiongkok.

Sebagai negara dengan militer terkuat di dunia, Amerika Serikat juga memiliki angkatan laut terkuat di dunia. Berdasarkan *World Directory of Modern Military Ships*, tercatat bahwa pada tahun 2023, Amerika Serikat memiliki sebanyak 243 unit kapal. Jumlah tersebut kurang dari jumlah kapal Republik Rakyat Tiongkok. Namun, secara tonase kotor, Angkatan Laut Amerika Serikat (US Navy) lebih unggul dibanding Angkatan Laut Tiongkok (People's Liberation Army Navy) karena pemeringkatan tersebut tidak hanya diukur dari kuantitas, namun juga kualitas dari tiap kapal perang.

Dalam rangka menjaga perdamaian dunia serta keamanan dan stabilitas regional, angkatan bersenjata Amerika Serikat diharapkan dapat memberikan kehadiran militer di setiap kawasan dari tiap benua terutama pada daerah-daerah yang rawan konflik serta menjadi ancaman bagi dunia internasional. Militer Amerika Serikat hadir tidak hanya melalui pangkalan-pangkalan militer yang tersebar di berbagai negara, tetapi juga melalui angkatan laut yang ditugaskan ke berbagai wilayah laut demi memberikan daya pukul yang efektif bagi segala ancaman yang ada.

Gambar 1. *USS John C. Stennis (CVN-74) steams through the Arabian Sea on Dec. 14, 2018.*
Sumber : https://news.usni.org/2018/12/31/usni-news-fleet-marine-tracker-dec-31-2018

Sebagai perwakilan dari US Navy, Amerika Serikat menugaskan sebanyak 11 gugus tempur kapal induk atau yang biasa dikenal dengan CSG (*Carrier Strike Group*) ke berbagai belahan dunia. Tiap-tiap CSG biasanya diberangkatkan dari suatu pangkalan angkatan laut Amerika Serikat dalam suatu rute tertentu melalui berbagai laut dan negara. Jalur operasi militer tersebut dapat dimodelkan secara matematis dalam bentuk graf. Meskipun tiap jalur operasi yang dilewati dari waktu ke waktu terkadang berbeda, jalur tersebut dapat disederhanakan dan dibuat asumsi-asumsi tertentu ketika membuat model matematika.

Dalam penelitian ini, persoalan dalam jalur operasi militer CSG dimodelkan dalam rangka mencari rute tercepat melalui *shortest path problem* dengan menggunakan algoritma A*. Jalur operasi militer yang digunakan pada penelitian ini adalah jalur operasi militer dari CSG-2 (*Carrier Strike Group-2*) dengan (CVN-69) USS Dwight D. Eissenhower sebagai kekuatan kapal utama yang diberangkatkan dari *Norfolk Naval Station* menuju ke perairan sekitar Timur Tengah, tepatnya di sekitar Teluk Persia/Teluk Arab yang dilaksanakan mulai dari tanggal 14 Oktober 2023. Dilakukan pula pencarian jalur alternative lain yang mungkin menjadi jalur operasi militer CSG-2 apabila jalur utama tidak memungkinkan untuk dilalui.

II. LANDASAN TEORI

A. Carrier Strike Group

Gambar 2. *USS Nimitz Carrier Strike Group in Indian Ocean in 2020.*
Sumber: https://www.taiwannews.com.tw/en/news/4892989

Sebuah *Carrier Strike Group* (CSG) adalah formasi angkatan laut yang tangguh, mewakili salah satu pilar proyeksi kekuatan maritim modern Amerika Serikat. Terdiri dari kapal induk sebagai pusatnya, CSG adalah kekuatan yang serbaguna dan kuat, mampu memberikan kekuatan darat, laut, dan udara yang masif dan berkelanjutan, menjalankan operasi keamanan maritim, dan memproyeksikan pengaruhnya di samudera yang luas di berbagai penjuru dunia. Kapal induk itu sendiri berfungsi sebagai pangkalan udara terapung, menjadi tuan rumah bagi berbagai jenis pesawat, mulai dari jet tempur F-35B Lightning, F/A-18 Super Hornet hingga pesawat pengintaian EA-18G Growler, helicopter Seahawk SH-60, serta pesawat terbang tiltrotor Bell Boeing V-22 Osprey. Satu armada CSG bisa membawa 5000 hingga 7500 kru kapal dalam sekali operasi militer.

Mengelilingi kapal induk antara lain kapal pengawal, seperti kapal penjelajah (*cruiser)*, penghancur *(destroyer)*, kapal suplai logistic (*replenishment ship)*, serta kapal selam (*submarine)* yang membentuk perisai pelindung dan serangan. Kapal-kapal ini dilengkapi dengan senjata canggih, seperti peluru kendali, sistem pertahanan rudal, dan kemampuan anti-kapal selam. Selain itu, sebuah kapal selam dapat menyertai kelompok ini, menambahkan kemampuan penyamaran dan perlindungan bawah air. Sinergi di antara tiap elemen dalam CSG ini meningkatkan kemampuan CSG untuk merespons berbagai ancaman dan serangan, menjadikannya pilar utama dalam strategi angkatan laut kontemporer dan simbol kekuatan militer US Navy di lautan yang luas.

B. Graf

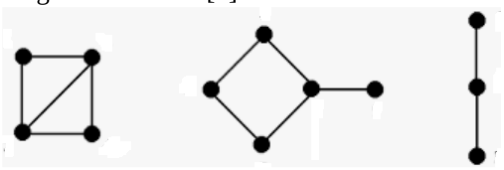
Graf digunakan untuk merepresentasikan objek-objek diskrit dan hubungan antara objek-objek tersebut. Pada awalnya, graf muncul pada persoalan jembatan Konigsberg, sebuah permasalahan yang dialami oleh Leonhard Euler di mana jembatan-jembatan tersebut tidak bagus. [1]

Graf dapat dituliskan dalam bentuk $G = (V, E)$ yang dalam hal ini V = himpunan tidak-kosong dari simpul-simpul (vertices), V = { v1 , v2 , ... , vn } serta E = himpunan sisi (edges) yang menghubungkan sepasang simpul, E = {e1 , e2 , ... , en }. [1]

Berdasarkan ada tidaknya gelang atau sisi ganda pada suatu graf. Maka graf digolongkan menjadi dua jenis:

1. Graf Sederhana (simple graph).

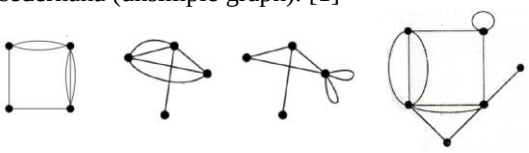
Graf yang tidak mengandung gelang maupun sisi ganda dinamakan graf sederhana. [1]



Gambar 3. Graf sederhana
Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/

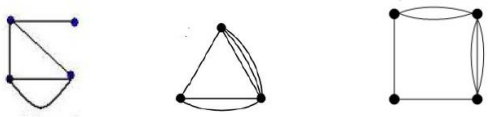
2. Graf tak-sederhana (*unsimple-graph*)

Graf yang mengandung sisi ganda atau gelang dinamakan graf tak-sederhana (unsimple graph). [1]

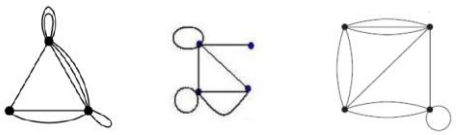


Gambar 4. Graf tak-sederhana
Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/

Graf tak-sederhana dapat dibagi lagi menjadi dua yaitu, graf ganda dan graf semu. Graf ganda (*multi-graph*) adalah graf yang mengandung sisi ganda. Graf semu (*pseudo-graph*) adalah graf yang tiap sisi atau minimal salah satu dari sisinya merupakan sisi gelang. Contoh graf ganda dan graf semu adalah sebagai berikut. [1]



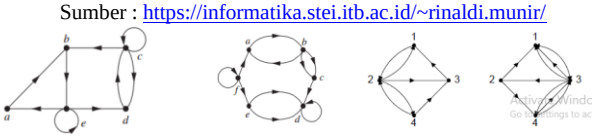
Gambar 5. Graf ganda
Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/



Gambar 6. Graf semu
Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/

Berdasarkan orientasi arah pada sisi, graf dibedakan menjadi 2 jenis, yaitu graf tak-berarah (undirected graph) dan graf berarah (directed graph). *Undirected graph* merupakan graf yang tiap-tiap sisinya tidak mempunyai orientasi arah, sedangkan *directed graph* merupakan graf yang tiap-tiap sisi dari graf tersebut diberikan orientasi arah. Arah tersebut bisa menjauhi maupun mengarah ke node tertentu. Contoh dari graf berarah dan graf tak-berarah adalah sebagai berikut. [1]

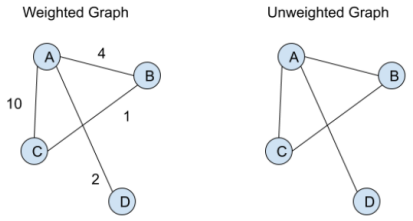
Gambar 7. Graf tak-berarah
Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/



Gambar 8. Graf berarah
Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/

C. Graf Berbobot

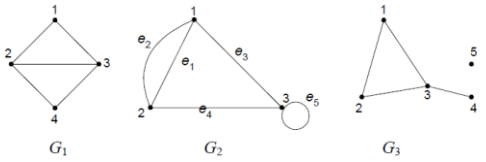
Graf berbobot (*weighted graph*) adalah graf yang setiap cabangnya diberi bobot numerik. Sebuah graf berbobot merupakan jenis khusus dari graf berlabel di mana tiap-tiap labelnya berupa angka yang biasanya diambil sebagai bilangan positif. [1]



Gambar 9. Graf berbobot dan graf tak-berbobot
Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/

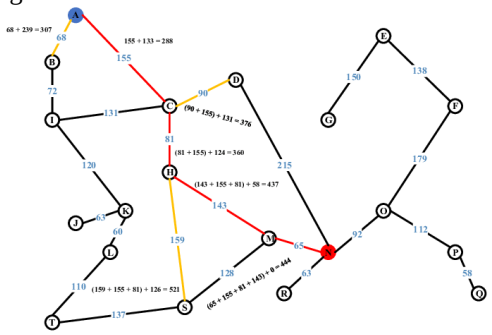
D. Adjacency

Ketetanggan atau *adjacency* merupakan salah satu terminologi dalam teori graf. Dalam konteks graf, *adjacency* menggambarkan keterhubungan atau relasi antara simpul-simpul. Simpul-simpul yang terkoneksi melalui sisi dalam graf disebut sebagai simpul ketetanggan. Konsep *adjacency* umumnya diterapkan untuk mengukur jarak antar simpul, mencari lintasan atau jalur dalam suatu graf, serta menyelesaikan berbagai masalah yang terkait dengan struktur graf. Sebagai contoh, dalam pencarian rute terpendek, algoritma yang digunakan akan memperhatikan keterhubungan *adjacency* antar simpul. Dengan demikian, algoritma ini dapat secara efisien menemukan jalur optimal dalam suatu graf. [1]



Gambar 10. Contoh adjacency dalam graf
Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/

E. Algoritma A*



Gambar 11. Contoh pengujian *A* Algorithm*
Sumber : https://www.researchgate.net/figure/Testing-A-Star-algorithm-on-graph_fig5_348997309

A* *algorithm* adalah algoritma komputer yang secara umum digunakan dalam pencarian jalur (*pathfinding*) dan traversal graf (*graph traversal*). Algoritma ini memanfaatkan *weighted graph*. Algoritma ini digunakan untuk merencanakan jalur efisien antara titik-titik, yang disebut node. Terkenal karena kinerjanya dan ketepatannya, A* telah diperluas untuk berbagai bidang. A* mencapai kinerja yang lebih baik dengan menggunakan heuristik. Algoritma ini memanfaatkan metode Pencarian Terbaik Pertama (Best First Search atau BFS) dan menemukan jalur dengan biaya terkecil (least-cost path) dari node awal (initial node) ke node tujuan (goal node). Untuk menentukan urutan pencarian melalui node-node di dalam pohon, algoritma ini menggunakan fungsi heuristik jarak ditambah biaya (biasanya disimbolkan dengan f(x)). Algoritma A* mengubah konsep heuristik menjadi algoritma pencarian graf dan merupakan pengembangan dari algoritma Dijkstra dengan beberapa ciri khas dari Breadth-First Search untuk menemukan sebuah lintasan dengan bobot minimum dari simoul awal ke simpul akhir. [6]

Notasi dalam algoritma A* dituliskan sebagai berikut:

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

dengan f(n) merupakan total bobot estimasi terendah yang melalui simpul n, g(n) merupakan bobot saat ini untuk mencapai ke simpul n, serta h(n) merupakan estimasi bobot dari simpul n menuju ke simpul tujuan. Algoritma ini memanfaatkan *closedlist* dan *openlist*. *Open list* adalah tempat menyimpan data simpul yang memungkinan untuk diakses dari starting point atau simpul awal maupun simpul yang sedang dijalankan saat ini, sedangkan closed list adalah tempat untuk penyimpanan data simpul/node sebelum A yang juga merupakan bagian dari pemilihan jalur terpendek yang telah berhasil dicapai. Metode perhitungan dengan menggunakan algoritma A* antara lain sebagai berikut: [6]

1. Menambahkan simpul/node awal ke dalam openlist.
2. Mencari simpul/node dengan nilai bobot yang terendah menuju ke simpul-simpul lain yang bertetangga (*adjacent).*
3. Simpul yang bertetangga (*adjacent)* dengan node saat ini diiterasi hingga simpul dapat diraih. Apabila simpul tidak dapat diraih dan tidak ada pada openlist, periksalah apakah lintasan tersebut nilai bobotnya kurang dari lintasan terkini, apabila iya, tukarlah lintasan tersebut menjadi lintasan terkini.
4. Algoritma program akan berhenti ketika simpul/node yang ingin dituju sudah dicapai ataupun sudah tidak ada lagi kemungkinan untuk mencapai simpul tersebut melalui berbagai simpul yang lain.

F. Shortest Path Problem

Shortest path problem merupakan sebuah persoalan yang melibatkan pencarian jalur terpendek antara dua titik (simpul) dalam sebuah graf. Algoritma seperti algoritma Floyd-Warshall dan variasi dari algoritma Dijkstra digunakan untuk mencari solusi terhadap *shortest path problem*. Aplikasi dari masalah jalur terpendek mencakup penggunaan dalam jaringan jalan, logistik, komunikasi, desain elektronik, analisis kontingensi jaringan listrik, dan deteksi komunitas. Dalam penelitian ini, *shortest path problem* digunakan untuk mencari jalur operasi militer dari CSG yang paling efisien jarak serta waktu.

III. PENGAPLIKASIAN ALGORITMA A*

A. Pengumpulan Data Rute Operasi Militer CSG-2 USS Dwight D. Eisenhower

Dalam penelitian ini, dilakukan pengamatan yang terfokus pada Carrier Strike Group-2 (CSG-2) dengan USS Dwight D Eisenhower sebagai pusatnya. Dalam rangka membuat pemodelan rute operasi militer CSG-2, diperlukan adanya data rute terlebih dahulu. Target dari operasi militer CSG adalah daerah perairan di sekitar Timur Tengah untuk melaksanakan misi menjaga keamanan dan pencegahan konflik yang lebih lanjut dan lebih meluas di beberapa negara di Timur Tengah.

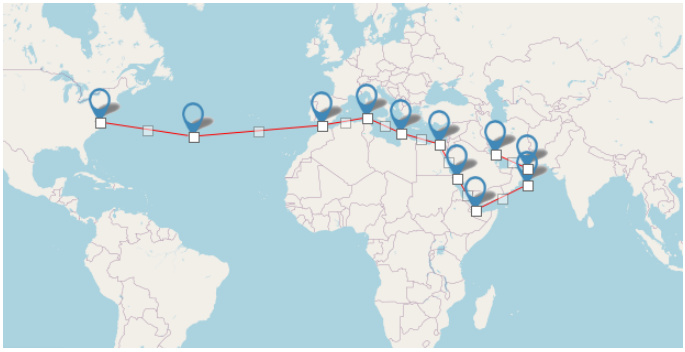
Berdasarkan data dari [2], *Fleet and Marine Tracker*, telah didapat *update* posisi dari tiap-tiap armada CSG yang dimiliki oleh US Navy. Lokasi berupa titik tersebut diperbarui setiap dua hari sekali atau paling sedikit satu minggu sekali. Berdasarkan informasi tersebut, rute perjalanan CSG-2 dapat digambarkan.

Gambar 12. Contoh update lokasi CSG dari US Navy pada tanggal 23 Oktober 2023
Sumber: https://news.usni.org/2023/10/23/usni-news-fleet-and-marine-tracker-oct-23-2023

Berdasarkan data yang telah dikumpulkan didapat tabel lokasi CSG-2 sebagai berikut:

Tanggal	Posisi	(Lattitude, Longtitude)
14 Oktober 2023	Norfolk Naval Station	(36.96942316374587,-76.29618644714355)
17 Oktober 2023	Atlantic Ocean	(33.137551192346145,-46.40625000000001)
28 Oktober 2023	Gibraltar Strait	(35.9602229692967,-5.449218750000001)
30 November 2023	Mediterranean Sea (1)	(38.06539235133249, 9.140625000000002)
3 November 2023	Mediterranean Sea (2)	(34.08906131584996, 19.951171875000004)
4 November 2023	Suez Canal	(31.119970059854737, 32.30529785156251)
5 November 2023	Red Sea	(21.23946244109169, 37.92480468750001)
10 November 2023	Gulf of Aden	(11.523087506868514, 43.94531250000001)
16 November 2023	Gulf of Oman	(19.210022196386095, 60.38085937500001)
26 November 2023	Strait of Hormuz	(24.206889622398023, 60.29296875000001)
28 November 2023	Persian Gulf	(28.3648185914011, 50.185546875)

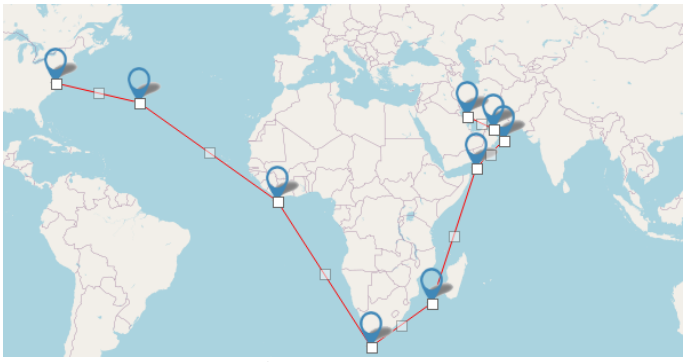
Berdasarkan tabel tersebut, rute operasi militer dapat digambarkan. Proses penggambaran rute operasi militer digambarkan melalui [3]. Dari sana perkiraan jarak dari satu posisi ke posisi berikutnya beserta perkiraan jarak seluruhnya juga berhasil didapatkan. Rute utama yang dilalui adalah sepanjang 16172.25 km. Rute operasi militer CSG-2 dapat digambarkan sebagai berikut:



Gambar 13. Rute Operasi Militer CSG-2 secara keseluruhan
Sumber: dokumentasi pribadi

Berdasarkan gambar tersebut, dilakukan penyederhanaan dengan cara penggambaran rute menggunakan 11 simpul dan 11 sisi untuk mempermudah pemodelan dan perhitungan. Jalur tersebut merupakan jalur utama yang dilalui CSG-2.

Akan tetapi, diperlukan jalur alternatif lain yang dapat dilewati dalam keadaan darurat apabila jalur utama tidak memungkinkan untuk dilalui. Untuk itu, dalam penelitian ini, dilakukan pembuatan jalur alternatif berupa asumsi. Terdapat dua jalur alternatif yang digambarkan. Jalur alternatif pertama adalah sebagai berikut:



Gambar 14. Rute Alternatif 1 Operasi Militer CSG-2 secara keseluruhan
Sumber: dokumentasi pribadi

Berdasarkan gambar tersebut, dilakukan penyederhanaan dengan cara penggambaran rute menggunakan 9 simpul dan 9 sisi untuk mempermudah pemodelan dan perhitungan. Jalur alternatif pertama adalah sebagai berikut:

Gambar 15. Rute Alternatif 2 Operasi Militer CSG-2 secara keseluruhan
Sumber: dokumentasi pribadi

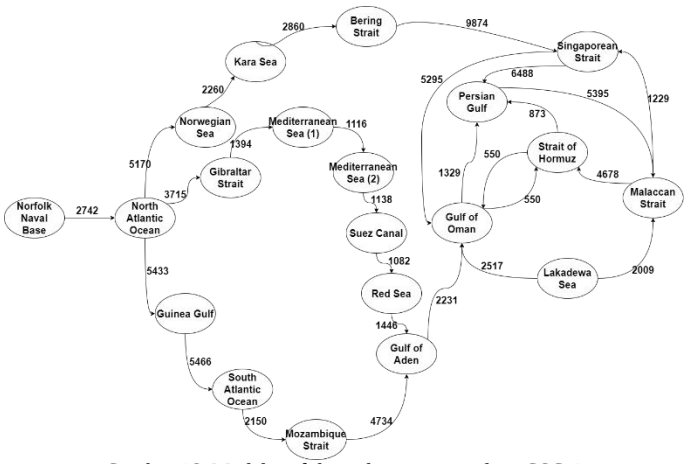
Berdasarkan gambar tersebut, dilakukan penyederhanaan dengan cara penggambaran rute menggunakan 11 simpul dan 11 sisi untuk mempermudah pemodelan dan perhitungan.

B. Pemodelan Graf

Setelah mendapatkan rute utama dan rute alternatif, pemodelan jalur operasi militer CSG-2 dapat dilakukan. Pemodelan graf dilakukan dengan membuat graf berbobot dan berarah berdasarkan rute operasi militer yang telah didapatkan untuk mendapatkan jalur dengan jarak terpendek dan tercepat. Simpul dalam graf merupakan perairan (laut, samudera, selat, terusan, ataupun teluk) yang dilewati CSG-2 dalam perjalanan, sedangkan sisi yang digambarkan secara berarah merupakan lintasan pelayaran yang dilalui CSG-2 antara tiap perairan yang digambarkan dengan simpul. Pada graf, juga terdapat bobot yang merepresentasikan jarak antar sisi (jarak antar simpul).

Demi menyederhanakan proses pemodelan graf serta mempermudah proses perhitungan, asumsi jarak dibuat lebih sederhana serta tidak setiap perairan terhubung oleh perairan lain. Faktor kecepatan kapal (knots), status perairan, hubungan Amerika Serikat dengan negara pemilik perairan yang dilewati, serta keamanan perairan yang dilewati diabaikan.

Secara keseluruhan, graf yang telah dimodelkan mengandung 20 buah simpul dari 3 rute yang telah digabung (jalur utama, jalur alternatif 1, dan jalur alternatif 2). Simpul-simpul yang ada antara lain Norfolk Naval Base, North Atlantic Ocean, Gibraltar Strait, Mediterranean Sea (1), Mediterranean Sea (2), Suez Canal, Red Sea, Gulf of Aden, Mozambique Strait, South Atlantic Ocean, Guinea Gulf, Norwegian Sea, Kara Sea, Bering Strait, Singaporean Strait, Malaccan Strait, Lakadewa
Sea, Gulf of Oman, Strait of Hormuz, dan Persian Gulf. Bobot pada tiap sisi bervariasi, bergantung pada jarak simpul dengan simpul berikutnya. Penggambaran graf secara keseluruhan adalah sebagai berikut.



Gambar 16. Model graf dari jalur operasi militer CSG-2
Sumber: dokumentasi pribadi

C. Source Code dalam Python

Setelah memodelkan graf, program dibuat dalam bahasa pemrograman Python dengan memanfaatkan algoritma A* yang telah dirumuskan sebelumnya. Dari graf yang telah digambarkan tersebut, dibuat pemodelan graf dalam bentuk adjacency array yang ditulis sebagai dictionary/ hash table dengan pasangan key dan value masing-masing. Key merupakan perairan-perairan (simpul), sedangkan value merupakan jarak perairan (bobot). Array tersebut dimodelkan sesuai arah pada masing-masing simpul. Array adjacent yang dibuat adalah sebagai berikut.

```
array_adjacent_military = {
    "norfolk" : [("north_atlantic", 2742)],
    "north_atlantic" : [("norwegian_sea", 5170), ("gibraltar_strait", 3715), ("guniea_gulf",
    5433)],
    "norwegian_sea" : [("kara_sea", 2260)],
    "kara_sea" : [("bering_strait", 2860)],
    "bering_strait" : [("singaporean_strait", 9874)],
    "gibraltar_strait" : [("mediterranean_sea1", 1394)],
    "mediterranean_sea1" : [("mediterranean_sea2", 1116)],
    "mediterranean_sea2" : [("suez_canal", 1138)],
    "suez_canal" : [("red_sea", 1082)],
    "red_sea" : [("gulf_of_aden", 1446)],
    "guniea_gulf" : [("south_atlantic_ocean", 5466)],
    "south_atlantic_ocean" : [("mozambique_strait", 2150)],
    "mozambique_strait" : [("gulf_of_aden", 4734)],
    "singaporean_strait" : [("persian_gulf", 6488), ("gulf_of_oman", 5295)],
    "persian_gulf" : [("malaccan_strait", 5395)],
    "malaccan_strait" : [("singaporean_strait", 1229)],
    "strait_of_hormuz" : [("persian_gulf", 873), ("gulf_of_oman", 550)],
    "gulf_of_oman" : [("persian_gulf", 1329), ("strait_of_hormuz", 550)],
    "lakadewa_sea" : [("gulf_of_oman", 2517), ("malaccan_strait", 2009)]
}
```

Gambar 17. Pemodelan array adjacent dalam Python
Sumber: dokumentasi pribadi

```
class grafMiliter:
    def _init_(self, array_adj):
        self.array_adj = array_adj
    def adjacency(self, perairan):
        return self.array_adj[perairan]
    def heucs(self, jumlahPerairan):
        H = {"norfolk" : 1, "north_atlantic" : 1, "norwegian_sea" : 1, "kara_sea" : 1, "bering_strait" : 1,
        "gibraltar_strait" : 1, "mediterranean_sea1" : 1, "mediterranean_sea2" : 1, "suez_canal" : 1, "red_sea" : 1,
        "guniea_gulf" : 1, "south_atlantic_ocean" : 1, "mozambique_strait" : 1, "singaporean_strait" : 1,
        "persian_gulf" : 1, "malaccan_strait" : 1, "strait_of_hormuz" : 1, "gulf_of_oman" : 1, "lakadewa_sea" : 1,}
        return H[jumlahPerairan]
```

Gambar 18. Proses inisiasi dalam Python
Sumber : dokumentasi pribadi

Berdasarkan gambar 18, terdapat sebuah class grafMiliter yang merupakan sebuah class dalam Python untuk membuat sebuah array adjacent di dalamnya. Terdapat method init dengan array_adj sebagai parameter. Method init ini merupakan constructor dan bertugas menginisiasi sebuah instance dari suatu class. Dalam method tersebut terdapat self.array_adj yang merupakan sebuah variabel instance yang menyimpan daftar simpul-simpul adjacency/ketetanggaan (array_adj) untuk

nantinya digunakan di dalam class. Terdapat pula method "adjacency" yang berfungsi untuk mengambil parameter nextPerairan dan me-*return* array_adj untuk *key* nextPerairan yang diberikan. Terdapat metode "heucs" yang merupakan heuristic. Metode ini memiliki parameter jumlahPerairan dan me-return nilai heuristik yang terkait dengan lokasi perairan tersebut. Nilai-nilai heuristik telah ditentukan sebelumnya dalam kamus H. Nilai-nilai heuristik adalah konstan (diasumsikan nilai heuristik nya adalah 1) dan digunakan dalam beberapa algoritma graf, seperti algoritma pencarian A*.

```
def jarak_terkecil(self, initial, final):
    openList = [initial]
    closedList = []
    adj_map = {initial: initial}
    currDistance = {initial: 0}
    while len(openList) > 0:
        cek = None
        for perairan in openList:
            if cek is None or ((currDistance[perairan] + self.heucs(perairan)) <
                               (currDistance[cek] + self.heucs(cek))):
                cek = perairan
        if cek is None:
            print("Jarak terdekat rute not found.")
            return None
        for (node, weight) in self.adjacency(cek):
            if node not in openList and node not in closedList:
                openList.append(node)
                adj_map[node] = cek
                currDistance[node] = currDistance[cek] + weight
            else:
                if currDistance[node] > currDistance[node] + weight:
                    currDistance[node] = currDistance[node] + weight
                    adj_map[node] = cek
                    if node in closedList:
                        closedList.remove(node)
                        openList.append(node)
        openList.remove(cek)
        closedList.append(cek)
        if cek == final:
            walker = currDistance[cek]
            path = []
            while adj_map[cek] != cek:
                path.append(cek)
                cek = adj_map[cek]
            path.append(initial)
            path.reverse()
            print("Jarak rute pelayaran = %d km" % walker)
            print("Jarak terdekat dalam rute pelayaran =", initial, end='')
            for i in path:
                if i != initial:
                    print(" ----> ", i, end='')
            return path
```

Gambar 19. Function jarak_terkecil
Sumber: dokumentasi pribadi

Gambar 19 merupakan function jarak_terkecil. Di dalam function tersebut, terdapat variable openList yang merupakan array perairan yang masih perlu dieksplorasi, variabel closedList yang merupakan array perairan yang sudah dieksplorasi, variabel adj_map yang merupakan pemetaan untuk melacak jalur terpendek serta variabel currurrDistance yang merupakan map (pemetaan) untuk melacak jarak terpendek.

Terdapat while loop yang akan berjalan selama masih ada variabel perairan dalam openList. Di dalam while loop terdapat algoritma seleksi untuk memilih perairan dengan jarak terpendek dari openList secara heuristik. Terdapat juga handling untuk adjacency yang melakukan tterasi melalui array adjacency perairan yang dipilih. Apabila perairan yang akan dimasukkan sudah ada pada openList dan closedList, maka program akan memperbarui jarak terpendek jika ditemukan jarak yang lebih kecil. Namun, jika perairan belum ada di openList dan closedList, maka program akan menambahkannya ke openList dan memperbarui informasi jalur dan jarak terpendek. Program juga akan menghapus perairan yang tidak sesuai dari openList dan menambahkannya ke closedList. Apabila perairan yang terpilih adalah tujuan (*final*), maka program akan menghasilkan jalur yang dilewati dan melakukan print informasi jarak terdekat ke dalam terminal.

```
print("Input posisi awal:")
initial = input()
print("Input posisi akhir (tujuan):")
final = input()



graph    =    grafMiliter(array_adjacent_military)
graph.jarak_terkecil(initial, final)
```

Gambar 20. Input berupa posisi awal dan akhir untuk dapat menjalankan program
Sumber: dokumentasi pribadi

Pada gambar 20, program akan menerima input berupa posisi awal (*initial*) dan posisi akhir (*final)*. Program juga akan membuat object dari class grafMiliter dengan menggunakan array adjacency yang telah didefinisikan sebelumnya (array_adjacent_military). Program akan memanggil method jarak_terkecil pada object graph dengan menggunakan posisi awal (*initial*) dan posisi akhir (*final*) yang telah dimasukkan oleh user sebelumnya.

```
PS D:\Perkuliahan\SMT 3\Matdis - Bulik Ulfa\makalahku> d:; cd 'd:\Perkuliahan\SMT 3\Matdis - Bulik Ulfa\makalahku'; & 'C:\Users\user\AppData
\Local\Programs\Python\Python39\python.exe' 'c:\Users\user\.vscode\extensions\ms-python.python-2023.22.0\pythonFiles\lib\python\debugpy\adapt
er/../..\debugpy\launcher' '58618' '--' 'D:\Perkuliahan\SMT 3\Matdis - Bulik Ulfa\makalahku\plye_yo.py'
Input posisi awal (initial):
Input posisi awal (initial):
norfolk
persian_gulf
Jarak rute pelayaran = 16193 km
Jarak terkecil dalam rute pelayaran = norfolk ----> north_atlantic ----> gibraltar_strait ----> mediterranean_sea1 ----> mediterranean_sea2 ----> suez_c
anal ----> red_sea ----> gulf_of_aden ----> gulf_of_oman ----> persian_gulf
```

Gambar 21. Hasil program saat dijalankan
Sumber: dokumentasi pribadi

Gambar 21 merupakan hasil program saat telah dijalankan. Sebagai simpul asal atau posisi awal (initial), dimasukkan Norfolk yang merupakan Norfolk Naval Station dan simpul tujuan atau posisi akhir (final) dimasukkan Persian_gulf yang merupakan Persian Gulf. Didapat output program berupa jarak rute pelayaran tercepat sebesar 16193 km dan jarak terkecil dalam rute pelayaran adalah norfolk - north_atlantic - gibraltar_strait - mediterranean_sea1 - mediterranean_sea2 - suez_canal - red_sea - gulf_of_aden - gulf_of_oman - persian_gulf.

IV. KESIMPULAN

Graf merupakan salah satu algoritma yang sangat berguna dalam memodelkan suatu persoalan secara matematis. Dalam hal ini, penggunaan graf sudah sangat sesuai dengan persoalan mencari jalur tercepat dalam rute operasi militer *Carrier Strike Group-2* USS Dwight D. Eissenhower (CVN-69). Melalui penggabungan graf dengan Algoritma A*, persoalan dalam mencari rute tercepat dapat diselesaikan. Dalam penelitian ini, digunakan graf berbobot dan berarah yang memetakan hubungan tiap-tiap perairan yang dilewati CSG-2 dengan 3 macam jalur pelayaran untuk menyelesaikan persoalan. Jarak antar tiap perairan digambarkan melalui bobot dalam pemodelan graf di persoalan ini sehingga solusi mencari rute terdekat dari dermaga awal (Norfolk Naval Station) menuju ke perairan Persian Strait bisa didapatkan.

REFERENCES

[1] https://informatika.stei.itb.ac.id/~rinaldi.munir/ diakses pada 10 Desember 2023, pukul 08.00
[2] https://news.usni.org/category/fleet-tracker diakses pada 10 Desember 2023, pukul 18.00
[3] https://www.freemaptools.com/measure-distance.htm diakses pada 10 Desember 2023, pukul 19.00
[4] https://www.researchgate.net/figure/Testing-A-Star-algorithm-on-graph_fig5_348997309 diakses pada 10 Desember 2023, pukul 22.00
[5] https://worldview.stratfor.com/article/us-naval-update-map-oct-19-2023 diakses pada 10 Desember 2023, pukul 23.49
[6] Dalem, Ida Bagus Gede Wahyu Antara. "Penerapan algoritma A*(Star) menggunakan graph untuk menghitung jarak terpendek." Jurnal RESISTOR (Rekayasa Sistem Komputer) 1, no. 1 (2018): 41-47.

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, 11 Desember 2023

Eduardus Alvito Kristiadi
13522004