Penerapan *Dynamic Programming* dalam Menentukan Rute Chey-Yo Terpendek pada Robot ABU Robocon 2023

Muhammad Fadhil Amri - 13521066
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13521066@std.stei.itb.ac.id

***Abstrak*— ABU(Asia-Pacific) Robot Contest adalah kompetisi robot yang diikuti oleh universitas-universitas di wilayah Asia Pasifik. Tema ABU Robocon 2023 adalah *Casting Flowers Through Angkor Wat*. Inti dari permainan ini adalah dua tim dipertandingkan untuk saling melempar *ring* ke *pole* yang telah disediakan. *Pole* yang disediakan telah dikonfigurasi susunannya sehingga memiliki bentuk yang menyerupai Angkor Wat. Salah satu tipe kemenangan dalam permainan ini adalah Chey-Yo. Chey-Yo adalah suatu kondisi saat *pole* yang disyaratkan dimiliki hanya oleh suatu tim. Untuk mencapai Chey-Yo dengan cepat, robot harus bisa mencapai semua target dengan jalur terpendek dan melempar dari jarak yang dekat agar akurat. Pada makalah ini pemilihan rute Chey-Yo dilakukan dengan memanfaatkan *dynamic programming* agar panjang lintasan yang dilalui oleh robot dapat menjadi semninimum mungkin. Dalam pengujian program, disimpulkan bahwa solusi yang dihasilkan optimal dengan menggunakan *cost* yang diperoleh dari penghitungan *Euclidean distance*.**

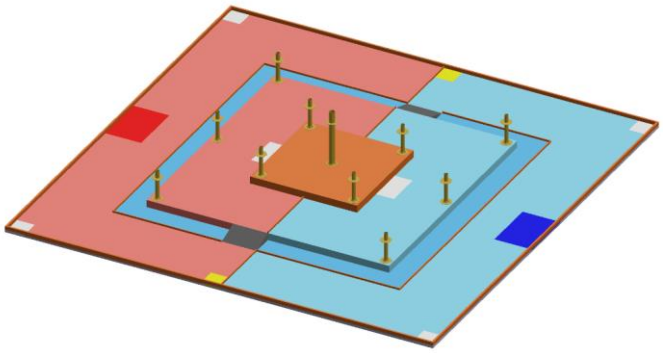
Kata Kunci—Robot, Dynamic, Rute, Chey-Yo, Minimum

I. PENDAHULUAN

Perlombaan ABU Robocon merupakan kontes internasional dalam bidang robotika yang memiliki tema yang bervariasi setiap tahunnya. Perlombaan ini diikuti oleh berbagai tim dari beragam universitas di wilayah Asia Pasifik. Perwakilan Indonesia dipilih melalui Kontes Robot Indonesia dalam bidang lomba Kontes Robot ABU Indonesia (KRAI). Perlombaan tahun 2023 memiliki tema "*Casting Flowers Through Angkor Wat*". Tema ini terinspirasi dari budaya masyarakat Kamboja yang melemparkan bunga di dalam tarian tradisionalnya dan juga di dalam penyambutan tamu. Selain itu, tema ini juga terinspirasi dari permainan tradisional Kamboja berupa permainan melempar koin ke sebuah area melingkar pada tanah.

Tugas dari robot yang dibuat untuk mengikuti perlombaan ini adalah melempar *ring* ke *pole-pole* yang ada pada lapangan permainan. Lapangan dari permainan ini memiliki desain menyerupai candi. Untuk memperoleh poin, robot harus bisa memasukkan *ring* ke dalam *pole* yang memiliki poin yang berbeda tergantung tipenya. Robot dari suatu tim yang mengikuti perlombaan ini bisa berjumlah satu atau dua. Tim yang memiliki satu dan tim yang memiliki dua robot tidak dibedakan dalam kompetisi tersebut sehingga pemenang dari permainan mutlak ditentukan berdasarkan poin dan kondisi Chey-Yo. Chey-Yo adalah kondisi saat suatu tim memiliki semua *Pole* kecuali *Pole* ada pada area lawan.

Ring yang terdapat pada suatu *pole* dapat ditimpa oleh *ring* lain. Namun, ring yang terhitung sebagai poin adalah *ring* yang berada paling atas pada *pole*. Dalam proses pelemparan *ring*, kedua robot dapat bekerja sama dan dapat bekerja secara parallel. Akan tetapi, pada makalah ini kasus yang dibahas hanya mengacu pada salah satu dari dua robot yang ada. Robot yang dibahas juga melakukan pelemparan dari jarak dekat untuk menjamin keakuratan dari pelemparan *ring*. Oleh karena itu, rute perjalanan robot memiliki peran yang signifikan dalam meningkatkan peluang kemenangan sebuah tim.



Gambar 1.1 Ilustrasi arena perlombaan ABU Robocon 2023

Sumber: [2]

Dynamic programming adalah sebuah teknik yang sering digunakan dalam optimasi karena keandalannya dalam menangani persoalan dengan hasil yang optimal, tetapi tetap memiliki kompleksitas yang lebih rendah dibandingkan dengan *exhaustive search*. *Dynamic programming* dipilih karena persoalan penentuan rute Chey-Yo terpendek termasuk ke dalam persoalan minimasi. *Dynamic programming* dirasa paling sesuai untuk memecahkan permasalahan ini karena gerakan robot menuju *pole-pole* untuk Chey-Yo dapat dimodelkan sebagai tahapan-tahapan yang saling berkaitan. Selain itu, *dynamic programming* juga sesuai karena dapat

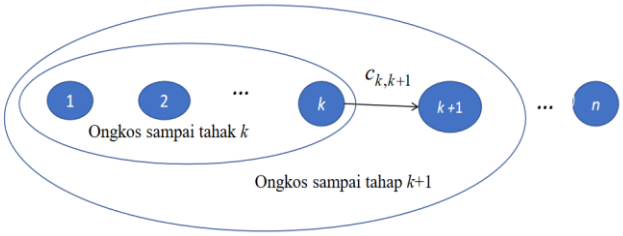
memecahkan permasalahan optimasi TSP yang mirip dengan permasalahan ini.

II. TEORI DASAR

A. *Dynamic Programming*

Dynamic programming adalah sebuah teknik pemrograman komputer yang bekerja dengan cara memecah masalah menjadi beberapa tahapan yang saling berkaitan. Setiap terjadi pemecahan masalah pada suatu tahap, hasilnya disimpan untuk digunakan pada tahap selanjutnya. Hal ini menyebabkan *dynamic programming* selalu lebih baik dibandingkan dengan proses rekursif murni karena tidak perlu melakukan pemrosesan ulang untuk hasil yang telah didapatkan dari proses pada tahap-tahap sebelumnya. Pada setiap tahapan dilakukan optimasi untuk menemukan solusi menyeluruh dari sebuah permasalahan.

Dynamic programming sering digunakan untuk memecahkan masalah optimasi karena hasil yang diperoleh dengan teknik ini dijamin selalu optimal, tetapi kompleksitasnya lebih rendah daripada *exhaustive search*. *Dynamic programming* memecahkan permasalahan dengan mempertimbangkan lebih dari satu rangkaian keputusan sehingga solusi optimal yang diperoleh adalah solusi optimal global. Rangkaian keputusan yang optimal diperoleh dengan menggunakan prinsip optimalitas, yaitu jika solusi total optimal, bagian solusi sampai tahap ke-k juga optimal.



Gambar 2.1 Ilustrasi Prinsip Optimalitas

Sumber: [8]

Dynamic programming dapat menggunakan pendekatan maju atau mundur. Pendekatan maju melakukan penghitungan dari tahap pertama ke tahap terakhir, sedangkan pendekatan mundur melakukan penghitungan dengan arah sebaliknya. Meskipun *dynamic programming* dapat memecahkan permasalahan dengan dua pendekatan, terdapat karakteristik dari permasalahan yang bisa dipecahkan dengan baik menggunakan *dynamic programming*. Berikut beberapa karakteristik tersebut.

1. Permasalahan dapat dibagi menjadi beberapa tahapan dan hanya diambil satu keputusan untuk setiap tahapan.
2. Setiap tahapan terdiri atas beberapa status yang berhubungan dengan tahap tersebut.
3. Hasil keputusan yang diambil pada setiap tahap ditransformasikan dari suatu status ke tahap selanjutnya.
4. *Cost* meningkat secara teratur seiring meningkatnya jumlah tahapan dan bergantung pada *cost* tahap-tahap sebelumnya serta *cost* ke tahap berikutnya.
5. Terdapat hubungan rekursif yang menyebabkan prinsip optimalitas berlaku pada permasalahan.

Dynamic programming secara umum terdiri atas Langkah-langkah berikut.

1. Karakteristikkan struktur solusi optimal

   Langkah ini menekankan pada proses pendefinisian tahap, variabel keputusan, status, dan solusi.

2. Definisikan hubungan rekursif solusi optimal

   Langkah ini menekankan pada proses pendefinisian hubungan antara nilai optimal suatu tahap dengan nilai optimal tahap sebelumnya.

   $F_1(s) = c_{x1,\,s}$ (Basis)

   $F_k(s) = \min \{f_{k-1}(x_k) + c_{xk,\,s}\}$ (Rekurens)

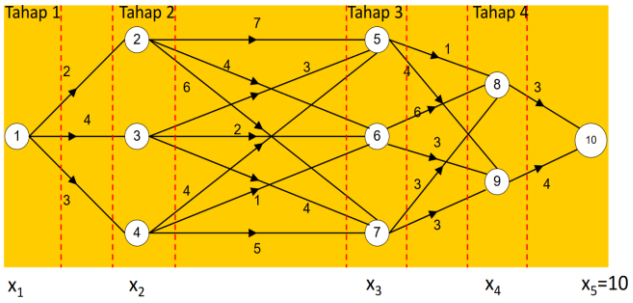
   k = 1, 2, .., n

3. Hitung nilai solusi optimal secara maju atau mundur

   Langkah ini bisa dilakukan menggunakan tabel untuk memetakan hubungan tahapan sebelumnya dan status pada tahap sekarang untuk melakukan optimasi.

4. Rekonstruksi solusi optimal secara mundur.

   Langkah ini adalah sebuah langkah opsional jika dibutuhkan rekonstruksi solusi dari suatu permasalahan.



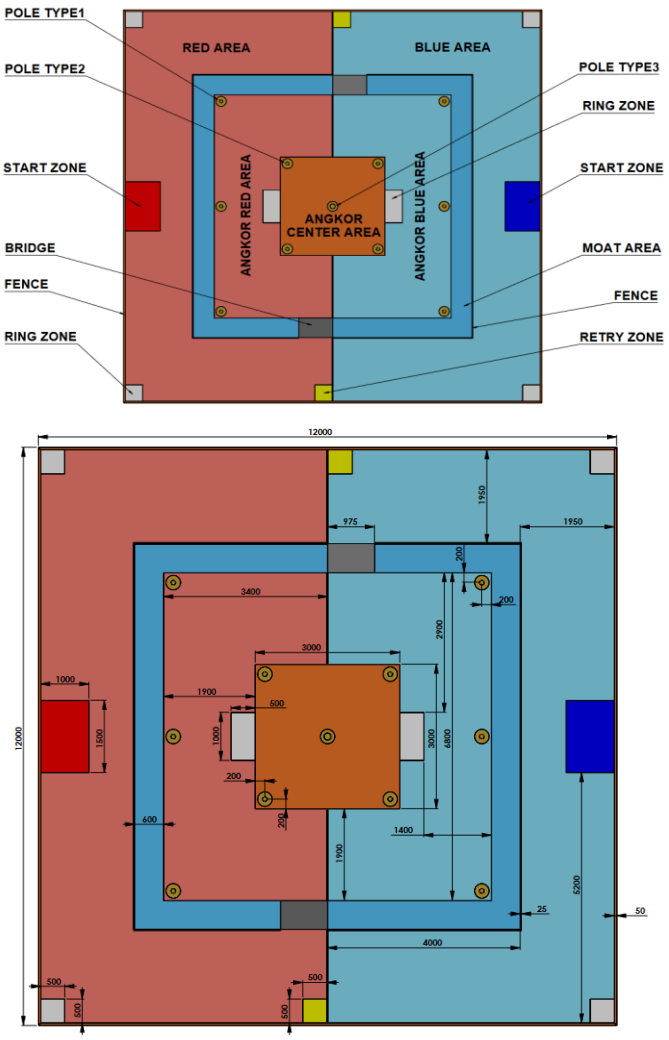
Gambar 2.2 Contoh pemodelan tahapan *dynamic programming*

Sumber: [8]

B. *Permainan Casting Flower Over Angkor Wat ABU Robocon 2023*

Permainan *Casting Flower Over Angkor Wat* ini mempertandingkan dua tim yang berasal dari dua universitas yang berbeda. Permainan dengan waktu normal berlangsung selama tiga menit. *Flower* pada permainan ini dilambangkan dengan *ring*. Setiap tim bisa memiliki maksimal dua buah robot, *elephant robot(ER)* dan *rabbit robot(RR)*. Kedua robot ini dapat bekerja sama untuk melemparkan *ring* yang telah disediakan ke dalam *pole* yang terdapat pada *Angkor Wat Area* (*Angkor red area*, *Angkor blue area*, dan *Angkor center area*). Sebagai tambahan, *Angkor center area* adalah area netral

sehingga kedua tim dapat memasuki area tersebut. Namun, ER tidak diperbolehkan untuk memasuki *Angkor Wat Area* sehingga pada makalah ini hanya akan dibahas tentang konfigurasi lapangan dan rute perjalanan dari RR yang dapat bergerak bebas di *Angkor Wat Area*.



Gambar 2.3 Detail Arena perlombaan ABU Robocon 2023

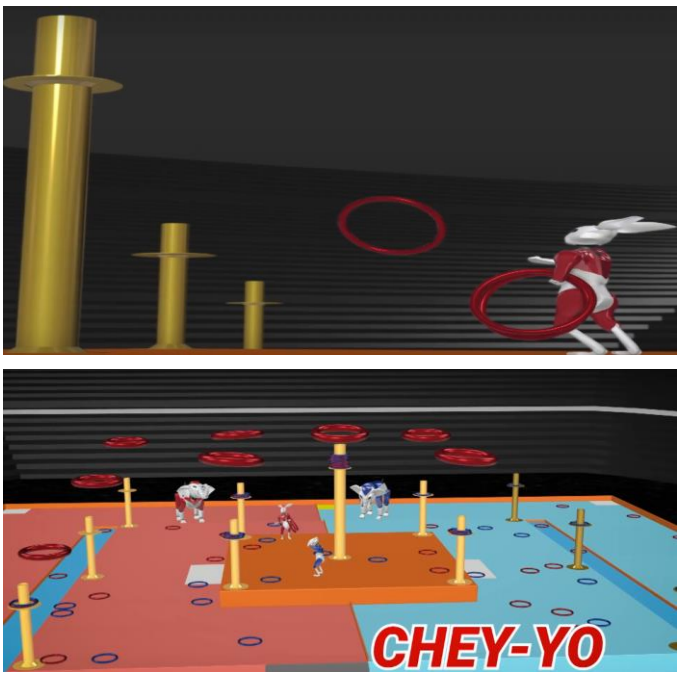
Sumber: [2]

Rabbit robot (RR) dapat bergerak pada *blue area* atau *red area* kecuali *moat area* tergantung warna tim. RR memulai permainan dari *start zone*, tetapi dapat juga dimulai dari *retry zone* jika RR telah memasuki *Angkor Wat Area* dan melakukan *retry* saat terjadi suatu masalah pada RR. *Ring* yang digunakan oleh RR dapat diambil dari *ring zone*. Pada makalah ini pembahasan mengasumsikan bahwa RR sudah membawa *ring* tanpa harus memikirkan proses pengambilannya karena hal tersebut bukanlah pokok pembahasan dari makalah ini. Dengan demikian, pembahasan pada makalah bisa difokuskan pada rute yang ditempuh RR untuk memasukkan *ring* ke dalam setiap *pole*.

Permainan berakhir saat salah satu tim dinyatakan sebagai pemenang. Kemenangan pada permainan ini ditentukan oleh jumlah poin yang diperoleh dari setiap tim saat waktu permainan habis atau terdapat tim yang memperoleh Chey-Yo selama permainan berlangsung. Poin yang diperoleh saat memasukkan *ring* ke dalam *pole* berbeda untuk tipe *pole* yang berbeda. *Pole* tipe 1 memiliki 10 poin, *Pole* tipe 2 memiliki 30 poin, dan *Pole* tipe 3 memiliki 70 poin. Tim yang diakui untuk mendapatkan poin dari suatu pole adalah tim yang berhasil menempatkan *ring* timnya pada posisi teratas pada *pole* sehingga jika terdapat *ring* tim lain yang berada di bawahnya, *ring* tersebut tidak diakui untuk mendapatkan poin. Meskipun terdapat dua buah tipe kemenangan, pembahasan pada makalah ini hanya akan difokuskan pada tipe kemenangan Chey-Yo karena Chey-Yo merupakan permasalahan yang dapat dipastikan kemenangannya dan dapat diselesaikan menggunakan *dynamic programming*.

C. Chey-Yo

Chey-Yo adalah salah satu tipe kemenangan pada perlombaan ABU Robocon 2023. Chey-Yo merupakan kondisi saat suatu tim berhasil menempatkan ringnya pada posisi teratas pada delapan *pole*. Delapan *pole* yang dimaksud adalah semua *pole* tipe 1 tim (tiga buah), semua *pole* tipe 2 (empat buah), dan *pole* tipe 3 (satu buah). Pendekatan yang akan digunakan untuk memperoleh Chey-Yo pada pembahasan makalah ini adalah dengan menghampiri delapan *pole* yang dibutuhkan. Hal ini ditujukan agar akurasi dari RR tidak menjadi suatu permasalahan sehingga keberhasilan memperoleh Chey-Yo lebih cepat hanya difokuskan pada proses pengambilan jalur yang ditempuh oleh RR. Faktor-faktor eksternal lain yang mempengaruhi proses memperoleh Chey-Yo, seperti *ring* yang datang dari tim lain atau *ring* yang dilemparkan RR tidak berhasil memasuki *pole* dihilangkan karena bukan merupakan permasalahan dalam mencari rute terpendek.



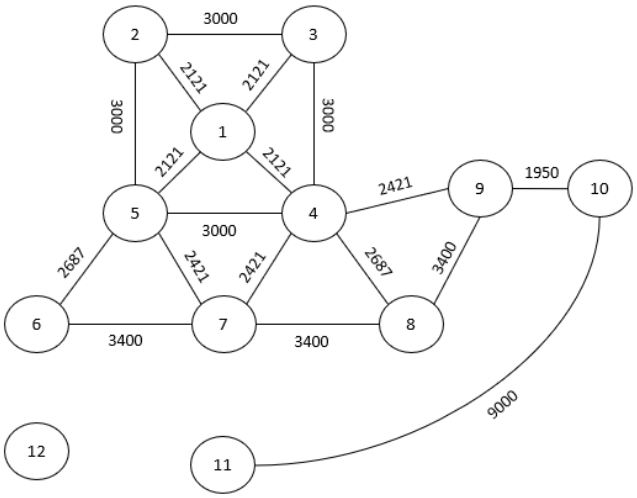
Gambar 2.3 Ilustrasi RR Melempar *ring* dan Chey-Yo

Sumber: [4]

III. PEMBAHASAN

A. *Deskripsi Persoalan*

Lapangan permainan pada pembahasan ini menggunakan hanya sebagian lapangan untuk tim merah karena hanya berfokus pada Chey-Yo. Lapangan ini dimodelkan dalam graf dengan simpul-simpulnya adalah titik-titik penting yang ada pada lapangan. Setiap simpul dinamai dengan angka 1 sampai 12. Angka 1 melambangkan *pole* tipe 3 yang berada di tengah-tengah *Angkor Center Area*. Angka 2 sampai 5 melambangkan *pole* tipe 2 yang berada pada setiap pojok *Angkor Center Area*. Angka 6 sampai 8 melambangkan *pole* tipe 1 milik tim yang bersangkutan. Angka 9 meambangkan *bridge* yang menghubungkan antara *red area* dengan *Angkor red area*. Angka 10 melambangkan *retry zone* yang berfungsi sebagai tempat RR memulai ulang permainan jika terjadi masalah atau permintaan *retry* jika sebelumnya RR sudah memasuki *Angkor wat area*. Angka 11 melambangkan *start zone* yang merupakan tempat RR saat permainan pertama kali dimulai. Angka 12 melambangkan robot RR. Posisi simpul 12 dapat berubah-ubah sesuai dengan kondisi penempatan RR saat ingin dicari rute Chey-Yo terpendek.



Gambar 3.1 Model Graf dari Lapangan ABU Robocon 2023

Sumber: Dokumen Pribadi

Sisi yang menghubungkan antarsimpul pada graf diperoleh dengan menghitung *Euclidean distance* dalam satuan milimeter. Terdapat beberapa simpul yang tidak terhubung secara langsung. Hal itu terjadi karena ada daerah yang tidak bisa dilalui oleh RR sehingga harus terhubung melalui simpul perantara. Selain itu, robot yang didesain bisa bergerak secara horizontal, vertical, dan diagonal juga membuat munculnya sisi-sisi yang beririsan sehingga untuk mempermudah pemecahan masalah. Sisi-sisi yang beririsan dihilangkan.

B. *Algoritma*

Berikut algoritma yang digunakan untuk pencarian rute Chey-Yo terpendek.

```
public double fun(Integer newNode, List<Integer> path)
{
    String key = newNode.toString() + path.toString();
    // System.out.println(key);
    if (path.size() == 0){ // Basis
        dpTable.put(key, graf.get(startNode).get(newNode));
        return graf.get(startNode).get(newNode);
    } else { // Basis
        if (dpTable.containsKey(key)){
            return dpTable.get(key);
        } else { // Rekurens
            double minCost = 99999.0;
            List<Integer> minPath = new ArrayList<Integer>();
            for (int i = 0; i < path.size(); i++) {
                List<Integer> newPath = new ArrayList<Integer>(path);
                newPath.remove(i);
                double cost = fun(path.get(i), newPath) + graf.get(path.get(i)).get(newNode);
                if (cost < minCost){
                    minCost = cost;
                    minPath = new ArrayList<Integer>(newPath);
                }

            }
            dpTable.put(key, minCost);
            if (!pathTable.containsKey(key)){
                String prevkey = newNode.toString() + minPath.toString();

                if (pathTable.containsKey(prevkey)){
                    List<Integer> bestPath = new ArrayList<Integer>(pathTable.get(prevkey));
                    bestPath.add(newNode);
                    pathTable.put(key, bestPath);
                } else {
                    List<Integer> bestPath = new ArrayList<Integer>(path);
                    // bestPath.remove(minIndex);
                    pathTable.put(key, bestPath);
                }
            } else {
                List<Integer> bestPath = new ArrayList<Integer>(pathTable.get(key));
                bestPath.add(newNode);
                pathTable.put(key, bestPath);
            }
            return minCost;
        }
    }
}
```

Gambar 3.2 Implementasi Dynamic Programming

Sumber: Dokumen Pribadi

Berikut penjelasan secara umum algoritma yang digunakan (gambar yang dilampirkan hanya bagian algoritma TSP yang dimodifikasi sisanya ada pada repository di lampiran).

1) Lakukan inisialisasi graf, yaitu dengan melakukan *setting coordinate* dan matriks ketetanggaan berbobot yang berisi jarak antar simpul.

2) Minta masukan koordinat dari RR, lalu lakukan inisialisasi *path* dan *cost* agar RR dapat berada pada *Angkor Wat Area*. Dalam tahap ini terdapat pemrosesan berikut.

- Apabila posisi RR yang dimasukkan berada di luar lapangan atau di dalam area yang dilarang, tampilkan pesan kesalahan
- Apabila posisi RR berada pada *red area* dengan koordinat x lebih kecil daripada koordinat x *start zone*, RR harus melewati *start zone* terlebih dahulu, lalu ke *retry zone*, dan ke *bridge*.
- Apabila posisi RR berada pada *red area*, tetapi koordinat x lebih besar daripada koordinat x *start zone* dan lebih kecil daripada koordinat x *bridge*, RR harus melewati *retry zone* terlebih dahulu, lalu ke *bridge*.
- Apabila posisi RR berada pada *red area*, tetapi koordinat x lebih besar daripada koordinat x *bridge*, RR dapat langsung menuju *bridge*.
- Apabila posisi RR berada pada *Angkor wat area*, RR diposisikan ke *pole* terdekat.

- Apabila setelah dipindahkan, posisi RR berada pada *bridge*, RR dipindahkan menuju *pole* terdekat dari *bridge*.

3) Lakukan penghitungan *total cost* dengan memanggil fungsi fun() yang mengimplementasikan *dynamic programming* untuk mendapatkan rute Chey-Yo terpendek

4) Pada fungsi fun() terdapat pemrosesan berikut.

- Jika berada pada basis, yaitu fun(*node*, []), isi tabel dengan jarak dari *node* menuju *start node*, lalu kembalikan nilainya
- Apabila fungsi dengan atribut yang sama telah pernah dipanggil, nilai kembalian langsung diambil dari *table* yang menyimpan nilai kembalian fungsi tersebut.
- Apabila berada pada rekurens, saat fungsi yang dipanggil belum pernah dipanggil sebelumnya (tidak ada nilai kembalian pada tabel), lakukan pencarian nilai minimum dengan persamaan minimasi yang ada pada teori dasar.
- Saat memasuki rekurens, lakukan *update* pada pathTable yang berisi *path* dari tur yang optimal.

5) Tampilkan rute Chey-Yo *terpendek* dan *total cost*-nya

C. Pengujian

Pada pembahasan ini, dilakukan pengujian sebanyak 7 kasus. Pengujian ini ditujukan untuk melihat program yang memanfaatkan *dynamic programming* dapat memecahkan persoalan pencarian rute Chey-Yo terpendek. Berikut beberapa pengujian tersebut.

1. Kasus RR berada di luar lapangan

```
Masukkan koordinat x:
-1
Masukkan koordinat y:
-1
Koordinat start node tidak valid
```

Pada kasus ini, pesan kesalahan dimunculkan saat dilakukan inisialisasi dari *start node* karena didapati bahwa RR memiliki koordinat yang tidak valid karena berada di luar lapangan.

2. Kasus RR berada pada *moat area*

```
Masukkan koordinat x:
2000
Masukkan koordinat y:
2000
Koordinat start node berada di tempat yang dilarang
```

Pada kasus ini, pesan kesalahan dimunculkan saat dilakukan pemeriksaan klasifikasi dari posisi RR. Saat didapati bahwa RR berada pada area terlarang seperti area musuh ataupun *moat area*.

3. Kasus RR berada pada *red area* di kiri *start zone*

```
Masukkan koordinat x:
100
Masukkan koordinat y:
100
Jalur terpendek yang dapat ditempuh oleh CheyYo adalah:
11 10 9 8 3 0 1 2 4 5 6 7
Dengan total jarak: 41687.84739677278
```

Pada kasus ini, proses penghitungan berhasil. RR terlebih dahulu digerakkan menuju *start zone*, lalu dilanjutkan menuju *retry zone*, *bridge*, dan *pole* 4 (3 karena pergeseran indeks).

4. Kasus RR berada pada *red area* di kanan *start zone* dan di kiri *bridge*

```
Masukkan koordinat x:
7000
Masukkan koordinat y:
1000
Jalur terpendek yang dapat ditempuh oleh CheyYo adalah:
11 9 8 3 0 1 2 4 5 6 7
Dengan total jarak: 33858.067811865476
```

Pada kasus ini, proses penghitungan berhasil. RR terlebih dahulu digerakkan menuju *retry zone*, *bridge*, dan *pole* 4 (3 karena pergeseran indeks).

5. Kasus RR berada pada *red area* di kanan *bridge*

```
Masukkan koordinat x:
11000
Masukkan koordinat y:
1000
Jalur terpendek yang dapat ditempuh oleh CheyYo adalah:
11 8 3 0 1 2 4 5 6 7
Dengan total jarak: 29936.019513592786
```

Pada kasus ini, proses penghitungan berhasil. RR terlebih dahulu digerakkan menuju *bridge*, dan *pole* 4 (3 karena pergeseran indeks).

6. Kasus RR berada pada *angkor wat area*

```
Masukkan koordinat x:
6000
Masukkan koordinat y:
6000
Jalur terpendek yang dapat ditempuh oleh CheyYo adalah:
11 0 1 2 3 4 5 6 7
Dengan total jarak: 22416.0
```

Pada kasus ini, proses penghitungan berhasil. RR langsung ditempatkan pada *pole* terdekat, yaitu *pole* 1 (0 karena pergeseran indeks).

7. Kasus RR berada pada *angkor wat area*

```
Masukkan koordinat x:
7000
Masukkan koordinat y:
4000
Jalur terpendek yang dapat ditempuh oleh CheyYo adalah:
11 3 0 1 2 4 5 6 7
Dengan total jarak: 23123.106781186547
```

Pada kasus ini, proses penghitungan berhasil. RR langsung ditempatkan pada *pole* terdekat, yaitu *pole* 4 (3 karena pergeseran indeks).

IV. PENUTUP

A. Simpulan

Dynamic programming dapat digunakan untuk memperoleh hasil yang optimal dalam mencari rute Chey-Yo terpendek. Pemanfaatan prinsip optimalitas membuat *dynamic programming* memberikan hasil yang selalu optimal global. Selain itu, Penggunaan *tabulation* dalam *dynamic programming* juga mampu memberikan peningkatan kecepatan dalam menangani pemanggilan rekursif yang sama berulang

kali. Hal inilah yang menjadikan *dynamic programming* menjadi suatu pilihan algoritma yang menjanjikan.

Algoritma yang dibuat pada makalah ini sangat bergantung pada manufaktur fisik dan fungsional robot. Pada makalah ini, kondisi yang dimiliki oleh robot adalah kondisi ideal dengan asumsi setiap pelemparan *ring* yang dilakukan oleh robot selalu masuk ke dalam *pole* dan tidak adanya *ring* musuh yang mengambil alih *pole* yang telah diambil oleh *rabbit robot* (RR) sebelumnya.

B. Saran

Simplifikasi yang dilakukan masih cukup kasar sehingga *reliability* dari pemecahan masalah ini masih kurang. Pertimbangan lebih lanjut diperlukan untuk memperoleh jalur yang benar-benar optimal karena terdapat perbedaan ketinggian dari lapangan yang dapat menyebabkan perbedaan kecepatan tempuh robot.

UCAPAN TERIMA KASIH

Puji dan syukur penulis panjatkan ke hadirat Allah SWT atas limpahan Rahmat, karunia, dan hidayah-Nya sehingga penulis dapat menyelesaikan penulisan makalah ini dengan baik. Penulis berterima kasih kepada keluarga dan teman-teman yang telah memberi dukungan kepada penulis dalam menyelesaikan makalah ini. Penulis juga berterima kasih kepada dosen pengampu mata kuliah Strategi Akgoritma Semester 4 2022/2023 kelas K02, Ibu Dr. Nur Ulfa Maulidevi, S.T., M.Sc. atas pengajaran yang beliau berikan sehingga saya bisa memahami materi, terutama materi *dynamic programming* ini dengan lebih baik. Terakhir, penulis memohon maaf apabila di dalam penulisan makalah ini terdapat kesalahan baik disengaja, maupun tidak disengaja. Penulis berharap makalah ini dapat bermanfaat bagi banyak orang.

REFERENSI

[1] ABU Robocon 2023 Online Rulebook. Diakses melalui https://www.aburobocon2023.com/game-rules pada 21 Mei 2023

[2] ABU Robocon 2023 Online GameField. Diakses melalui https://www.aburobocon2023.com/_files/ugd/cd69bb_e1f81641e117463e903c234f728c0623.pdf pada 21 Mei 2023

[3] Cuemath. 2022. Euclidean Distance Formula. Diakses melalui https://www.cuemath.com/euclidean-distance-formula/ pada 21 Mei 2023

[4] Robocon Official. (2022). ABU ROBOCON 2023. Diakses melalui https://www.youtube.com/watch?v=eEnqrpgW8jU pada 22 Mei 2023

[5] Munir, Rinaldi. 2020. Graf (bag.1) Bahan Kuliah IF2120 Matematika Diskrit. Diakses melalui https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf- 2020-Bagian1.pdf pada 21 Mei 2023

[6] GeeksForGeeks. 2022. Travelling Salesman Problem using Dynamic Programming. Diakses melalui https://www.geeksforgeeks.org/travelling-salesman-problem-using-dynamic-programming/ pada 22 Mei 2023

[7] Spiceworks. 2022. What is Dynamic Programming? Working, Algorithms, and Examples. Diakses melalui https://www.spiceworks.com/tech/devops/articles/what-is-dynamic-programming/ pada 22 Mei 2023

[8] Munir, Rinaldi. 2020. Program Dinamis (bag.1) Bahan Kuliah IF2120 Matematika Diskrit. Diakses melalui https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Program-Dinamis-2020-Bagian1.pdf pada 21 Mei 2023

[9] Munir, Rinaldi. 2020. Program Dinamis (bag.2) Bahan Kuliah IF2120 Matematika Diskrit. Diakses melalui https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Program-Dinamis-2020-Bagian2.pdf pada 21 Mei 2023

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, 22 Mei 2023

Ttd

Muhammad Fadhil Amri 13521066