Nama: Firas Maulana Lasidi

NIM : 1102204539

Persiapan

- tuliskan NIM Anda, ambil 2 digit terakhir nim anda (misal 21011900XY)

- X akan menjadi 0,1,2,,...,9. Dan Y menjadi A,B,C,...,J (1=A,2=B,...,9=I,0=J)

- variasi soal akan mengikuti nilai X dan Y tersebut.

(untuk koding algoritma yang anda buat, hanya lampirkan link file-nya)

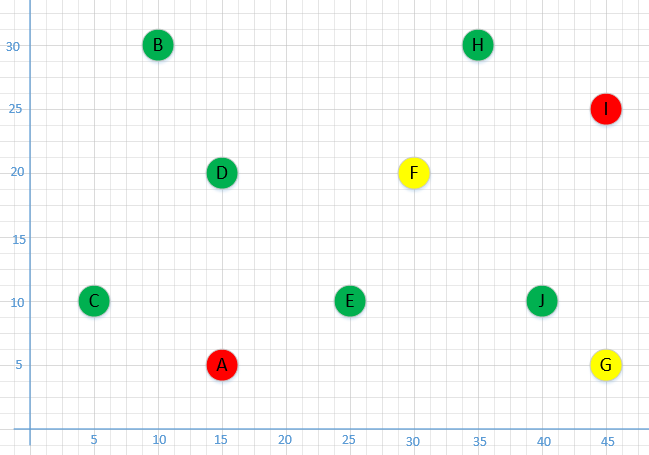
----------------------------------------------------------------------------------------------------------------------------------------------------

Seorang abang Delivery hendak mengirim paket ke 10 pelanggan yang letaknya seperti ini. Titik pertama si abang adalah titik Y (sesuai dengan catatan dibawah), dan tujuan selanjutnya adalah sedemikian sehingga supaya :

1. Semua barang terantarkan secepat mungkin
2. Pelanggan Kuning adalah pelanggan prioritas, harus diterima maksimal 2 Jam semenjak si abang mulai bergerak. Jika telat, denda Rp 5.000 tiap 30mnt keterlambatan
3. Pelanggan Merah adalah pelanggan Istimewa, harus diterima maksimal 3 Jam semenjak si abang mulai bergerak. Jika tepat waktu ada bonus tambahan Rp. 25.000, tapi jika telat, denda Rp 10.000 tiap 30mnt keterlambatan.

Kondisi :

1. Koordinat adalah dalam KM
2. Kecepatan si abang bergerak adalah 1 KM/Menit, tidak ada delay lainnya.
3. Si abang dibekali Rp. 50.000 di awal keberangkatan
4. Setiap sukses delivery si abang mendapat bonus Rp. 15.000 (apapun jenis pelanggan)
5. Setiap KM perjalanan menghabiskan bensin Rp. 1000.



Gambar 1.

Catatan :

Jika NIM anda diakhiri angka 0 🡪 pelanggan pertama (Y) adalah J,

Jika NIM anda diakhiri angka 1 🡪 pelanggan pertama (Y) adalah A,

Jika NIM anda diakhiri angka 3 🡪 pelanggan pertama (Y) adalah C, dst (lihat bagian persiapan)

Soal :

UTS-1. Buatlah perhitungan yang meminimalkan waktu pengantaran menggunakan algoritma Ant-Colony Optimization atau Particle-Swarm Optimization. Dan Sebutkan berapa waktu tempuh terkecil-nya. (opsional : buatlah coding yang mengimplementasikan algoritma penyelesaian anda)

Jawab :

Rute terbaik: ['I', 'J', 'G', 'H', 'F', 'E', 'A', 'C', 'D', 'B']

Waktu tempuh terbaik: 119 menit

UTS-2. Buatlah perhitungan yang mengoptimalkan pendapatan abang delivery menggunakan algoritma Genetik. Dan sebutkan berapa total pendapatan si Abang hari itu. (opsional : buatlah coding yang mengimplementasikan algoritma penyelesaian anda)

Jawab:

Waktu tempuh ke titik F: 15 menit

Waktu tempuh ke titik G: 20.0 menit

Waktu tempuh ke titik A: 36 menit

Rute terpendek: ('I', 'H', 'F', 'A', 'C', 'B', 'D', 'E', 'J', 'G')

Panjang rute terpendek: 122 KM

bekal abang = Rp50000

total uang abang = bekal abang + sukses delivery + bonus tambahan

total uang abang = 50000 + 15000\*10 + 50000 = Rp250000

bensin yang dihabiskan Rp 179000

penghasilan abang = total uang abang - bensin yang dihabiskan

penghasilan abang = 250000 - 147000 = Rp 103000

UTS-3. Jika ada peta jalan seperti pada gambar di gambar 2 dan jarak-garis-lurus (heuristic) dari kota2 itu ada di gambar 2,

Diagram, radar chart

Description automatically generated

Gambar 2. Peta jarak antar kota

Tabel 2. Jarak ‘garis-lurus’ (heuristic) antar kota

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | #A | #B | #C | #D | #E | #F | #G | #H | #I | #J | #ABC |
| #0 | 300 | 250 | 275 | 265 | 750 | 700 | 690 | 460 | 650 | 645 | 640 |
| #1 | 370 | 320 | 345 | 335 | 820 | 770 | 760 | 490 | 720 | 715 | 710 |
| #2 | 500 | 425 | 450 | 350 | 790 | 740 | 730 | 480 | 690 | 685 | 680 |
| #3 | 510 | 470 | 490 | 450 | 700 | 680 | 630 | 685 | 580 | 625 | 590 |
| #4 | 305 | 325 | 280 | 250 | 650 | 600 | 590 | 420 | 550 | 545 | 570 |
| #5 | 375 | 325 | 350 | 340 | 795 | 745 | 735 | 485 | 695 | 690 | 695 |
| #6 | 335 | 285 | 310 | 280 | 640 | 590 | 580 | 410 | 450 | 465 | 455 |
| #7 | 395 | 365 | 370 | 340 | 635 | 585 | 575 | 430 | 440 | 455 | 445 |
| #8 | 405 | 375 | 380 | 350 | 630 | 580 | 470 | 450 | 430 | 445 | 455 |
| #9 | 325 | 345 | 430 | 280 | 630 | 580 | 450 | 480 | 350 | 370 | 360 |
| #11 | 470 | 450 | 460 | 350 | 400 | 395 | 340 | 405 | 180 | 350 | 210 |
| #12 | 420 | 385 | 390 | 260 | 380 | 365 | 310 | 375 | 140 | 260 | 101 |
| #13 | 370 | 355 | 350 | 170 | 415 | 390 | 340 | 340 | 240 | 265 | 211 |

carilah rute terbaik dari kota #Y ke kota #X (jika nim anda 2102180101, maka dari kota #J ke kota #1), dengan menggunakan:

a. metode Breadth First Search

b. metode Depth First Search

c. metode Uniform Cost Search

d. metode Greedy Search

c. metode A\* (A-star)

Jelaskan/deskripsikan tiap tahapannya!.  
(opsional : buatlah coding yang mengimplementasikan algoritma penyelesaian anda)

Jawab:

1. metode Breadth First Search
2. Mulai dari titik awal (atau simpul awal) dan tandai simpul tersebut sebagai sudah dikunjungi. Kemudian tambahkan simpul tersebut ke dalam antrian (queue) untuk proses pencarian selanjutnya.
3. Ambil simpul pertama dari antrian, lalu ekspansi simpul tersebut dengan menemukan semua simpul tetangga yang belum dikunjungi. Tandai setiap simpul tetangga yang baru ditemukan sebagai sudah dikunjungi dan tambahkan ke dalam antrian.
4. Ulangi proses ekspansi pada setiap simpul yang ada dalam antrian, sampai semua simpul dalam graf telah dikunjungi atau sampai titik tujuan ditemukan.
5. Selama proses pencarian, simpan jalur yang telah dilalui dari titik awal ke setiap simpul yang dikunjungi. Dengan cara ini, setelah titik tujuan ditemukan, jalur yang ditempuh dapat dikembalikan.
6. Selama ekspansi simpul, pastikan simpul yang baru ditemukan belum pernah dikunjungi sebelumnya. Hal ini mencegah terjadinya siklus dalam graf.
7. BFS menggunakan antrian (queue) untuk menyimpan simpul yang akan diekspansi. Antrian ini mengikuti konsep "First-In-First-Out" (FIFO), yang berarti simpul yang pertama kali dimasukkan ke dalam antrian akan diekspansi terlebih dahulu.

* Node yang Dikunjungi: {'I'}
* Node yang Dikunjungi: {'I', 'ABC'}
* Node yang Dikunjungi: {'13', 'I', 'ABC'}
* Node yang Dikunjungi: {'13', '12', 'I', 'ABC'}
* Node yang Dikunjungi: {'13', '12', 'J', 'I', 'ABC'}
* Node yang Dikunjungi: {'13', '12', 'J', 'I', '4', 'ABC'}
* Node yang Dikunjungi: {'13', '12', 'J', 'I', '11', '4', 'ABC'}
* Node yang Dikunjungi: {'13', '12', 'J', '9', 'I', '11', '4', 'ABC'}
* Node yang Dikunjungi: {'13', '12', 'J', '9', 'I', '11', '4', 'F', 'ABC'}
* Node yang Dikunjungi: {'13', 'G', '12', 'J', '9', 'I', '11', '4', 'F', 'ABC'}
* Node yang Dikunjungi: {'13', 'G', '12', 'J', '9', 'I', '11', 'H', '4', 'F', 'ABC'}
* Node yang Dikunjungi: {'13', 'G', '12', 'J', '9', 'I', '11', '0', 'H', '4', 'F', 'ABC'}
* Node yang Dikunjungi: {'13', 'G', '12', 'J', '9', 'I', '11', '0', '2', 'H', '4', 'F', 'ABC'}
* Node yang Dikunjungi: {'13', 'G', '12', 'J', '9', 'I', '11', '0', '2', 'H', '1', '4', 'F', 'ABC'}
* Node yang Dikunjungi: {'13', 'G', '12', 'J', '9', 'I', '11', '0', '2', 'H', '8', '1', '4', 'F', 'ABC'}
* Node yang Dikunjungi: {'13', 'E', 'G', '12', 'J', '9', 'I', '11', '0', '2', 'H', '8', '1', '4', 'F', 'ABC'}
* Node yang Dikunjungi: {'13', 'E', 'G', '12', 'J', '9', 'I', '11', '0', '2', 'H', '8', '1', 'D', '4', 'F', 'ABC'}
* Node yang Dikunjungi: {'13', 'E', 'G', '12', 'J', '9', 'I', '11', '0', '2', 'H', '8', '1', 'D', 'C', '4', 'F', 'ABC'}
* Rute terbaik: ['I', 'ABC', '13', '4', '2', '3']
* Jarak Total: 658

1. metode Depth First Search
2. Mulai dari titik awal (atau simpul awal) dan tandai simpul tersebut sebagai sudah dikunjungi. Kemudian tambahkan simpul tersebut ke dalam tumpukan (stack) untuk proses pencarian selanjutnya.
3. Ambil simpul teratas dari tumpukan, lalu ekspansi simpul tersebut dengan menemukan satu simpul tetangga yang belum dikunjungi. Tandai simpul tetangga tersebut sebagai sudah dikunjungi dan tambahkan ke dalam tumpukan.
4. Ulangi proses ekspansi pada simpul tetangga yang baru saja ditemukan. Proses ini dilakukan secara rekursif sampai tidak ada lagi simpul tetangga yang dapat diekspansi atau sampai titik tujuan ditemukan.
5. Pastikan simpul yang baru ditemukan belum pernah dikunjungi sebelumnya. Hal ini mencegah terjadinya siklus dalam graf.
6. DFS menggunakan tumpukan (stack) untuk menyimpan simpul yang akan diekspansi. Tumpukan ini mengikuti konsep "Last-In-First-Out" (LIFO), yang berarti simpul yang terakhir dimasukkan ke dalam tumpukan akan diekspansi terlebih dahulu.
7. Selama proses pencarian, simpan jalur yang telah dilalui dari titik awal ke setiap simpul yang dikunjungi. Dengan cara ini, setelah titik tujuan ditemukan, jalur yang ditempuh dapat dikembalikan.

Langkah Pencarian:

* Mulai dari I dan pergi ke ABC
* Mulai dari ABC dan pergi ke 13
* Mulai dari 13 dan pergi ke 4
* Mulai dari 4 dan pergi ke 9
* Mulai dari 9 dan pergi ke 11
* Mulai dari 11 dan pergi ke 8
* Mulai dari 8 dan pergi ke 7
* Mulai dari 7 dan pergi ke 6
* Sampai di 3
* Rute Terbaik: ['I', 'ABC', '13', '4', '9', '11', '8', '7', '6', '3']
* Jarak Total: 1002

1. metode Uniform Cost Search
2. Mulai dari titik awal (atau simpul awal) dan tambahkan simpul tersebut ke dalam antrian prioritas (priority queue) dengan biaya 0. Simpan juga biaya sejauh ini untuk mencapai setiap simpul dalam sebuah kamus.
3. Ambil simpul dengan biaya terendah dari antrian prioritas. Ini berarti simpul dengan biaya terendah akan diekspansi terlebih dahulu.
4. Untuk setiap simpul yang diekspansi, tinjau setiap tetangganya. Jika tetangga belum pernah dikunjungi sebelumnya atau terdapat jalur baru dengan biaya yang lebih rendah, tambahkan tetangga tersebut ke antrian prioritas dengan biaya total yang diperbarui.
5. Perbarui biaya untuk mencapai setiap simpul jika jalur baru dengan biaya yang lebih rendah ditemukan.
6. Saat mengekspansi simpul, periksa apakah simpul yang diekspansi adalah titik tujuan yang dicari. Jika iya, hentikan pencarian dan kembalikan jalur serta biaya total yang diperlukan.
7. UCS menggunakan antrian prioritas untuk menyimpan simpul yang akan diekspansi. Dalam antrian ini, simpul dengan biaya yang lebih rendah akan diberikan prioritas lebih tinggi sehingga akan diekspansi terlebih dahulu.

Langkah pencarian :

* Ekspansi node I, tambahan: ABC, jarak total: 90
* Ekspansi node ABC, tambahan: 13, jarak total: 301
* Ekspansi node ABC, tambahan: 12, jarak total: 191
* Ekspansi node ABC, tambahan: J, jarak total: 175
* Ekspansi node J, tambahan: F, jarak total: 273
* Ekspansi node J, tambahan: G, jarak total: 244
* Ekspansi node J, tambahan: H, jarak total: 317
* Ekspansi node 12, tambahan: 11, jarak total: 329
* Ekspansi node 12, tambahan: 9, jarak total: 288
* Ekspansi node F, tambahan: E, jarak total: 359
* Ekspansi node 9, tambahan: 4, jarak total: 368
* Ekspansi node H, tambahan: D, jarak total: 417
* Ekspansi node H, tambahan: C, jarak total: 409
* Ekspansi node 11, tambahan: 8, jarak total: 449
* Ekspansi node 4, tambahan: 0, jarak total: 503
* Ekspansi node 4, tambahan: 2, jarak total: 508
* Ekspansi node 4, tambahan: 1, jarak total: 519
* Ekspansi node C, tambahan: A, jarak total: 479
* Ekspansi node C, tambahan: B, jarak total: 496
* Ekspansi node 8, tambahan: 7, jarak total: 524
* Ekspansi node 2, tambahan: 5, jarak total: 583
* Ekspansi node 2, tambahan: 3, jarak total: 626
* Ekspansi node 7, tambahan: 6, jarak total: 594
* Route terbaik: ['I', 'ABC', '12', '9', '4', '2', '3']

1. metode Greedy Search
2. Mulai dari simpul awal dan tambahkan simpul tersebut ke dalam daftar atau antrian prioritas (priority queue).
3. Ambil simpul dari daftar atau antrian prioritas. Simpul yang dipilih adalah simpul yang paling "dekat" dengan tujuan menurut fungsi heuristik yang diberikan. Fungsi heuristik ini biasanya memberikan perkiraan jarak dari simpul saat ini ke tujuan.
4. Untuk simpul yang diekspansi, tinjau semua tetangganya. Tetangga yang belum dieksplorasi akan ditambahkan ke dalam daftar atau antrian prioritas. Pemilihan tetangga biasanya didasarkan pada nilai heuristik, di mana tetangga yang lebih "dekat" dengan tujuan akan diberikan prioritas.
5. Saat mengekspansi simpul, periksa apakah simpul yang diekspansi adalah tujuan yang dicari. Jika iya, hentikan pencarian dan kembalikan jalur serta biaya total yang diperlukan.
6. Fungsi heuristik digunakan untuk mengevaluasi setiap simpul dan memberikan perkiraan seberapa dekat simpul tersebut dengan tujuan. Fungsi ini harus adil dan konsisten untuk menghasilkan solusi optimal.
7. Greedy Search menggunakan pendekatan "greedy" di mana simpul yang dipilih pada setiap langkah adalah simpul yang paling "dekat" dengan tujuan menurut fungsi heuristik. Ini tidak menjamin solusi optimal, tetapi sering kali memberikan solusi yang cukup baik dalam waktu yang cepat.

Langkah pencarian :

* Ekspansi node I, tambahan: ABC, jarak total: 90
* Ekspansi node ABC, tambahan: 13, jarak total: 211
* Ekspansi node ABC, tambahan: 12, jarak total: 101
* Ekspansi node 12, tambahan: 11, jarak total: 138
* Ekspansi node 12, tambahan: 9, jarak total: 97
* Ekspansi node 9, tambahan: 4, jarak total: 80
* Ekspansi node 4, tambahan: 0, jarak total: 135
* Ekspansi node 11, tambahan: 8, jarak total: 120
* Ekspansi node 8, tambahan: 7, jarak total: 75
* Ekspansi node 7, tambahan: 6, jarak total: 70
* Ekspansi node 6, tambahan: 3, jarak total: 111
* Route: ['I', 'ABC', '12', '11', '8', '7', '6', '3']
* Total Distance: 111

1. metode A\* (A-star)
2. Mulai dari simpul awal dan tambahkan simpul tersebut ke dalam daftar terbuka (open list). Set nilai g(n) (biaya sejauh ini) untuk simpul awal menjadi 0 dan set nilai h(n) (estimasi biaya dari simpul ke tujuan) berdasarkan fungsi heuristik.
3. Ambil simpul dari daftar terbuka yang memiliki nilai f(n) terkecil, di mana f(n) = g(n) + h(n). Simpul yang dipilih adalah simpul yang paling obyektif mengarah ke tujuan.
4. Untuk simpul yang diekspansi, tinjau semua tetangganya. Hitung biaya sejauh ini (g(n)) untuk setiap tetangga dengan menambahkan biaya dari simpul awal ke tetangga tersebut. Hitung juga estimasi biaya dari tetangga ke tujuan (h(n)) berdasarkan fungsi heuristik.
5. Jika tetangga belum ada di daftar terbuka atau memiliki biaya yang lebih rendah, perbarui nilai g(n) dan h(n) untuk tetangga tersebut, serta atur simpul induk sebagai simpul penghubung (parent).
6. Saat mengekspansi simpul, periksa apakah simpul yang diekspansi adalah tujuan yang dicari. Jika iya, hentikan pencarian dan rekonstruksi jalur dari simpul awal ke simpul tujuan dengan mengikuti simpul induk yang telah disimpan.
7. Fungsi heuristik digunakan untuk memberikan estimasi biaya dari setiap simpul ke tujuan. Fungsi ini harus admissable (tidak melebih biaya sebenarnya) dan konsisten (monotonik) untuk menjamin optimalitas algoritma.

Langkah pencarian :

* Step: I -> ABC, Cost: 90, Priority: 90
* Step: ABC -> 13, Cost: 301, Priority: 301
* Step: ABC -> 12, Cost: 191, Priority: 191
* Step: 12 -> 11, Cost: 329, Priority: 329
* Step: 12 -> 9, Cost: 288, Priority: 288
* Step: 9 -> 4, Cost: 368, Priority: 368
* Step: 11 -> 8, Cost: 449, Priority: 449
* Step: 4 -> 0, Cost: 503, Priority: 503
* Step: 8 -> 7, Cost: 524, Priority: 524
* Step: 7 -> 6, Cost: 594, Priority: 594
* Step: 6 -> 3, Cost: 705, Priority: 705
* Route: ['I', 'ABC', '12', '11', '8', '7', '6', '3']
* Total Distance: 705

Total Distance: 626 UTS-4. Rancanglah suatu algoritma Fuzzy-Logic yang memiliki minimal 2 Input, 1 Output, beberapa aturan (rules) dan berkaitan dengan bidang aplikasi seperti yg tertera pada table 3 dibawah.

Tabel 3. Variasi soal berdasarkan X dan Y (dari NIM)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| X | Minimum Jenis Input-1 | Minimum Jenis Input-2 | Minimum jumlah Rules | Minimum Jenis Output |  | Y | Bidang Aplikasi |
| #0 | 3 | 2 | 4 | 5 |  | #A | Pertanian/ Perkebunan |
| #1 | 3 | 3 | 4 | 4 |  | #B | Lingkungan Hidup |
| #2 | 3 | 4 | 4 | 3 |  | #C | Telekomunikasi |
| #3 | 3 | 5 | 4 | 2 |  | #D | IoT / jaringan-sensor |
| #4 | 2 | 3 | 4 | 5 |  | #E | Perangkat Rumah-tangga |
| #5 | 2 | 4 | 4 | 4 |  | #F | Bencana Alam |
| #6 | 2 | 5 | 4 | 3 |  | #G | Lalu Lintas |
| #7 | 4 | 2 | 4 | 4 |  | #H | Otomotif |
| #8 | 4 | 3 | 4 | 3 |  | #I | Kesehatan |
| #9 | 4 | 4 | 4 | 2 |  | #J | Keamanan |

Desain system fuzzy-logic ini harus menyertakan:

1. Cerita latar belakang/ deskripsi sistem yang anda pilih
2. Diagram/grafik membership function dari input-1, input-2 dan output
3. Table aturan (rules) antara input menjadi output
4. Suatu contoh nilai (angka) input-input tertentu menghasilkan suatu nilai output berapa!
5. Deskripsi penjelasan tiap Langkah.
6. (opsional : buatlah coding yang mengimplementasikan algoritma penyelesaian anda)

Jawab :

a. Deskripsi Sistem:

Sistem ini digunakan untuk memantau kesehatan seseorang berdasarkan tekanan darah dan kadar gula darah. Kesehatan ditentukan berdasarkan kedua faktor ini, dengan kategori baik atau buruk. Ini membantu individu untuk memahami kondisi kesehatan mereka dan memberikan saran jika diperlukan.

b. Diagram/Grafik Membership Function:

* Input-1 (Tekanan Darah):
* Input-2 (Kadar Gula Darah):
* Output (Kesehatan):

c. Tabel Aturan:

|  |  |  |
| --- | --- | --- |
| Tekanan Darah (Input-1) | Kadar Gula Darah (Input-2) | Kesehatan (Output) |
| Rendah | Rendah | Buruk |
| Rendah | Normal | Buruk |
| Normal | Rendah | Buruk |
| Tinggi | Tinggi | Buruk |
| Normal | Normal | Baik |
| Tinggi | Normal | Baik |
| Normal | Tinggi | Baik |
| Tinggi | Rendah | Baik |

d. Contoh Nilai Input-Input dan Output:

Jika tekanan darah = 0.4 (rendah) dan kadar gula darah = 0.6 (normal), maka kesehatan = 0.3 (buruk).

e. Deskripsi Langkah:

Input Fuzzy: Ambil nilai tekanan darah dan kadar gula darah.

Fuzzifikasi: Tentukan derajat keanggotaan masing-masing nilai input pada himpunan fuzzy.

Aturan Fuzzy: Gunakan tabel aturan untuk menentukan kontribusi masing-masing aturan terhadap output.

Aggregasi: Gabungkan kontribusi dari setiap aturan untuk mendapatkan nilai output fuzzy.

Defuzzifikasi: Konversi nilai output fuzzy menjadi nilai tunggal untuk output.

