EL2208 Praktikum Pemecahan Masalah dengan C (PPMC) TA 2023/2024

Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung

Pembuat Naskah : Jessen Javier Kurniawan (EB'20), Dr. Beni Rio Hermanto, S.T, MM., Dr. Rahadian Yusuf, S.T., M.T., Dr. Reza Darmakusuma, S.T, M.T.

Tugas besar ini bertujuan untuk menguji pemahaman dan eksplorasi mahasiswa tentang konsep-konsep dasar pada praktikum ini khususnya terkait Algoritma dan Struktur Data yang diimplementasikan dalam konteks Pemecahan Masalah dengan C.

#1 : Maze Problem

Labirin adalah struktur data dua dimensi yang terdiri dari sel-sel, di mana beberapa sel adalah dinding yang tidak dapat dilalui dan yang lainnya adalah jalur yang dapat dilalui. Tujuan dari proyek ini adalah untuk menemukan jalur dari titik awal ke titik akhir dalam labirin.

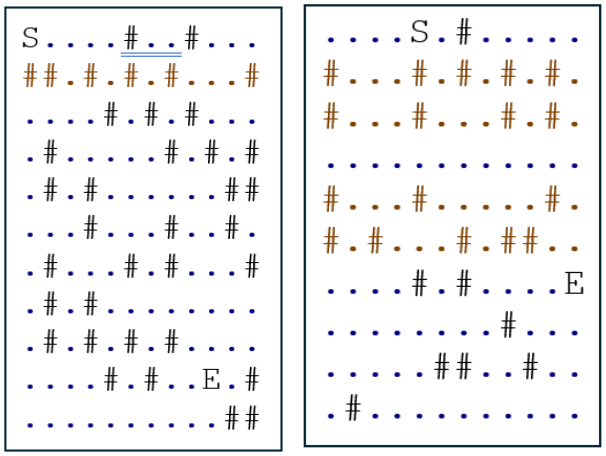
Setiap Anggota kelompok **diwajibkan** untuk menyelesaikan problem dengan **memilih 1 buah metode algoritma yang** kemudian akan dibandingkan dan dianalisis dari segi **k**ompleksitas, waktu, dan Tingkat efisiensinya.

Jenis Metode Alternatif yang dapat dipakai :

1. Algoritma Djikstra
2. Breadth First Search (BFS)
3. Depth First Search (DFS)
4. Dynamic Programming
5. Algoritma A* (A-Star)
6. Algoritma Backtracking
7. Algoritma Greedy
8. Divide & Conquer Algorithm

Spesifikasi Tugas :

1. **Implementasi Labirin**
   1) Buat program yang dapat membaca masukan dari file teks yang mewakili labirin. Setiap karakter dalam file mewakili satu sel dalam labirin, di mana karakter "#" menunjukkan dinding yang tidak dapat dilalui dan karakter "." menunjukkan jalur yang dapat dilalui. Sel "S" menunjukkan posisi Start dan Sel "End menunjukkan posisi End (Selesai). Contoh Struktur File txt dapat dilihat sebagai berikut (Maze.txt) :



   2) Labirin dapat berukuran berbeda-beda, dan program harus mampu menangani labirin dengan ukuran minimal 7 x 7. Dimensi labirin tidak harus memiliki Panjang dan lebar yang sama.

2. **Implementasi Metode Algoritma**
   1) Implementasikan metode algoritma yang anda pilih.
   2) Pastikan algoritma tersebut mampu menemukan jalur dari titik awal ke titik akhir dalam labirin.
   3) Program harus dapat mencetak seluruh kemungkinan jalur yang dapat dilalui, mencetak jalur terpendek, dan mencetak jalur terpanjang.

3. **Pengujian Kinerja**
   1) Lakukan pengujian kinerja antar metode dengan menggunakan berbagai labirin yang berbeda ukuran dan kompleksitasnya.
   2) Catat waktu yang dibutuhkan oleh masing-masing algoritma untuk menyelesaikan labirin tersebut. Di dalam program, tambahkan kode untuk mengukur waktu yang dibutuhkan oleh masing-masing algoritma) untuk menyelesaikan labirin. Anda dapat menggunakan fungsi seperti *clock()* atau *gettimeofday()* di C.

3) Setelah selesai menjalankan semua pengujian, analisis hasilnya. Perhatikan waktu yang dibutuhkan oleh masing-masing algoritma untuk menyelesaikan labirin dengan ukuran dan kompleksitas yang berbeda. Anda dapat membuat grafik atau tabel untuk membandingkan waktu yang diperlukan oleh setiap algoritma.
4) Berdasarkan hasil pengujian, buatlah kesimpulan tentang kinerja relatif dari algoritma – algoritma yang dipilih. Apakah ada tren yang menarik? Algoritma mana yang lebih efisien untuk labirin dengan ukuran atau kompleksitas tertentu?
5) Bandingkan hasil pengujian tersebut dan analisis keunggulan serta kelemahan dari masing-masing algoritma.

Contoh Eksekusi Program dapat dilihat dibawah ini :

Note : Apabila anda dapat mendesain format eksekusi program yang lebih mudah terbaca oleh pengguna akan diberikan poin tambahan sebesar maksimal 10 poin.

```
Masukkan File Txt Struktur Maze : Maze1.txt

All possible paths from start to end:

Path 1: (0, 4) -> (1, 4) -> (2, 4) -> (3, 4) -> (4, 4) -> (5, 4) -> (5, 5) ->
(6, 5)
Path 2: (0, 4) -> (1, 4) -> (2, 4) -> (3, 4) -> (4, 4) -> (5, 4) -> (5, 3) ->
(6, 3) -> (6, 4) -> (6, 5)
Path 3: (0, 4) -> (1, 4) -> (2, 4) -> (3, 4) -> (4, 4) -> (5, 4) -> (5, 3) ->
(6, 3) -> (6, 2) -> (6, 1) -> (6, 0) -> (5, 0) -> (4, 0) -> (3, 0) -> (2, 0)
-> (1, 0) -> (0, 0) -> (0, 1) -> (0, 2) -> (0, 3) -> (1, 3) -> (2, 3) -> (3,
3) -> (4, 3) -> (5, 3) -> (5, 4) -> (5, 5) -> (6, 5)
Path 4: (0, 4) -> (1, 4) -> (2, 4) -> (3, 4) -> (4, 4) -> (5, 4) -> (5, 3) ->
(6, 3) -> (6, 2) -> (6, 1) -> (6, 0) -> (5, 0) -> (4, 0) -> (3, 0) -> (2, 0)
-> (1, 0) -> (0, 0) -> (0, 1) -> (0, 2) -> (0, 3) -> (1, 3) -> (2, 3) -> (3,
3) -> (4, 3) -> (5, 3) -> (5, 4) -> (6, 4) -> (6, 5)

Total number of paths: 4

Longest path from start to end:

Path: (0, 4) -> (1, 4) -> (2, 4) -> (3, 4) -> (4, 4) -> (5, 4) -> (5, 3) ->
(6, 3) -> (6, 2) -> (6, 1) -> (6, 0) -> (5, 0) -> (4, 0) -> (3, 0) -> (2, 0)
-> (1, 0) -> (0, 0) -> (0, 1) -> (0, 2) -> (0, 3) -> (1, 3) -> (2, 3) -> (3,
3) -> (4, 3) -> (5, 3) -> (5, 4) -> (6, 4) -> (6, 5)

Shortest path from start to end:

Path: (0, 4) -> (1, 4) -> (2, 4) -> (3, 4) -> (4, 4) -> (5, 4) -> (5, 5) ->
(6, 5)
```

Related Topic : Pointer & Function, File Processing, Advanced Algorithms

#2 The Travelling Salesman Problem

Seorang pedagang ingin menjual barang dagangannya ke sejumlah kota. Untuk menghemat waktu dan uang, dia perlu menentukan rute perjalanan terpendek yang membawanya mengunjungi setiap kota dalam daftar tujuannya tepat satu kali dan kembali ke kota keberangkatannya.

Masalah ini dikenal sebagai ***the Travelling Salesman Problem*** (TSP). Dari himpunan kota beserta lokasinya dan sebuah kota keberangkatan, anda ditugaskan mencari rute perjalanan terpendek yang melalui setiap kota tepat satu kali dan kembali ke kota keberangkatan. Sebagai contoh, bila kita berada di kota Bandung, dan ingin pergi ke Bali, Semarang, dan Surabaya. Maka, rute perjalanan terpendeknya adalah Bandung-Semarang-Surabaya-Bali-Bandung.

Pada soal ini, anda akan diberikan informasi daftar kota yang ingin dikunjungi beserta *latitude* dan *longitude* dari koordinat Bumi setiap kota sebagai *input*. Berikut adalah contoh informasi yang akan anda peroleh :

No	Nama Kota	*Latitude* (°)	*Longitude* (°)
1	Bandung	-6.9175	107.6191
2	Bali	-8.3405	115.0920
3	Semarang	-7.0051	110.4381
4	Surabaya	-7.2575	112.7521

Untuk memperoleh jarak antara dua titik dari dua koordinat Bumi, anda dapat menggunakan formula Haversine berikut :

$$d = 2r \arcsin \sqrt{\sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos(\varphi_1)\cos(\varphi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}$$

dengan r merupakan konstanta jari-jari Bumi (6371 km), ϕ merupakan *latitude* dalam radian, dan λ merupakan *longitude* dalam radian. Bumi diasumsikan berbentuk bulat sempurna.

Anda kemudian diminta untuk memberikan rute dengan jarak total terpendek yang melalui seluruh kota dalam *file input* tepat satu kali sebelum kembali ke kota keberangkatan yang di-*input* oleh pengguna.

Pemecahan Masalah pada soal ini dapat diselesaikan dengan beberapa metode alternatif :

1. Algoritma Greedy
2. Algoritma Brute Force (Exhaustive Search)
3. Breadth First Search (BFS)
4. Depth First Search (DFS)
5. Algoritma Branch and Bound
6. Algoritma Integer Linear Programming (ILP)
7. Algoritma Genetika
8. Algoritma Ant Colony Optimization
9. Algoritma Particle Swarm Optimization

Setiap anggota kelompok **wajib** memilih 1 buah metode algoritma untuk menyelesaikan masalah.

Pada soal ini, rute yang dihasilkan diharapkan untuk memenuhi sifat berikut :

1. Memberikan rute yang memenuhi syarat : Setiap kota dikunjungi tepat satu kali, tidak kurang ataupun lebih, kecuali kota keberangkatan. Kemudian, Rute diawali dan diakhiri di kota keberangkatan.
2. Memberikan rute dengan jarak total terpendek dari seluruh kemungkinan rute yang ada.
3. Kota yang dipilih adalah kota – kota yang terdapat di wilayah Indonesia
4. Program dapat digunakan untuk mencari rute untuk minimal 6 kota dan maksimal 15 kota.

Pengujian Kinerja

1) Catat waktu yang dibutuhkan oleh masing-masing algoritma untuk menyelesaikan masalah. Anda dapat menggunakan fungsi seperti *clock()* atau *gettimeofday()* di C.
2) Setelah selesai menjalankan semua pengujian, analisis hasilnya. Perhatikan waktu yang dibutuhkan oleh masing-masing algoritma untuk menyelesaikan masalah. Anda dapat membuat grafik atau tabel untuk membandingkan waktu yang diperlukan oleh setiap algoritma.
3) Berdasarkan hasil pengujian, buatlah kesimpulan tentang kinerja relatif dari algoritma – algoritma yang dipilih. Algoritma mana yang lebih efisien?
4) Bandingkan hasil pengujian tersebut dan analisis keunggulan serta kelemahan dari masing-masing algoritma.

Segala hal yang tidak dicantumkan pada naskah soal ini dapat diasumsikan atau ditanyakan kepada asisten pembimbing kelompok.

Ketentuan Input Output

Input dari program ini terdiri atas sebuah *file* yang menyimpan daftar kota yang harus dikunjungi beserta koordinatnya dalam satuan derajat dan *input* nama kota keberangkatan yang diberikan oleh pengguna saat *runtime*. Tiap entri kota dalam *file input* dari program ini di-*format* sebagai berikut:

```
<Nama Kota>,<Koordinat Lintang>,<Koordinat Bujur>
```

Berikut adalah contoh isi *file input* yang diberikan:

```
Bandung,-6.9175,107.6191
Bali,-8.3405,115.0920
Semarang,-7.0051,110.4381
Surabaya,-7.2575,112.7521
```

File di atas menyimpan informasi dari empat kota dengan koordinatnya. Harap perhatikan bahwa program harus mampu mengenali bila *file* yang diberikan tidak ada, memiliki salah *formatting*, atau kosong. Berikut adalah beberapa ketentuan tambahan dari *input* program ini:

1. *File input* diberikan ke program dengan meng-*input* nama *file input* tersebut. *File input* disimpan dalam ekstensi *.csv.
2. Kota keberangkatan diberikan ke program dengan meng-*input* nama kota tersebut setelah meng-*input* nama *file*.

Output dari program ini adalah:

1. Rute perjalanan yang memenuhi syarat yang telah dijelaskan sebelumnya
2. Total jarak yang ditempuh dan Waktu yang diperlukan untuk menghitung solusi

Contoh Eksekusi Program :

Anda tidak harus mengikuti contoh eksekusi program ini, tetapi apabila Anda dapat mendesain format eksekusi program yang lebih mudah terbaca akan diberikan poin tambahan sebesar maksimal 10 poin. Silakan atur format eksekusi program selama masih memenuhi ketentuan input output di atas serta mudah dibaca oleh pengguna. Output dari program ini tidak akan diberikan pada autograder. Underline menunjukkan input dari pengguna.

```
kota_01.csv
Bandung,-6.9175,107.6191
Bali,-8.3405,115.0920
Semarang,-7.0051,110.4381
Surabaya,-7.2575,112.7521
```

```
#1
Enter list of cities file name: kota_01.csv
Enter starting point: Bandung
Best route found:
Bandung -> Bali -> Surabaya -> Semarang -> Bandung
Best route distance: 1691.29199 km
Time elapsed: 0.0000040000 s
```

Related Topic : Pointer & Function, File Processing, Advanced Algorithms

Kriteria Penilaian :

1. Presentasi
2. Pembagian Tugas Kerja antar Anggota (Logbook)
3. Laporan & Source Code
4. Peer-to-peer Evaluation

Informasi Kontak Asisten

NIM	Nama	ID Line
18320005	Maheswara Apta Adiyatma	daunituhijau
18320019	Eunike Kristianti	eunike07
18320012	Nicholas Manuel Tjahjadi	kageroish
18321008	Jasmine Callista Aurellie Irfan	jasminecallista
13219071	Reynaldo Averill Adji Putra	reynaldoaverill
13220018	Michael Andreas Manullang	manullangmichael
18321012	Muhammad Rafli	mraflii_2
18321016	Isnaini Azhar Ramadhan Wijaya	ramadhanw8

NIM	Nama	ID Line
18321013	Rayyi Rahmaid Adha	rayyirahmaid
13220031	Emmanuella Pramudita Rumanti	emrumanti
23223031	Agape D'sky	agapeagapedsky
13220029	Fariz Iftikhar Falakh	farizfalakh
18321011	Wikan Priambudi	wikanpriambudi
13220047	Muhammad Daris Nurhakim	darisnurhakim12
18320014	Jessen Javier Kurniawan	javier_40

~ The secret of getting ahead is getting started.