Ανάπτυξη Αλγορίθμου Δρομολόγησης με τον Αλγόριθμο Dijkstra

Αναστάσιος Βήττας 197

Εισαγωγή

Αυτή η αναφορά περιγράφει την ανάπτυξη και εφαρμογή ενός αλγορίθμου που εντοπίζει την πιο σύντομη διαδρομή σε ένα δίκτυο δρομολογίων, χρησιμοποιώντας πραγματικά δεδομένα από το αρχείο **routes.csv**. Το αρχείο περιέχει πληροφορίες για διάφορους κόμβους (π.χ., πόλεις, αεροδρόμια) και τις μεταξύ τους συνδέσεις με διαφορετικά μέσα μετακίνησης.

Δομή των Δεδομένων

Από το αρχείο **routes.csv** θα επιλέξουμε για την υλοποίηση του αλγορίθμου τις εξής στήλες:

- **source**: Η αρχική τοποθεσία (πόλη/αεροδρόμιο).
- **destination**: Η τελική τοποθεσία.
- **type**: Το είδος της μετακίνησης (π.χ., boat, bus, plane, train, truck).
- **time**: Ο χρόνος μετάβασης.
- **cost**: Το κόστος της μετάβασης.

Ανάγνωση των Δεδομένων

Ο κώδικας διαβάζει τα δεδομένα από το αρχείο **routes.csv** μέσω της συνάρτησης **read_csv** και τα τοποθετεί σε μια λίστα. Με την χρήση της συνάρτησης διαβάζουμε το αρχείο **csv** γραμμή προς γραμμή και λέξη προς λέξη με την χρήση της **csv.reader()**. Έπειτα κάνουμε κανονικοποίηση των ονομάτων των προορισμών (**from_city** και **to_city**) για να αποκλείσουμε την πιθανότητα να υπάρχουν κεφαλαίοι χαρακτήρες σε κάποιους προορισμους και σε κάποιους άλλους όχι.

Δημιουργία του Γραφήματος

Η συνάρτηση **build_graph** δημιουργεί το “γράφημα”, δηλαδή την διάσυνδεση του κάθε προορισμού με κάποιον άλλο με βάση την λίστα από το άρχειο csv που πήραμε από την **read_csv** με την εξής μορφή:

{'ABU DHABI': [('CANBERRA (USE SYDNEY)', 'plane', 24.0, 1339.0), ('LIMA', 'plane', 30.0, 1967.0), ('LONDON', 'plane', 10.0, 666.0)… κλπ.

Δηλαδή σε μια απεικόνηση ενώς γραφήματος θα μπορούσαμε να πούμε ότι από τον κόμβο Abu Dhabi πηγαίνουμε στον κόμβο Lima με βάρος 10.

Υλοποίηση του Αλγορίθμου Dijkstra

Ο αλγόριθμος Dijkstra χρησιμοποιείται για την εύρεση της συντομότερης διαδρομής. Σκοπός της συνάρτησης **dijkstra** δηλαδή είναι να βρει την βέλτιστη διαδρομή από έναν αρχικό κόμβο σε ένα τελικό κόμβο σε έναν γράφο, με το μικρότερο συνολικό βάρος. Στην συγκεκριμένη περίπτωση λύνουμε κατά χρόνο συνεπώς το βάρος μας είναι η μεταβλητή **time**.

Ο αρχικός κόμβος είναι η αφετηρία που δώσαμε από το **command line** και ο τελικός κόμβος ο προορισμός που δώσαμε.

Η συνάρτηση χρησιμοποιεί μια ουρα προτεραιότητας από την οποία κάθε φορά παίρνουμε τον κόμβο με το μικρότερο βάρος. Με την χρήση της **visited** παρακολουθούμε τους κόμβους που έχουν ήδη επισκεφθεί για να μην κάνουμε “κύκλους”. Κάθε φορά που παίρνουμε έναν κόμβο από την ουρά προτεραιότητας αθροίζουμε το βάρος της ακμής που τον συνδέει με τον προηγούμενο, με το προηγούμενο συνολικό βάρος. Η λήξη της **while** πραγματοποιείται όταν βρεθεί ο τερματικός κόμβος.

Παρουσίαση Αποτελεσμάτων

Ο κώδικας αποτελείται από δυο αρχεία.

- Το αρχείο **main.py** της κύριας συνάρτησης **main()** που επιτρέπει στον χρήστη να εισαγάγει την αρχική και τελική τοποθεσία, επιστρέφοντας τις συντομότερες διαδρομές είτε με βάση τον χρόνο είτε με βάση το κόστος.
- Το αρχείο **dijkstra_solution.py** με την τις συναρτήσεις υλοποίησης **read_csv**, **build_graph**, **dijkstra**.

Για την εκτέλεση του αρχείου γράφουμε την εξής εντολή στο command line:

```
tasosvittas@Tasoss-MacBook-Pro Dijkstra_Routes % python main.py
```

Παράδειγμα Εκτέλεσης:

- **Είσοδος**: Από την πόλη Α προς την πόλη Β.
- **Έξοδος**:

    1. Διαδρομή [Α, Γ, Β] με συνολικό χρόνο Χ ώρες.
    2. Μέσο μετακίνησης.
    3. Διαδρομή [Α, Γ, Β] με συνολικό με κόστος 50 ευρώ.

```
Enter the starting location and the destination you want to travel to.
From: (UPPERCASE ONLY)
ATHENS
To: (UPPERCASE ONLY)
CAIRO
Shortest path by time: ['ATHENS', 'SOFIA', 'CAIRO'] with total time 6.4
To travel from ATHENS to CAIRO you have to take: ['plane', 'plane']
Shortest path by cost: ['ATHENS', 'LONDON', 'CAIRO'] with total cost 513.0
To travel from ATHENS to CAIRO you have to take: ['plane', 'plane']
```

```
Enter the starting location and the destination you want to travel to.
From: (UPPERCASE ONLY)
LONDON
To: (UPPERCASE ONLY)
AMSTERDAM
Shortest path by time: ['LONDON', 'AMSTERDAM'] with total time 1.0
To travel from LONDON to AMSTERDAM you have to take: ['plane']
Shortest path by cost: ['LONDON', 'AMSTERDAM'] with total cost 60.0
To travel from LONDON to AMSTERDAM you have to take: ['plane']
```

```
Enter the starting location and the destination you want to travel to.
From: (UPPERCASE ONLY)
OSLO
To: (UPPERCASE ONLY)
KABUL
Shortest path by time: ['OSLO', 'LONDON', 'BAKU', 'TEHRAN', 'KABUL'] with total time 22.667
To travel from OSLO to KABUL you have to take: ['plane', 'plane', 'plane', 'plane']
Shortest path by cost: ['OSLO', 'LONDON', 'TEHRAN', 'BAKU', 'YEREVAN', "T'BILISI", 'KABUL'] with total cost 620.44
To travel from OSLO to KABUL you have to take: ['plane', 'plane', 'plane', 'bus', 'bus', 'bus']
```

```
Enter the starting location and the destination you want to travel to.
From: (UPPERCASE ONLY)
SINGAPORE
To: (UPPERCASE ONLY)
KIEV
Shortest path by time: ['SINGAPORE', 'AMSTERDAM', 'LONDON', 'KIEV'] with total time 17.0
To travel from SINGAPORE to KIEV you have to take: ['plane', 'plane', 'plane']
Shortest path by cost: ['SINGAPORE', 'LONDON', 'WARSAW', 'KIEV'] with total cost 551.6
To travel from SINGAPORE to KIEV you have to take: ['plane', 'plane', 'bus']
```

Συμπεράσματα

Η υλοποίηση του αλγορίθμου Dijkstra επιτρέπει την εύρεση βέλτιστων διαδρομών με βάση τα κριτήρια που θα του ανατεθούν, δηλαδή τα βάρη.