Технічні науки

УДК 004.021

Середа Дар'я Антонівна

студентка

Харківського національного університету радіоелектроніки

Середа Дарья Антоновна

студентка

Харьковского национального университета радиоэлектроники

Sereda Daria

Student of the

Kharkiv National University of Radio Electronics

Науковий керівник:

Афанасьєва Ірина Віталіївна

доцент кафедри ПІ

Харківський національний університет радіоелектроніки

АЛГОРИТМ ЗНАХОДЖЕННЯ ОПТИМІЗОВАНОГО ТА НАЙКОРОТШОГО ШЛЯХУ В ПРОГРАМНІЙ СИСТЕМІ ДЛЯ АВТОМАТИЗАЦІЇ УПРАВЛІННЯ ТАРИФІКАЦІЄЮ ТРАНСПОРТУ ТА НАДАННЯ ІНФОРМАЦІЇ ЩОДО ТРАНСПОРТНИХ МАРШРУТІВ

АЛГОРИТМ НАХОЖДЕНИЯ ОПТИМИЗИРОВАННОГО И КРАТЧАЙШЕГО ПУТИ В ПРОГРАММНОЙ СИСТЕМЕ ДЛЯ АВТОМАТИЗАЦИИ УПРАВЛЕНИЯ ТАРИФИКАЦИЕЙ ТРАНСПОРТА И ПРЕДОСТАВЛЕНИЯ ИНФОРМАЦИИ ПО ТРАНСПОРТНЫМ МАРШРУТАМ

AN ALGORITHM FOR FINDING THE OPTIMIZED AND SHORTEST PATH IN A SOFTWARE SYSTEM FOR AUTOMATION OF

TRANSPORT TARIFFIC MANAGEMENT AND PROVISION OF INFORMATION ON TRANSPORT ROUTES

Анотація. *Досліджено питання щодо розробки алгоритму для знаходження найкоротшого шляху метрополітеном до об'єктів.*

Ключові слова: *алгоритм, граф, сфера, вершини, ребра, центральний кут, відстань, шлях.*

Аннотация. *Исследованы теоретические вопросы о разработке алгоритма для нахождения кратчайшего пути метрополитеном к объектам.*

Ключевые слова: *алгоритм, граф, сфера, вершины, ребра, центральный угол, расстояние, путь.*

Summary. *Theoretical questions about the questions about the development of an algorithm for finding the shortest subway path to objects.*

Key words: *algorithm, graph, sphere, vertices, edges, central angle, distance, path.*

Довжина дуги великого кола – найкоротша відстань між будь-якими двома точками, що знаходяться на поверхні сфери, виміряна вздовж лінії, що з'єднує ці дві точки (така лінія носить назву ортодромії) і проходить по поверхні сфери або іншої поверхні обертання.

Через будь-які дві точки на поверхні сфери, якщо вони прямо протилежні один одному (тобто не є антиподами), можна провести унікальне велике коло. Дві точки розділяють велике коло на дві дуги. Довжина короткої дуги – найкоротша відстань між двома точками. Між двома точками-антиподами можна провести нескінченну кількість великих кіл, але відстань між ними буде однаково на будь-якому колі і дорівнює половині кола, або pi * R, де R - радіус сфери.

Форма Землі може бути описана як сфера, тому рівняння для обчислення відстаней на великому колі важливі для обчислення найкоротшої відстані між точками на Землі і часто використовуються в навігації.

Обчислення відстані цим методом більш ефективне і в багатьох випадках більш точне, ніж обчислення його для спроектованих координат (у прямокутних системах координат), оскільки, по-перше, для цього не треба переводити географічні координати у прямокутну систему координат (здійснювати проекційні перетворення) і, по-друге, багато проекцій, якщо неправильно обрані, можуть призвести до значних спотворень довжин через особливості проекційних спотворень.

Важливим рівнянням у навігації, яке дозволяє обчислити відстань між точками на сфері за їхніми координатами (довгота, широта) є формула Гаверсинуса [1].

Для будь-яких двох точок на сфері, гаверсинус центрального кута між ними обчислюється за наступною формулою:

$$haversin\left(\frac{d}{r}\right) = haversin\,(\varphi_2 - \varphi_1) + \cos(\varphi_1)\cos(\varphi_2)\,haversin(\lambda_2 - \lambda_1)$$

де

- d — відстань між двома точками на великому колі;
- r - радіус сфери;
- φ_1 и φ_2 — широта першої та другої точок в радіанах;
- λ_1 и λ_2 — довгота першої та другої точок в радіанах;
- haversin - функція гаверсинуса θ - центральний кут:

$$haversin(\theta) = sin^2\left(\frac{\theta}{2}\right) = \frac{1 - \cos(\theta)}{2}$$

На рис. 1 продемонстрована програмна реалізація методу для знаходження відстані від місцезнаходження користувача до найближчої станції метрополітену за допомогою формули Гаверсинуса:

```
187  from math import radians, cos, sin, asin, sqrt
188  def dist(lat1, long1, lat2, long2):
189      # перетворення десядичних градусів в радіани
190      lat1, long1, lat2, long2 = map(radians, [lat1, long1, lat2, long2])
191      # формула гаверсинуса
192      dlon = long2 - long1
193      dlat = lat2 - lat1
194      a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
195      c = 2 * asin(sqrt(a))
196      # радіус Землі у кілометрах: 6371
197      km = 6371* c
198      return km
```

Рис. 1. Функція для знаходження відстані між двома точками

Код методу написаний мовою Python і повертає відстань між точками у кілометрах. Спочатку відбувається перетворення десяткових градусів у радіани, після чого знаходимо відстань між широтами і довготами та використовуємо формулу Гаверсина. Щоб отримати результат у кілометрах, необхідно помножити с на радіус землі в кілометрах, а саме 6371 км.

Отримавши відстані між усіма необхідними об'єктами, знаходимо об'єкт з мінімальною:

```
def find_nearest(lat, long):
    distances = stations.apply(
        lambda row: dist(lat, long, row['lat'], row['lon']),
        axis=1)
    return stations.loc[distances.idxmin(), 'name']
```

Рис. 2. Функція для знаходження найближчої станції поряд

Для того, щоб знайти найкоротший шлях для пересування метрополітеном, лінії метро представимо у вигляді зваженого графа зі своїми вершинами та ребрами, де в якості ваги - відстань між станціями у кілометрах [2].

Алгоритм Форда-Беллмана дозволяє знайти найкоротші шляхи з однієї вершини графа до інших, навіть графів, у яких ваги ребер може бути негативними [3]. Тим не менш, у графі не повинно бути циклів негативної ваги, що досягаються з початкової вершини, інакше питання про

найкоротші шляхи є безглуздим. У цьому алгоритм Форда-Беллмана дозволяє визначити наявність циклів негативної ваги, досяжних з початкової вершини.

Алгоритм Форда-Беллмана використовує динамічне програмування [4]. Введемо функцію динамічного програмування:

F[k][i] - Довжина найкоротшого шляху з початкової вершини до вершини i, що містить не більше k ребер.

Початкові значення поставимо для випадку k = 0. У цьому випадку F[0][start] = 0, а для решти вершин і F[0][i] = INF, тобто шлях, що складається з нуля ребер існує тільки від вершини start до вершини start, а до інших вершин шляху з нуля ребер немає, що відзначатимемо значенням INF (∞).

Далі будемо обчислювати значення функції F збільшуючи число ребер у дорозі k, тобто обчислимо найкоротші шляхи, що містять не більше 1 ребра, найкоротші шляхи, що містять не більше 2 ребер і т. д. Якщо в графі немає циклів негативної ваги, то найкоротший шлях між будь-якими двома вершинами містить не більше ребра (- число вершин у графі), тому необхідно обчислити значення F[n-1][i], які будуть довжинами найкоротших шляхів від вершини start до вершини i.

Розглянемо, як обчислюється значення F[k][i]. Нехай є найкоротший маршрут з вершини start до вершини i, що містить не більше k ребер. Нехай останнім ребром цього маршруту є ребро j-i. Тоді шлях до вершини j містить не більше k-1 ребра і є найкоротшим шляхом із усіх таких шляхів, отже, його довжина дорівнює F[k-1][j], а довжина шляху до вершини i дорівнює F[k-1][j] + W[j][i], де W[j][i] є вага ребра j-i. Далі необхідно перебрати всі вершини j, які можуть виступати як попередні, і вибрати мінімальне значення F[k-1][j] + W[j][i].

Розглянемо реалізацію алгоритму Форда-Беллмана з допомогою списків суміжності мовою Python.

Нехай граф заданий списками суміжності, а вага ребра j-i зберігається у словнику W[j, i], де ключ це кортеж з j, i, а значення — вага ребра. Тоді перебрати всі ребра графа, можна організувавши цикл за всіма ключами словника W.

Тобто, по суті, алгоритм Форда-Беллмана можна сформулювати так:

1) Проініціалізувати масив F значеннями F[start] = 0, F[i] = INF для інших i.

2) Пройтися по всіх ребрах j-i графа, намагаючись зрелаксувати ребро j-i.

3) Пункт 2 повторити n-1 раз.

Для відновлення цього шляху необхідно запам'ятовувати предка кожної вершини, оновлюючи його при успішній релаксації ребра. Алгоритм можна зупинити, якщо на жодному кроці жодне ребро не буде зрелаксовано.

Отже, для знаходження оптимального та найкоротшого шляху метрополітеном, було досліджено питання щодо розробки алгоритму знаходження оптимального шляху на основі формули Гаверсинуса та алгоритму Беллмана-Форда. Наявність такого алгоритма забезпечить пошук найближчої до людини станції метрополітену, а також найкоротшого шляху до станції біля необхідного об'єкту маючи лише координати місцезнаходження людини та кінцевий об'єкт.

Література

1. Glen Van Brummelen. Heavenly Mathematics: The Forgotten Art of Spherical Trigonometry. 2013. P. 191.
2. A Gentle Introduction To Graph Theory // basecs: [Веб-сайт]. URL: https://medium.com/basecs/a-gentle-introduction-to-graph-theory-77969829ead8 (дата звернення: 20.04.2022).

3. Bellman-Ford algorithm // Wikipedia: [Веб-сайт] https://en.wikipedia.org/wiki/Bellman%E2%80%93Ford_algorithm (дата звернення: 21.04.2022).
4. Рафгарден Т. Досконалий алгоритм. Жадібні алгоритми і динамічне програмування / Тім Рафгарден. 2020. 256 с. (Пітер Прес).