Міністерство освіти та науки України

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра Інформаційних Систем та Технологій

Теорія алгоритмів

Індивідуальне завдання

Виконав: студент 1 курсу

Білогуб К.С

Перевірила пос.

[Солдатова М. О.](http://epi.kpi.ua/Schedules/ViewSchedule.aspx?v=f1ad598f-3968-4de6-98a7-cfced073066c)

Київ – 2023

Індивідуальне завдання

1. Постановка задачі:

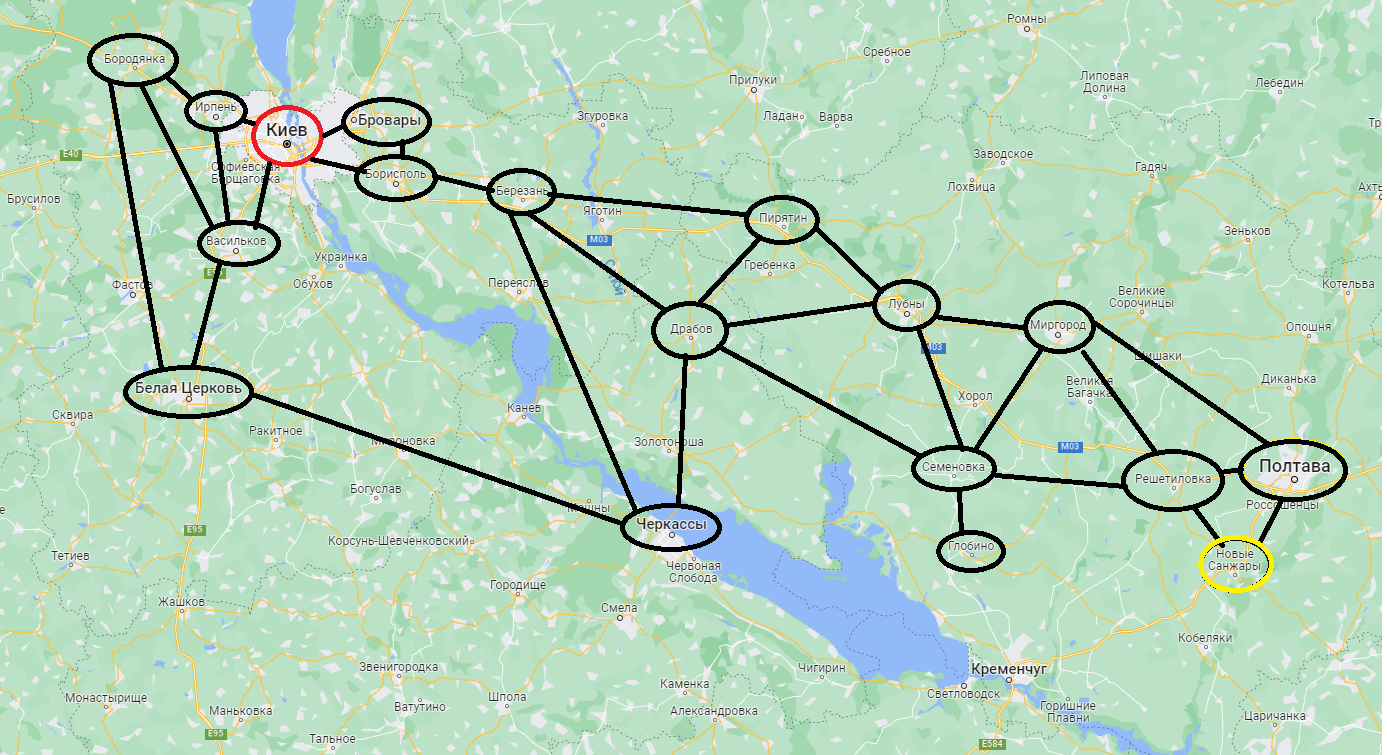
На основі карти України обрати частину території, яка включає в себе населений пункт проживання студента, населений пункт його рідних (не співпадає з попереднім) та ще декільканаселених пунктів навколо них. На основі виділеної території створити базовий граф, вершиниякого – населені пункти, ребра – дороги між ними. Кількість вершин повинна бути від 15 до 20, кількість ребер від 30 . Вагами ребер необхідно взяти:

a) довжину шляху між населеними пунктами в км,

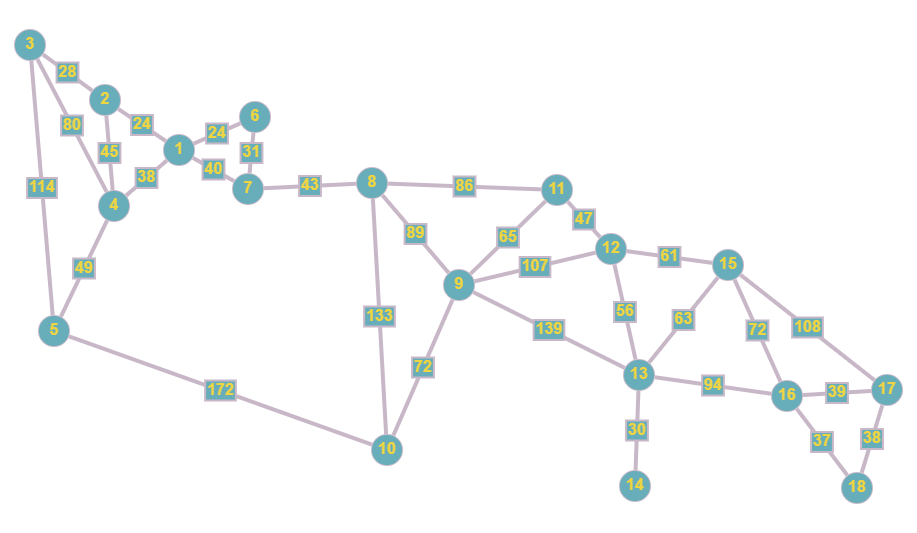
b) тип дороги, ставлячи йому відповідний пріоритет.

Для кожного населеного пункту визначити його координати та кількість населення.

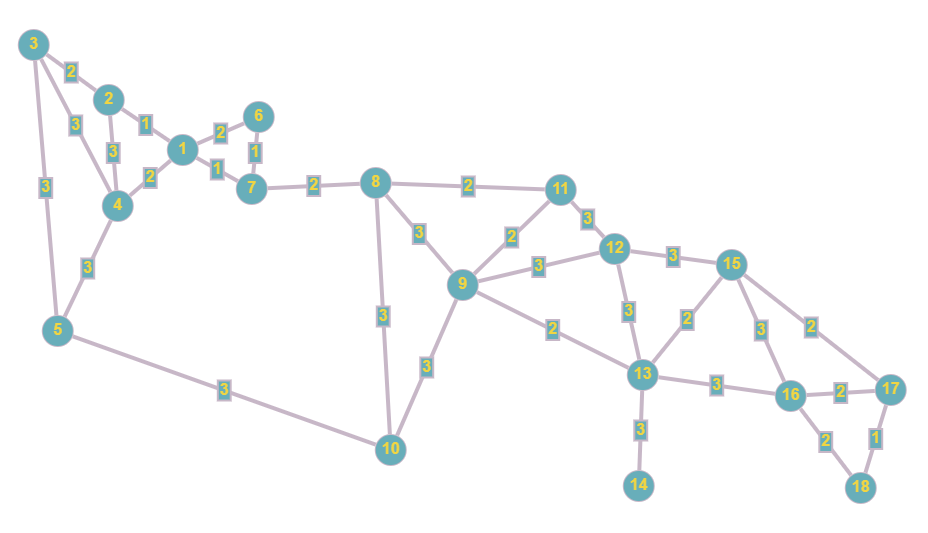
Демонстрація карти, на основі якої було побудовано граф, червоне коло – населений пункт проживання студента, жовте коло – населений пункт проживання його рідних



Базовий граф, в якому за ваги ребер взята довжина шляху між населеними пунктами в км:



Базовий граф, в якому за ваги ребер взят тип дороги (чим менше цифра, тим більша пріоритетність):



1 – Нове шосе

2 – Дорога регіонального значення

3 - Пересічена місцевість

Таблиця з показниками населених пунктів:



2. Завдання:

На основі базового графу необхідно розв’язати наступні завдання;

1. За допомогою одного з алгоритмів обходу графів обійти всі населені пункти на виділеній територій. Початком вважати населений пункт проживання студента, кінцевою точкою – пункт проживання його рідних. На довжину шляху та тип дороги не зважати.

Вибір алгоритму:

Для обходу всіх вершин графу може використовуватися BFS або DFS алгоритм. Обидва алгоритми мають складність O(V+E), де V - кількість вершин, а E - кількість ребер для найгіршого випадку, тому для цього завдання вибір конкретного з цих двох не є принциповим. Для цього завдання був обраний BFS алгоритм, який в реалізації не використовує рекурсію.

Опис алгоритму:

BFS, або пошук в ширину (Breadth-First Search), - це алгоритм обходу або пошуку в графі. Його основна ідея полягає в тому, що він спочатку відвідує всі вершини на одному рівні, перш ніж перейти до наступного рівня. Це робить BFS особливо корисним для знаходження найкоротшого шляху в невагомих графах або для знаходження всіх вершин, які можна досягти з початкової вершини.

Кілька основних характеристик BFS:

Охоплення всього графа: BFS відвідує всі вершини графа, якщо вони досяжні від початкової вершини.

Знаходження найкоротшого шляху: BFS може використовуватися для знаходження найкоротшого шляху в невагомих графах, оскільки він спочатку відвідує всі вершини на одному рівні, перш ніж перейти до наступного рівня.

Використання черги: BFS використовує структуру даних "черга" для відслідковування вершин для відвідування.

Пошук в ширину (BFS) працює наступним чином:

Ініціалізація: BFS починає свою роботу з початкової вершини (root node). Він створює структуру даних "черга" та додає початкову вершину до цієї черги.

Відвідування вершини: BFS видаляє вершину з початку черги та "відвідує" її. "Відвідування" вершини означає, що алгоритм виконує певну дію з цією вершиною, наприклад, друкує її значення або перевіряє, чи є вона цільовою вершиною.

Додавання сусідів до черги: Після відвідування вершини, BFS додає всі її невідвідані сусіди до кінця черги.

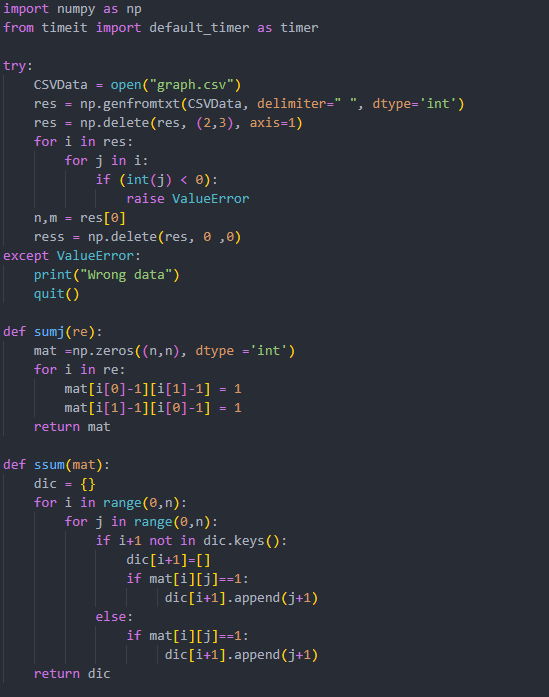
Повторення: Алгоритм продовжує цикл видалення вершини з початку черги, відвідування цієї вершини та додавання її невідвіданих сусідів до кінця черги, поки черга не стане порожньою.

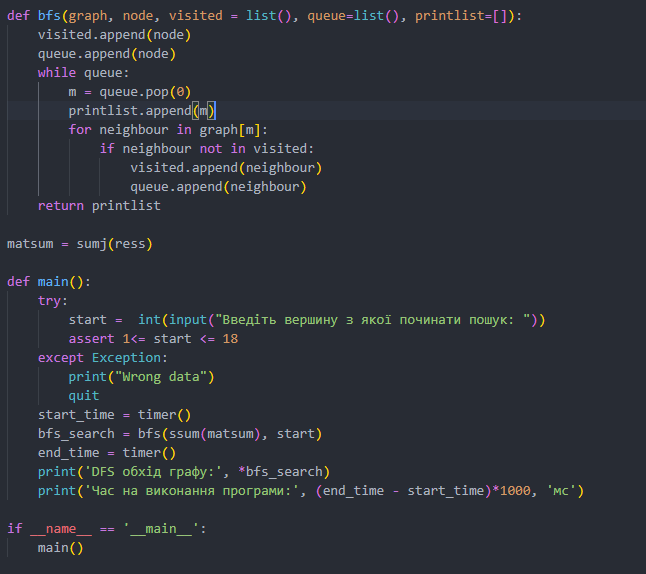
Кінець: Коли черга стає порожньою, це означає, що BFS відвідав всі вершини, які можна досягнути від початкової вершини, і алгоритм завершує свою роботу.

Особливості:

Важливо пам'ятати, що BFS відвідує вершини в порядку їх додавання до черги, тобто він спочатку відвідує всі вершини на одному рівні (вершини, які можна досягнути за один крок від початкової вершини), перш ніж перейти до вершин наступного рівня (вершини, які можна досягнути за два кроки від початкової вершини).

Реалізація BFS мовою Python 3.11.3:







**Таблиця тестування**

|  |  |  |  |
| --- | --- | --- | --- |
| Вхідні дані | | Результат | Призначення тесту |
| start | Graph.csv |
| 1 | 18 30 0 0  1 2 1 24  1 4 2 38  1 7 1 40  …….. | Відомий | Перевірка правильності роботи алгоритму та результату його роботи |
| 1<=Випадкове<=18 | Випадкове  Int, int, int, int | Невідомий, але легко обчислити | Перевірка правильності роботи алгоритму та результату його роботи |
| Випадкове<=1, Випадкове>=18 | - | Відомий | Реакція на невалідні дані |
| - | - | Відомий | Реакція на невалідні дані |
| 5 | 18 30 0 0  1 2 1hgh 24  1 4ds 2 38  1 7 1 40  ……… | Відомий | Реакція на невалідні дані |

1. За допомогою одного з алгоритмів знайти шлях від населеного пункту проживання студента до пункту проживання його рідних. На довжину шляху та тип дороги не зважати.

Вибір алгоритму:

Для цього завдання буде краще використати алгоритм DFS, бо він працює, спочатку відвідуючи якомога глибше від початкової вершини, перш ніж повертатися назад, тобто краще підходить для ситуацій, коли вершина, до якої треба знайти шлях знаходиться далеко від початкової (наш випадок). Як було зазначено вище, він має таку ж складність як і BFS: O(V+E), де V - кількість вершин, а E - кількість ребер для найгіршого випадку. При реалізації була використана рекурсія.

Опис алгоритму:

DFS, або пошук в глибину (Depth-First Search), - це алгоритм обходу або пошуку в графі, який працює, спочатку відвідуючи якомога глибше від початкової вершини, перш ніж повертатися назад.

Основні характеристики DFS:

Охоплення всього графа: як і BFS, DFS відвідує всі вершини графа, якщо вони досяжні від початкової вершини.

Використання стеку: DFS використовує структуру даних "стек" для відслідковування вершин для відвідування.

Відвідування глибоких вершин спочатку: відмінність між DFS і BFS полягає в тому, що DFS спочатку відвідує глибокі вершини (тобто вершини, які знаходяться далеко від початкової вершини).

Пошук в глибину (DFS) працює наступним чином:

Ініціалізація: DFS починає свою роботу з початкової вершини (root node). Він створює структуру даних "стек" та додає початкову вершину до цього стеку.

Відвідування вершини: DFS видаляє вершину з верхівки стеку та "відвідує" її. "Відвідування" вершини означає, що алгоритм виконує певну дію з цією вершиною, наприклад, друкує її значення або перевіряє, чи є вона цільовою вершини.

Додавання сусідів до стеку: Після відвідування вершини, DFS додає всі її невідвідані сусіди до верхівки стеку.

Повторення: Алгоритм продовжує цикл видалення вершини з верхівки стеку, відвідування цієї вершини та додавання її невідвіданих сусідів до верхівки стеку, поки стек не стане порожнім.

Кінець: Коли стек стає порожнім, це означає, що DFS відвідав всі вершини, які можна досягнути від початкової вершини, і алгоритм завершує свою роботу.

Важливо зазначити, що DFS відвідує вершини в порядку їх додавання до стеку, тобто він спочатку відвідує "глибокі" вершини (вершини, які знаходяться далеко від початкової вершини), перш ніж повертатися до менш глибоких вершин.

Реалізація DFS мовою Python 3.11.3: