# **Метод гілок та меж для розв’язання задачі комівояжера**

Підготували студенти групи ПМІм-13:  
Кобіневич Віталій,

Бутрий Іван,

Карабін Юрій,

Попель Іван,

Єршова Анна

# **Вступ**

Задача комівояжера (Traveling Salesman Problem, TSP) є однією з найвідоміших і найвивченіших задач комбінаторної оптимізації. У своїй основі вона ставить запитання: як комівояжер може обійти певний набір міст, відвідавши кожне з них рівно один раз, та повернутися до початкового міста, при цьому мінімізуючи загальні витрати (наприклад, відстань або час).

Одним з найбільш ефективних підходів до розв'язання TSP є метод гілок та меж (branch and bound), який дозволяє знаходити оптимальні рішення шляхом систематичного перебору можливих маршрутів з використанням обчислень меж для скорочення пошукового простору. У цьому рефераті ми розглянемо історію методу, його концепції та принципи роботи, його переваги та недоліки, а також застосування в контексті задачі комівояжера.

## **Постановка задачі комівояжера**

Задача комівояжера формулюється таким чином:

* Є набір міст, кожне з яких повинно бути відвідане лише один раз.
* Комівояжер повинен почати і завершити маршрут у певному місті (наприклад, у місті A).
* Відомі витрати або відстані для кожної пари міст, що визначають "ціну" переміщення між містами.
* Необхідно знайти найкоротший можливий маршрут, який дозволить комівояжеру відвідати кожне місто один раз і повернутися в початкову точку.

Математично задача комівояжера може бути представлена у вигляді зваженого графа, де кожна вершина (місто) з'єднана з іншими вершинами ребрами, що мають певну вагу (вартість переходу між містами).

Початкова умова може задаватись також як матриця суміжностей, де значення - відстані між містами чи витрати на переміщення, тобто "ціна".

## 

## **Загальні концепції методу гілок та меж**

Метод гілок та меж є систематичним підходом до пошуку оптимальних рішень, який включає в себе три основні компоненти: гілки, межі та відсічення. Розглянемо їх детальніше.

### **1. Гілки (Branching)**

Гілки представляють процес розбиття задачі на менші підзадачі. У контексті задачі комівояжера, на початку алгоритму вибирається стартове місто, з якого комівояжер розпочинає свій маршрут. Це місто вважається коренем дерева рішень. У цей момент всі шляхи ще відкриті для розгляду, оскільки жодне інше місто ще не відвідано. На кожному кроці алгоритм обирає нове місто для відвідування, формуючи відповідну гілку. Наприклад, на початку комівояжер може вибрати будь-яке місто як наступну точку призначення. З кожним кроком число доступних міст скорочується, і формуються нові підзадачі, що містять інформацію про всі вже відвідані та невідвідані міста.

Кожне нове місто, вибране для відвідування, розгалужує пошук на кілька можливих шляхів, формуючи дерево рішень. Процес розгалуження продовжується доти, поки комівояжер не відвідає всі міста. У результаті, дерево рішень містить усі можливі маршрути, які може пройти комівояжер.

### **2. Межі (Bounding)**

Для кожної з підзадач, що виникають у процесі гілок, розраховується нижня межа витрат (наприклад, відстані). Ця межа відображає мінімальні можливі витрати для даної підзадачі, виходячи з відомих даних про вже відвідані міста та залишені для відвідування.

Для обчислення меж можна використовувати різні методи, такі як:

* **Евристичні оцінки**: вони можуть дати швидку оцінку витрат, не проходячи через усі можливі маршрути.
* **Метод динамічного програмування**: він дозволяє зберігати результати обчислень для вже розглянутих підзадач і використовувати їх повторно.

Якщо обчислена межа для певної підзадачі перевищує вже знайдене оптимальне рішення, ця гілка (підзадача) може бути відсічена, оскільки немає сенсу продовжувати її дослідження.

### **3. Відсічення (Pruning)**

Відсічення є критично важливим етапом методу гілок та меж, оскільки воно дозволяє зменшити кількість обчислень шляхом відкидання неперспективних маршрутів. Якщо межа підзадачі вказує на те, що вона не може дати кращого результату, ніж уже знайдене оптимальне рішення, вона відсікається з подальшого розгляду.

Процес відсічення значно знижує обсяг даних, які потрібно проаналізувати, і допомагає зосередитися на найбільш перспективних варіантах рішень. У випадку задачі комівояжера, це може суттєво зменшити кількість маршрутів, які потрібно перевірити, що, в свою чергу, скорочує час на розв'язання.

### **4. Пошук оптимального рішення**

Метод гілок та меж продовжує своє функціонування до тих пір, поки не буде знайдено оптимальне рішення. Алгоритм зберігає найкраще знайдене рішення та порівнює його з межами нових підзадач.

Коли всі можливі гілки досліджені, а відсічені неефективні маршрути, алгоритм зупиняється, і знайдене оптимальне рішення оголошується остаточним.

Розв’язком вважається отримана впорядкована послідовність вузлів, що представляють міста.

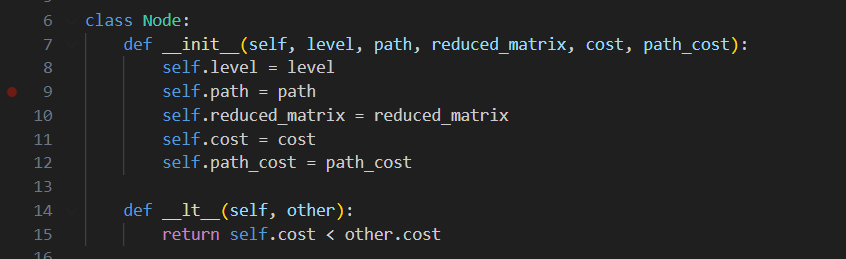
Загалом послідовність виконання алгоритму виглядає наступним чином:

**Звіт про Розв’язання Задачі Комівояжера**

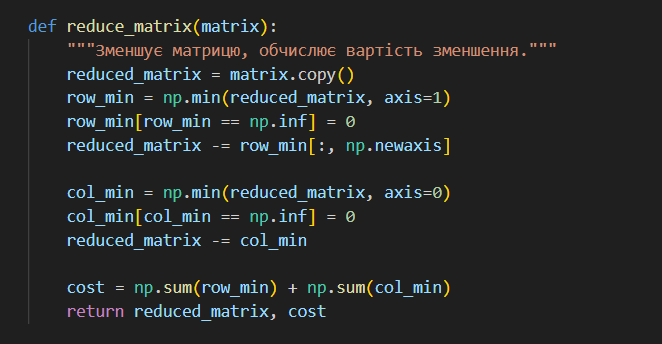
У цьому проєкті використовується алгоритм гілок та меж для вирішення задачі комівояжера — класичної проблеми в комбінаторній оптимізації, яка має важливі застосування в реальному світі. Від оптимізації маршрутів доставки до планування логістики, задача комівояжера допомагає ефективно вирішувати питання, пов’язані з мінімізацією часу або витрат. Метою цього проєкту є знайти найкоротший маршрут, який проходить через всі міста і повертається в початкову точку, мінімізуючи загальні витрати.

**Робота Програми**

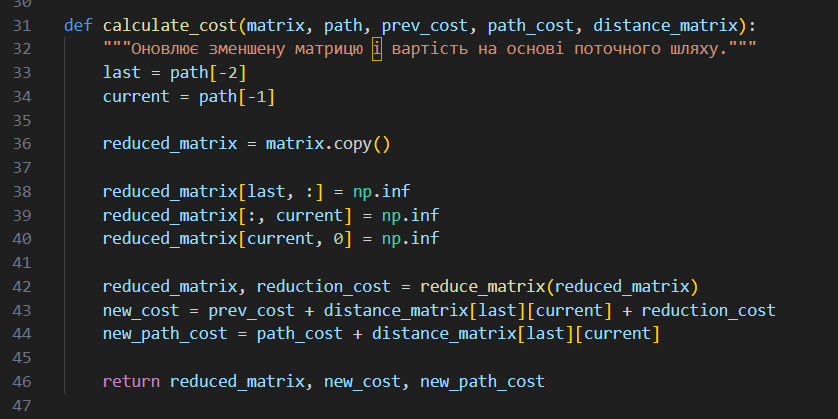
Програма розв'язує задачу комівояжера, застосовуючи метод гілок та меж, який забезпечує ефективний пошук оптимального маршруту. Основу її роботи складають кілька ключових компонентів. Почнемо з класу Node, що представляє вузол дерева пошуку. Кожен вузол зберігає інформацію про рівень, поточний шлях, зменшену матрицю витрат, загальну оцінку вартості маршруту та суму фактичних витрат на пройдений шлях. Вузли порівнюються між собою за вартістю, щоб алгоритм міг обирати для обробки ті, що мають найменшу вартість, і досліджувати їх у першу чергу.



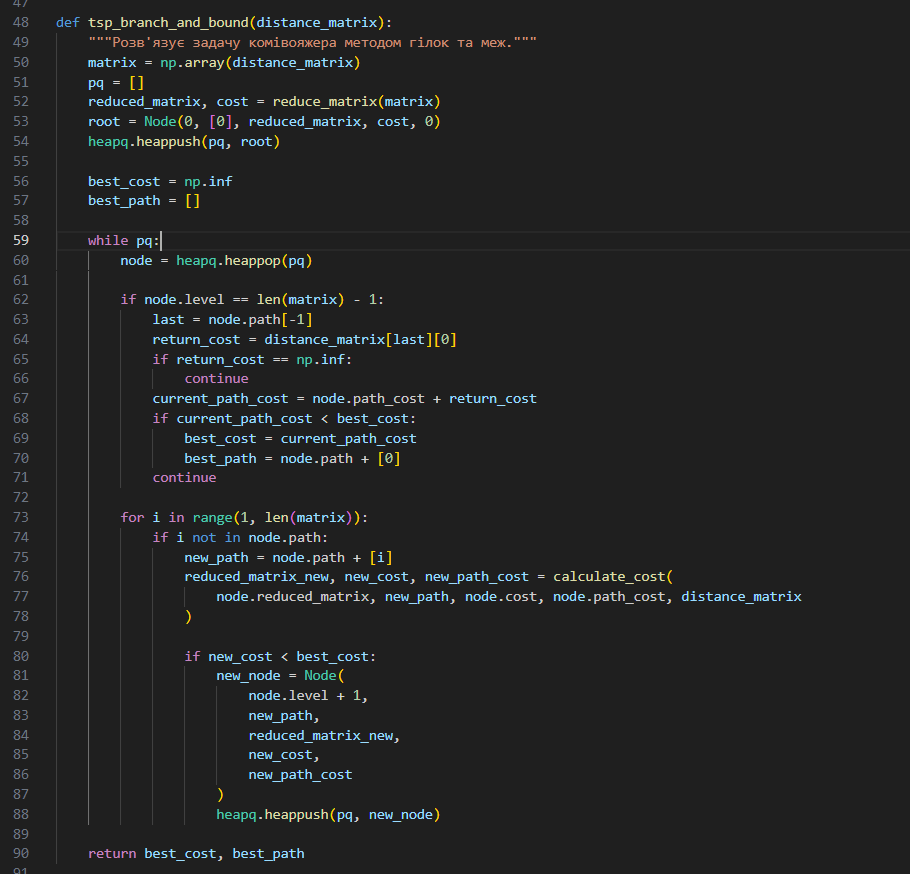
Одним із ключових процесів є зменшення матриці витрат, яке виконується у функції reduce\_matrix. Алгоритм знаходить мінімальні значення в кожному рядку та стовпці матриці й віднімає ці значення з усіх елементів відповідного рядка або стовпця. Це зменшення знижує загальну оцінку шляху, створюючи базову вартість, з якої починається розрахунок. Завдяки цьому процесу ми можемо відсікати маршрути з явно вищими витратами, навіть не розглядаючи їх детально. Це значно знижує кількість варіантів, які потрібно перевірити, і підвищує ефективність алгоритму.



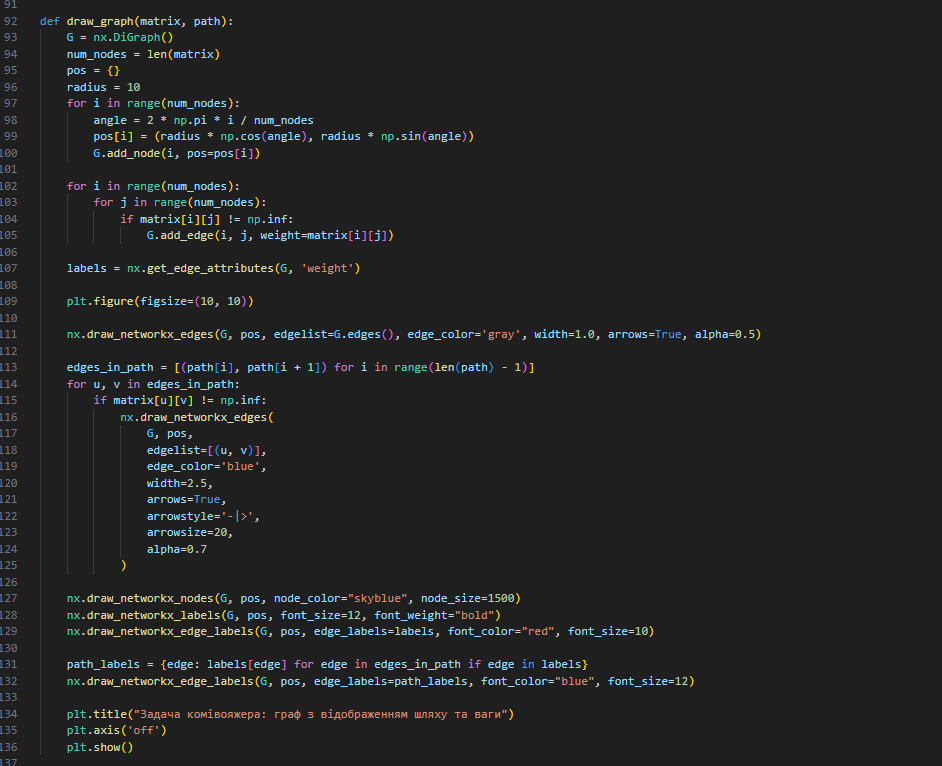
Функція calculate\_cost відповідає за оновлення зменшеної матриці витрат. Вона модифікує матрицю, блокуючи вже відвідані міста та додаючи нові витрати на перехід до наступного міста. Потім обчислюється зменшена матриця та нова загальна вартість шляху, яка враховує всі додані витрати. Це гарантує, що алгоритм не повторює маршрути та ефективно мінімізує витрати на кожному кроці.

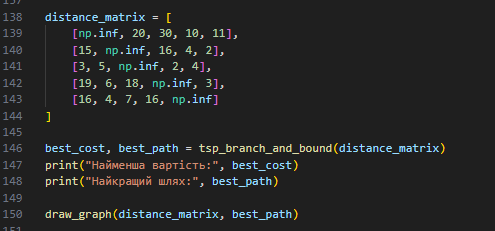


Головна функція tsp\_branch\_and\_bound реалізує метод гілок та меж. Алгоритм починає з кореневого вузла, де матриця витрат вже зменшена, і додає цей вузол у пріоритетну чергу. У циклі вузли обробляються за порядком з найменшою вартістю, і для кожного можливого нового міста створюються нові вузли, які додаються до черги, якщо їхня вартість перспективна. Якщо вузол досягає рівня, коли залишається відвідати лише одне місто, алгоритм обчислює повну вартість маршруту, включаючи повернення у початкове місто. Якщо ця вартість менша за найкращу знайдену, маршрут зберігається як оптимальний.



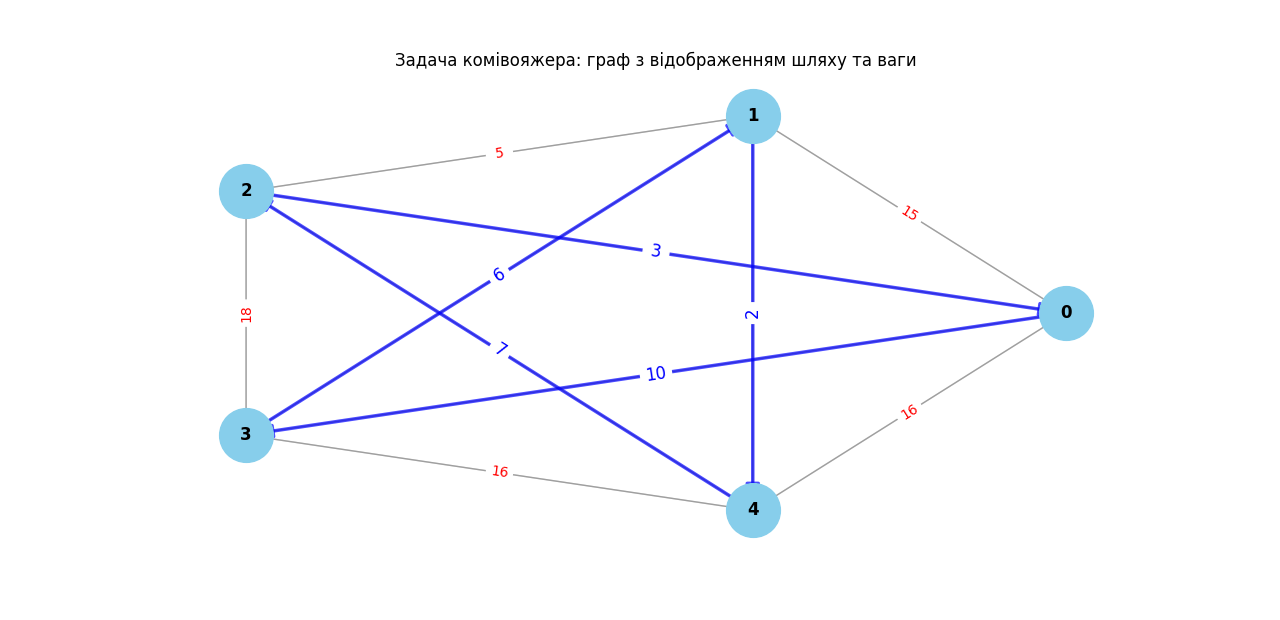
Візуалізація результату здійснюється у функції draw\_graph. Вона створює граф із вершинами, що представляють міста, і орієнтованими ребрами, які відображають можливі маршрути з відповідними вагами. Сині лінії виділяють оптимальний маршрут, показуючи найкращий варіант серед усіх можливих, тоді як інші шляхи зображені сірим кольором для контрасту. Відображення ваг ребер допомагає краще зрозуміти структуру графа і підтвердити правильність обчислень.



****

**Результати**

Програма успішно знайшла найкоротший маршрут, що дозволяє відвідати всі міста і повернутися до початкової точки з мінімальними витратами. Після виконання обчислень було визначено, що найменша вартість маршруту становить **28**, а оптимальний шлях має вигляд: **[0, 3, 1, 4, 2, 0]**. Це означає, що маршрут починається в місті 0, проходить через міста 3, 1, 4, 2 і завершується поверненням у місто 0.



Граф показує всі можливі маршрути між містами, де оптимальний шлях виділений синіми лініями. Червоні числа позначають ваги ребер, а сині ваги дублюють значення для оптимального шляху.

**Висновок**

Метод гілок та меж виявився ефективним для вирішення задачі комівояжера, оскільки дозволяє значно знизити кількість перевірених варіантів шляхом відсіювання нерентабельних маршрутів. Це забезпечує оптимальний результат у розумний час для задач середнього розміру, таких як наша. Програма не тільки обчислила оптимальний маршрут, але й дозволила візуалізувати його для кращого розуміння.