Практична робота №8. Жадібні алгоритми. Наближене розв’язання екстремальних задач

2025.11.23,м.Кременьчук	Створив: Огоновський О.Є.

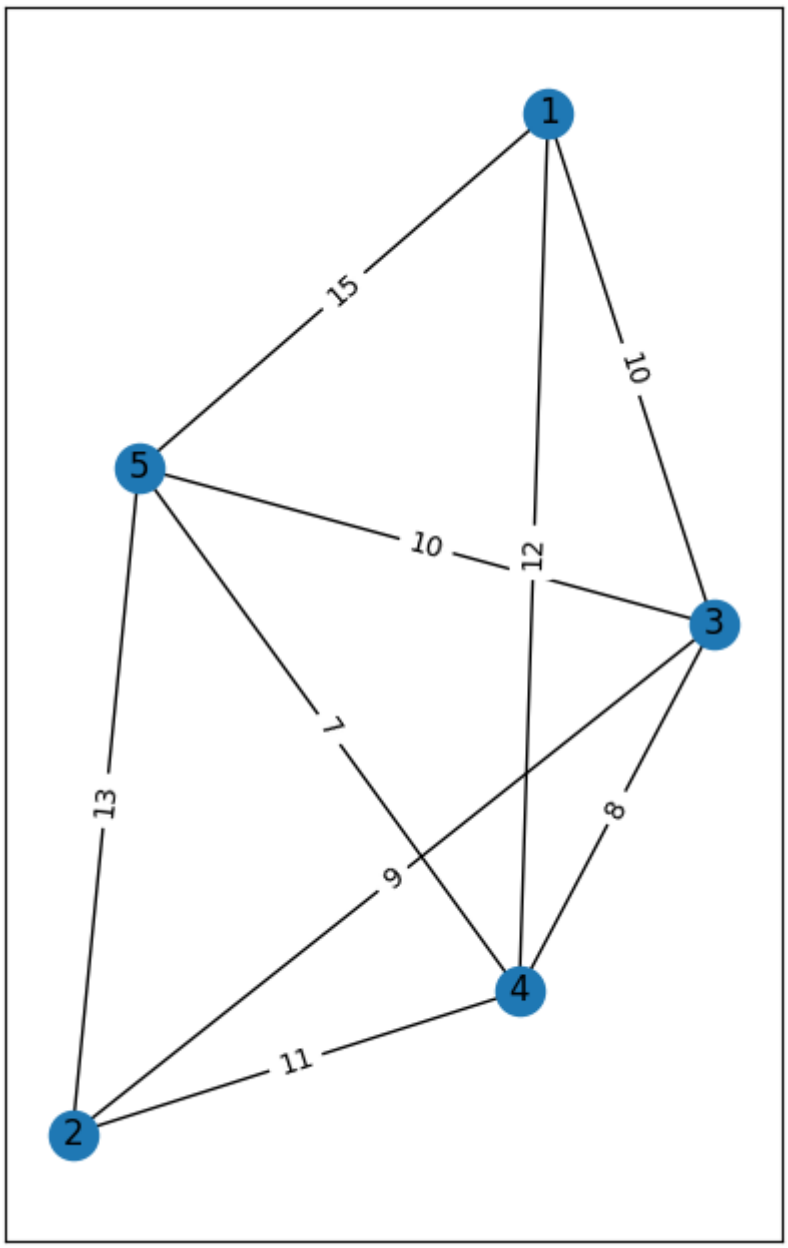
Мета: набути практичних навичок застосування деяких жадібних алгоритмів для розв’язання екстремальних задач.

Задача для самостійного розв’язання

Розв’язати задачу комівояжера для графа, заданого варіантом, використовуючи код, наведений вище.

```
import matplotlib.pyplot as plt
import networkx as nx

plt.figure(figsize=(5,8))


G = nx.Graph()
G.add_nodes_from(range(1,6))
G.add_weighted_edges_from([(1,3,10), (1,4,12), (1,5,15), (2,3,9), (2,4,11), (2,5,13),
(3,4,8), (3,5,10), (4,5,7)])

pos = nx.spring_layout(G)

nx.draw_networkx_nodes(G,pos)
nx.draw_networkx_labels(G,pos)
edges = [(u, v) for (u, v, d) in G.edges(data=True)]
nx.draw_networkx_edges(G, pos, edgelist=edges)
edge_labels = dict([((u, v), d['weight']) for u, v, d in G.edges(data=True)])
nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels)

plt.show()
```

Візуалізувати граф



Обґрунтувати асимптотику для алгоритмів

1. **Алгоритм Грубої сили (Brute Force)**

Суть:

- Перебираються всі можливі перестановки вершин (шляхів), які починаються з фіксованої початкової вершини.
- Обчислюється довжина кожного з них.
- Вибирається найкоротший.

Кількість можливих маршрутів:

- Для n вершин (з фіксованим стартом) існує $(n-1)!$ маршрутів.

Вартість обчислення довжини одного маршруту:

- Потрібно пройтися по n вершинах – O(n) Повна складність: O(n·(n−1)!)=O(n!) Висновок:
- Алгоритм дуже повільний, але дає точний розв’язок.
- Підходить лише для малих графів (n ≤ 10).

2. **Алгоритм Найближчого сусіда (Nearest Neighbor)**

Суть:

- Починаючи з деякої вершини, на кожному кроці вибирається найближча непрохіджена вершина.
- Повторюється, поки не буде відвідано всі вершини.
- Потім повертається у початкову.

Кількість дій:

- У першій ітерації — перевіряється n−1 сусідів,
- У другій — n−2,
- Останній крок — 1 сусід. Сума: (n−1)+(n−2)+⋯+1= $n(n-1)$

Повна складність: $O(n^2)$ Висновок:

- Набагато швидший, ніж Brute Force.
- Дає неточний (наближений) результат, але часто хороший на практиці.
- Підходить для великих графів, де потрібна швидкість

Порівняльна таблиця:

Алгоритм	Точність	Асимптотична складність	Підходить для
Грубої сили (Brute Force)	Висока	O(n!)	Малих графів
Найближчого сусіда	Середня	$O(n^2)$	Великих графів

Відповіді на контрольні питання

1. **Що таке жадібний алгоритм?**

Жадібний алгоритм — це метод розв’язання задачі, який на кожному кроці приймає локально оптимальне рішення, сподіваючись, що в результаті отримає глобально

оптимальний результат.

2. **Які головні принципи роботи жадібних алгоритмів?**

- На кожному кроці вибирається найкраще (найоптимальніше) локальне рішення.
- Прийняте рішення не змінюється у подальших кроках (відсутність повернень).
- Проблема має властивість «жадібної вибірки» — локальний вибір веде до глобального оптимуму.

3. **Яка головна відмінність між жадібними алгоритмами та динамічним програмуванням?**

- Жадібні алгоритми приймають рішення один раз і не повертаються назад.
- Динамічне програмування розглядає всі можливі підзадачі і запам’ятовує їх рішення, що дозволяє знаходити глобально оптимальний результат навіть при відсутності властивості жадібної вибірки.

4. **Наведіть приклади задач, які можна розв’язати за допомогою жадібнихалгоритмів.**

- алгоритм Дейкстри знаходження найкоротшого шляху від однієї вершини до всіх інших у зваженому ациклічному графі
- алгоритм Краскала, який знаходить мінімальне кістякове дерево у зваженому графі
- алгоритм оптимального кодування Гафмена
- задача про рюкзак.

5. **Які можуть бути обмеження у використанні жадібних алгоритмів для розв'язання екстремальних задач?**

- Жадібні алгоритми не гарантують глобального оптимуму для усіх задач.
- Вони можуть працювати некоректно, якщо задача не має властивості жадібного вибору.
- Підходять лише для задач, де локальний оптимум веде до глобального.

6. **Чому жадібні алгоритми часто використовуються для наближеного розв’язання екстремальних задач?**

- Вони прості і швидкі у реалізації.
- Дають хороші приблизні рішення, якщо точний розв’язок занадто складний або неможливий за прийнятний час.
- Дозволяють отримати результат з прийнятною точністю у великих або складних задачах.