2. Russell S. J., Norvig P. Artificial Intelligence: A Modern Approach. Pearson Education, Limited, 2021.
3. Twitter API Documentation. https://developer.twitter.com/en/docs.
Abstract. *The report discusses the use of the Python programming language and social media APIs for analysing agricultural markets and forecasting trends. The methods of collecting, processing and analysing data from Twitter, Facebook and Instagram are described. Algorithms for text cleaning, sentiment analysis, and building machine learning models to predict changes in demand for agricultural products are presented. The study shows that social media is a valuable source of information for the agricultural sector, and modern technologies can improve the efficiency of market decisions.*
Keywords: *agricultural market analysis, Python, social media APIs, machine learning, demand forecasting.*

Науковий керівник:
Пархоменко О. Ю.,
к. ф.-м. н., доцент кафедри економічної кібернетики, комп'ютерних наук та інформаційних технологій, Миколаївський національний аграрний університет

УДК 004.94:656.073.2:631.11

Оптимізація логістики транспортування врожаю за допомогою алгоритмів графів у Python (NETWORKX)

Віолетта Кім,
здобувачка вищої освіти спеціальності 122 Комп'ютерні науки
Миколаївський національний аграрний університет
м. Миколаїв, Україна

Анотація: *досліджено використання алгоритмів графів для оптимізації логістики транспортування врожаю. Розглянуто алгоритми пошуку найкоротших шляхів, мінімального остовного дерева та їх реалізацію в Python за допомогою бібліотеки NetworkX*
Ключові слова: *оптимізація логістики, транспортування врожаю, алгоритми графів, Python, NetworkX, найкоротший шлях, транспортна мережа, аграрний сектор.*

Оптимізація логістики транспортування є дуже важливою для сільського господарства. Вона допомагає зменшити витрати, скоротити час доставки та економно використовувати транспортні ресурси. Для розв'язання таких задач використовуються алгоритми Дейкстри, Флойда-Варшалла, Прима, Крускала і Беллмана-Форда.

Одним з найпоширеніших і найважливіших є алгоритм Дейкстри, що використовується для знаходження найкоротшого шляху від однієї точки до всіх інших у графі з невід'ємними вагами. Цей алгоритм допоможе визначити найкращий маршрут для транспортного засобу та зменшити витрати на перевезення.

Ще одним важливим є алгоритм Флойда-Варшалла, який дозволяє знаходити найкоротші шляхи між всіма парами вершин. Це важливо коли потрібно вдосконалити маршрути між кількома точками збору врожаю та місцями його зберігання.

Для задачі пошуку кістякового дерева використовують алгоритми Прима чи Крускала. Вони допомагають побудувати вигіднішу мережу транспортних маршрутів. Це дуже корисно при розгляданні нових маршрутів.

При роботі з графами найчастіше використовують мову програмування Python через її простоту та великий вибір різноманітних бібліотек. Однією з найкращих бібліотек для роботи з графами є NetworkX. Вона має зручні інструменти для роботи з графами, що включають в себе аналіз структури мережі, пошук найкоротшого транспортного шляху та багато інших можливостей. Бібліотека NetworkX може працювати з будь-якими графами, зокрема, неорієнтованими орієнтованими, ваговими та багатографами. Вершини виступають як точки збору або доставки врожаю, ребра – транспортні маршрути з параметрами, такими як відстань, витрати чи час у дорозі. Завдяки функції збереження вершин і ребер, бібліотека може створити реальні транспортні мережі ураховуючи додаткові параметри.

```
import networkx as nx
    #Створення графа
G=nx.Graph()
    #Додавання вершин
G.add_nodes_from(["Поле1", "Поле2", "Склад", "Переробка"])
    #Додавання ребер з вагами
G.add_edge("Поле1", "Склад", weight=10)
G.add_edge("Поле2", "Склад", weight=15)
G.add_edge("Склад", "Переробка", weight=20)
```

Також бібліотека може підтримувати візуалізацію графів, що допоможе краще оцінити транспортні маршрути:

```
import matplotlib.pyplot as plt
    #Візуалізація графа
pos=nx.spring_layout(G)
nx.draw(G, pos, with_labels=True, node_color='lightblue', edge_color='gray')
nx.draw_networkx_edge_labels(G, pos, edge_labels={(u, v): d['weight'] for u, v, d in G.edges(data=True)})
plt.show()
```

В Python flгоритм Дейкстри, для знаходження найкоротшого шляху, реалізується за допомогою функції nx.dijkstra_path() або nx.single_source_dijkstra, що допоможе знайти оптимальний маршрут.

```
# Знаходження найкоротшого шляху від Поля1 до Переробки
shortest_path = nx.dijkstra_path(G, source="Поле1", target="Переробка", weight="weight")
print("Найкоротший шлях:", shortest_path)
```

Цей код створює граф, де вершини представляють точки транспортування, а ребра – маршрути між ними з відповідними вагами.

Алгоритм Флойда-Варшалла для знаходження найкоротших шляхів між усіма парами вершин у графі, використовують у випадках коли потрібно розглянути всі можливі транспортні маршрути та знайти найкращі варіанти. Основна ідея алгоритму полягає в покращенні відстані між вершинам, процесом перевірки, чи існує коротший шлях через проміжну вершину. Це допомагає обробити одночасно всі вершини, що корисно при складних логістичних задачах.

Сам алгоритм Флойда-Варшалла реалізується так:

```
import networkx as nx
# Створення орієнтованого графа
G = nx.DiGraph()
# Додавання вершин та ребер з вагами
edges = [("Поле1", "Склад", 10), ("Поле2", "Склад", 15), ("Склад", "Переробка", 20), ("Поле1", "Поле2", 25)]
G.add_weighted_edges_from(edges)
# Знаходження найкоротших шляхів між всіма парами вершин
shortest_paths = dict(nx.floyd_warshall(G))
# Вивід матриці найкоротших шляхів
for source, targets in shortest_paths.items():
  for target, distance in targets.items():
    print(f"Найкоротший шлях між {source} і {target}: {distance}")
```

Алгоритми Прима та Крускала використовується для пошуку мінімального остовного дерева. В логістиці це допомагає знайти набір маршрутів, які дозволять зменшити витрати на перевезення врожаю.

Алгоритм Прима будує мінімальне остовне дерево, починаючи з однієї вершини і поступово додаючи до неї коротші ребра, що не утворюють циклів.

```
# Знаходження мінімального остовного дерева (MST) алгоритмом Прима
mst = nx.minimum_spanning_tree(G, algorithm='prim')
print("Ребра MST (Прима):", list(mst.edges(data=True)))
```

Алгоритм Крускала працює за іншим принципом. Він сортує всі ребра за зростанням ваги та додає їх до мінімального остовного дерева, тільки якщо це не створює циклів.

```
# Знаходження MST алгоритмом Крускала
mst_kruskal = nx.minimum_spanning_tree(G, algorithm='kruskal')
print("Ребра MST (Крускала):", list(mst_kruskal.edges(data=True)))
```

Список використаних джерел:

1. NetworkX. Документація NetworkX. URL: https://networkx.org/documentation/stable/
2. Matplotlib. Документація Matplotlib. URL: https://matplotlib.org/stable/
3. Floyd R.W. Алгоритм 97: Найкоротший шлях. Communications of the ACM, 1962, 5(6), 345.
4. Prim R.C. Найкоротші мережі зв'язку та деякі узагальнення. Bell System Technical Journal, 1957, 36(6), 1389–1401.

Anotation: *The use of graph algorithms for optimizing harvest transportation logistics has been studied. Algorithms for shortest path search and minimum spanning tree have been considered, along with their implementation in Python using the NetworkX library. The efficiency of these methods for the agricultural sector has been determined, as well as the prospects for their integration with modern optimization technologies..*
Key words: *logistics optimization, crop transportation, graph algorithms, Python, NetworkX, shortest path, transportation network, agricultural sector*.

Науковий керівник:
Пархоменко О. Ю.,
к. ф.-м. н., доцент кафедри економічної кібернетики, комп'ютерних наук та інформаційних технологій, Миколаївський національний аграрний університет

УДК 004.056

Основні аспекти забезпечення інформаційної безпеки

Дмитро Кулєшов,
здобувач вищої освіти спеціальність 204 Технологія ВППТ
Миколаївський національний аграрний університет
м. Миколаїв, Україна

Анотація: *у роботі розглядаються основні аспекти забезпечення інформаційної безпеки. Проаналізовано методи захисту інформації в інформаційних системах.*
Ключові слова: *інформаційна безпека, конфіденційність, цілісність, доступність, аутентифікація, авторизація.*

Забезпечення інформаційної безпеки – це процес захисту інформації від несанкціонованого доступу, пошкодження, змін або знищення, а також забезпечення її доступності та цілісності.

Конфіденційність є одним із основних аспектів інформаційної безпеки. Вона передбачає захист інформації від несанкціонованого доступу, зміни чи розголошення, тобто, забезпечення того, щоб інформація була доступна тільки тим особам або системам, яким це дозволено. Важливість конфіденційності полягає в захисті чутливої інформації, що може бути предметом комерційної