# ІНФОРМАЦІЙНО-АНАЛІТИЧНІ СИСТЕМИ В ОПЕРАТИВНОМУ ПЛАНУВАННІ: АРХІТЕКТУРА ДАНИХ ТА АЛГОРИТМІЧНІ СТРАТЕГІЇ ПОШУКУ

## Вступ

Сучасний оперативний простір визначається не лише кінетичним потенціалом збройних сил, але й, перш за все, здатністю командування досягати інформаційної переваги. У доктринальному вимірі країн НАТО та провідних армій світу інформація розглядається не просто як допоміжний елемент забезпечення, а як окремий операційний домен, рівнозначний суходолу, морю, повітрю чи космосу. Для військового аналітика оперативного рівня, який працює в штабах бригад, корпусів або об'єднаних угруповань військ, це означає фундаментальну зміну парадигми: від роботи з паперовими картами до управління складними потоками даних у реальному часі. Інформаційно-аналітичні системи (ІАС) виступають технологічним скелетом цього процесу, забезпечуючи збір, обробку, зберігання та розповсюдження критично важливих даних, що формують Загальну операційну картину (Common Operational Picture — COP).

Ефективність функціонування ІАС безпосередньо залежить від двох базових компонентів комп'ютерних наук: структур даних, які визначають спосіб організації інформації для її ефективного зберігання, та алгоритмів пошуку, що дозволяють вичленовувати релевантні факти з масиву "великих даних" (Big Data). В умовах "туману війни" (fog of war), коли обсяг вхідних повідомлень (розвідувальних зведень, логістичних звітів, сигналів сенсорів) перевищує когнітивні можливості людини, саме правильно підібрані алгоритми та структури даних дозволяють автоматизувати цикл OODA (Спостереження — Орієнтація — Рішення — Дія), скорочуючи час реакції на загрози.[1]

Цей звіт має на меті надати вичерпний огляд технічних аспектів роботи з даними в ІАС, адаптований для потреб військових аналітиків. Ми розглянемо, як абстрактні концепції програмування, такі як графи, дерева JSON або векторні простори, трансформуються у практичні інструменти для планування логістики, відстеження бойового порядку (ORBAT) та аналізу розвідувальних даних. Особлива увага приділяється застосуванню мови програмування Python як "швейцарського ножа" сучасного аналітика, а також використанню великих мовних моделей (LLM) через інтерфейс ChatGPT для прискорення розробки аналітичних скриптів.

# Частина 1. Структура даних в інформаційно-аналітичних системах (IAC)

Структура даних — це не просто технічний формат збереження бітів інформації; це логічна модель, що відображає реальний світ у цифровій формі. Від того, наскільки точно структура даних описує фізичні об'єкти (підрозділи, техніку, місцевість) та їхні взаємозв'язки, залежить здатність IAC надавати коректні відповіді на запити командування. У військовій сфері ми стикаємося з унікальним викликом: необхідністю поєднувати жорстко стандартизовані дані (наприклад, класифікатори НАТО) з хаотичними, неструктурованими даними, що надходять з поля бою.

## 1.1. Типологія та ієрархія військових даних

Розуміння природи даних є першим кроком до їх ефективного використання. В IAC оперативного рівня дані класифікуються за рівнем структурованості, що визначає методи їх обробки та інтеграції.

### 1.1.1. Структуровані дані (Structured Data): Фундамент сумісності

Структуровані дані характеризуються чітко визначеною схемою (schema), де кожен елемент має свій тип (число, рядок, дата) та семантичне значення. Це мова баз даних (SQL) та таблиць, яка є основою для автоматизованого обліку.

| Тип даних | Опис та військове застосування | Приклади | Джерела |
|---|---|---|---|
| **Геопросторові координати** | Числові значення, що визначають положення об'єкта у просторі. Вимагають чіткої вказівки системи координат (Datum), зазвичай WGS84. | MGRS: 36U VB 1234 5678<br><br>Lat/Lon: 48.450, 37.550 | [3] |

| | | | |
|---|---|---|---|
| **Часові мітки (Temporal)** | Точні відмітки часу для синхронізації дій. Критичні для відновлення хронології подій. Стандарт DTG (Date-Time Group). | 221430ZJAN26 (14:30 UTC, 22 січня 2026 року) | [5] |
| **Логістичні класифікатори** | Кодифіковані ідентифікатори ресурсів для автоматизації ланцюгів постачання. | NSN (National Stock Number): 1005-01-565-8954<br><br>Клас постачання: Class V (W) - Ground Ammo | [7] |
| **Ідентифікатори підрозділів** | Унікальні коди для однозначної ідентифікації одиниць у базах даних. | UIC (Unit Identification Code): WA1234<br><br>SIDC (Symbol ID Code): SFGPUCA--- | [9] |

Військові стандарти, такі як STANAG, вимагають жорсткої типізації для забезпечення інтероперабельності (сумісності) між підрозділами різних країн. Наприклад, помилка у форматі координат при передачі запиту на вогневу підтримку (Call for Fire) може мати фатальні наслідки, тому IAC використовують валідатори на рівні структури даних для перевірки вводу.[2]

## 1.1.2. Напівструктуровані дані (Semi-structured Data): Гнучкість JSON

Сучасна війна динамічна, і жорсткі схеми баз даних не завжди встигають за змінами організаційної структури. Напівструктуровані дані, зокрема формати JSON (JavaScript Object Notation) та XML, стали стандартом де-факто для обміну оперативною інформацією між системами C4ISR (Command, Control, Communications, Computers, Intelligence, Surveillance and Reconnaissance).

JSON дозволяє моделювати складні ієрархічні відносини, такі як **Бойовий порядок (Order of Battle — ORBAT)**. ORBAT описує ієрархію підпорядкування, яка може бути глибоко вкладеною (Корпус -> Бригада -> Батальйон -> Рота -> Взвод) і нерегулярною

(один батальйон має 3 роти, інший — 4 плюс приданий підрозділ БПЛА).[1]

**Концептуальна модель ORBAT у JSON:**

У Python така структура природним чином відображається через вкладені словники (dict) та списки (list). Це дозволяє аналітику програмно "обходити" дерево підпорядкування, підраховуючи загальну боєздатність (Combat Power) на будь-якому рівні ієрархії.

Python

```
# Приклад структури ORBAT (Python Dictionary / JSON)
orbat_data = {
  "unit_id": "UA-72-BDE",
  "name": "72-га окрема механізована бригада",
  "type": "Mechanized Infantry",
  "command_level": "Brigade",
  "status": "Green",
  "location": { # Вкладений об'єкт геоданих
    "mgrs": "36U VB 1234 5678",
    "lat": 48.45,
    "lon": 37.55
  },
  "subunits":
      }
    },
    {
      "unit_id": "UA-72-ART-GRP",
      "name": "Бригадна артилерійська група",
      "subunits": # Можлива подальша вкладеність (дивізіони, батареї)
    }
  ]
}
```

Така структура дозволяє додавати нові поля (наприклад, "drone_capability") без необхідності переписувати всю базу даних, що критично важливо для адаптації до нових засобів ведення війни.[2]

## 1.1.3. Неструктуровані дані (Unstructured Data): Океан інформації

Близько 80% розвідувальної інформації надходить у неструктурованому вигляді: текстові повідомлення в месенджерах, звіти SITREP у форматі вільного тексту, перехоплення радіопереговорів, відеопотоки з БПЛА та супутникові знімки.[13] Для ІАС головним викликом є перетворення цього "сирого" матеріалу на структуровані дані, придатні для аналізу.

Текстові звіти, такі як **SITREP (Situation Report)**, часто слідують формальному шаблону (наприклад, параграфи 1. OPERATIONAL, 2. INTELLIGENCE), але заповнюються людьми, що призводить до варіативності формулювань, помилок та скорочень. Парсинг таких даних вимагає складних алгоритмів обробки тексту.[5]

---

# 1.2. Спеціалізовані формати даних для військової аналітики

Для оперативного рівня критично важливими є спеціалізовані формати, що дозволяють кодувати простір, час та ресурси.

## 1.2.1. Геопросторові дані: GeoJSON та система MGRS

Військова картографія вимагає високої точності та стандартизації. Хоча цивільні системи (GPS) використовують широту/довготу, військовим стандартом НАТО для наземних операцій є **MGRS (Military Grid Reference System)**. MGRS розбиває поверхню Землі на сітку, що спрощує передачу цілевказання та оцінку відстаней.

**Структура координат MGRS:**

Координата 36U VB 12345 67890 складається з:

1. **GZD (Grid Zone Designator):** 36U — зона UTM розміром 6°×8°.
2. **Square ID:** VB — квадрат 100×100 км.
3. **Easting/Northing:** 12345 67890 — цифрова частина, що визначає місцезнаходження всередині квадрату з точністю до 1 метра.[3]

В ІАС ці дані часто інкапсулюються у формат **GeoJSON**.[15] Це розширення JSON для кодування географічних об'єктів. GeoJSON підтримує різні типи геометрії:

- Point: Розташування окремого підрозділу, спостережного пункту або інциденту.
- LineString: Маршрути руху колон (MSR), передній край оборони (FLOT).
- Polygon: Зони відповідальності (AOR), райони зосередження, зони ураження артилерії, заборонені для польотів зони (NFZ).

Використання GeoJSON дозволяє інтегрувати оперативну обстановку безпосередньо у веб-інтерфейси ІАС (на базі бібліотек Leaflet або OpenLayers) та обмінюватися шарами

карти між різними штабами.[2]

## 1.2.2. Логістичні дані та класи постачання

Логістика — це "кров" війни. Структурування логістичних даних базується на класифікації постачання НАТО та США (Classes of Supply). Це дозволяє агрегувати потреби та запаси у матричні структури даних для планування.[7]

**Таблиця 1. Основні класи постачання (NATO/US) та їх представлення в даних**

| Клас | Найменування | Структура даних в IAC (приклад полів) | Примітки |
|---|---|---|---|
| **I** | Продовольство (Subsistence) | {'type': 'MRE', 'qty': 5000, 'unit': 'packs', 'expiry': '2027-01'} | Включає воду та бойові раціони. |
| **III** | Паливо (POL - Petroleum, Oil, Lubricants) | {'type': 'Diesel', 'qty': 20000, 'unit': 'liters', 'container': 'tanker'} | Критичний ресурс для механізованих підрозділів. |
| **V** | Боєприпаси (Ammunition) | {'type': '155mm HE', 'qty': 300, 'fuzes': 300, 'charges': 310} | Вимагає детального обліку супутніх елементів (підривників). |
| **VIII** | Медичні матеріали (Medical) | {'type': 'Blood Plasma', 'qty': 50, 'storage': 'refrigerated'} | Висока чутливість до умов зберігання. |
| **IX** | Запчастини (Repair Parts) | {'nsn': '2530-01-123-4567', 'component': 'Track pad', 'qty': 100} | Найбільший за номенклатурою клас (тисячі позицій). |

В IAC логістичні дані часто моделюються як **графи (Graphs)**, де вузлами є склади та

підрозділи, а ребрами — транспортні шляхи з пропускною здатністю. Це дозволяє застосовувати алгоритми оптимізації потоків (Network Flow Analysis) для планування підвозу в умовах протидії (Contested Logistics).[17]

---

# 1.3. Інструментарій Python для роботи з даними IAC

Python став основною мовою для військової аналітики завдяки своїй екосистемі бібліотек, що покривають всі аспекти роботи з даними: від парсингу тексту до геопросторового аналізу.

## 1.3.1. Парсинг SITREP з використанням TextFSM та Regex

Стандартні звіти, хоч і мають структуру, часто надходять у вигляді "плоского тексту". Для перетворення їх у структуровані дані (наприклад, для занесення в базу даних подій) використовується підхід на основі шаблонів. **TextFSM** — це бібліотека від Google, розроблена для парсингу виводу мережевого обладнання, але вона ідеально підходить для напівструктурованих військових звітів.[19] Вона дозволяє визначити шаблон (template), який описує структуру звіту, і автоматично витягувати значення у таблицю.

Для простіших завдань використовуються **регулярні вирази (Regex)**. Наприклад, вилучення координат MGRS з тексту звіту:

Python

```
import re

text_snippet = "Enemy activity observed at 36U VB 12345 67890 moving south."
# Патерн для пошуку MGRS координат:
# \d{1,2}[A-Z] - Grid Zone (напр., 36U)
# \s+ - пробіли
# [A-Z]{2} - 100km Square ID (напр., VB)
# \s+ - пробіли
# \d{5}\s+\d{5} - Easting та Northing (по 5 цифр для точності 1м)
mgrs_pattern = r"(\d{1,2}[A-Z]\s+[A-Z]{2}\s+\d{4,5}\s+\d{4,5})"

matches = re.findall(mgrs_pattern, text_snippet)
print(f"Виявлені координати: {matches}")
# Результат:
```

Цей підхід є критичним для OSINT (Open Source Intelligence), коли необхідно обробити

тисячі повідомлень з соціальних мереж або Telegram-каналів для виявлення потенційних цілей або індикаторів активності.[13]

### 1.3.2. Робота з геоданими: MGRS та GeoPandas

Для аналізу геопросторових даних використовується бібліотека geopandas, яка розширює можливості pandas для роботи з геометричними об'єктами. Бібліотека mgrs забезпечує конвертацію між військовою системою координат та цивільною.[3]

Python

```
import mgrs

m = mgrs.MGRS()
coord_mgrs = '36U VB 12345 67890'
# Конвертація MGRS у Latitude/Longitude
lat, lon = m.toLatLon(coord_mgrs)
print(f"Lat: {lat}, Lon: {lon}")
# Це дозволяє нанести точку на карту Google Maps або OpenStreetMap
```

---

# 1.4. Завдання для ChatGPT (Тема: Структура даних)

Військовий аналітик може використовувати ChatGPT як "цифрового асистента" для написання коду, валідації даних та конвертації форматів. Нижче наведено 10 варіантів завдань, розроблених з урахуванням специфіки оперативної роботи. Кожне завдання містить мету, контекст та очікуваний результат.

**Інструкція:** Скопіюйте промпт у ChatGPT, додавши ваші специфічні дані.

1. **Конвертація ORBAT (Text to JSON)**
   - **Мета:** Автоматизувати створення цифрової моделі підрозділу.
   - **Промпт:** "Дій як фахівець з даних. Я маю текстовий опис структури мотопіхотної бригади (список батальйонів, рот, взводів). Перетвори цей текст у валідний JSON об'єкт, де кожен рівень ієрархії є вкладеним словником. Додай поля 'commander_rank' (звання командира) для кожного рівня, припускаючи стандартні звання (Brigade-Colonel, Battalion-LtCol тощо)."
2. **Валідація та екстракція MGRS**
   - **Мета:** Очищення "брудних" даних з розвідувальних зведень.
   - **Промпт:** "Напиши Python-скрипт з використанням re, який сканує вхідний текст

(SITREP) і витягує всі координати MGRS. Скрипт має перевіряти валідність формату (наприклад, правильність Grid Zone Designator) і стандартизувати їх, прибираючи зайві пробіли. Виведи список унікальних координат."

3. **Генерація GeoJSON шару обстановки**
    - **Мета:** Візуалізація позицій на карті.
    - **Промпт:** "У мене є CSV файл з колонками: 'Unit_Name', 'MGRS', 'Affiliation' (Friendly/Enemy/Neutral). Напиши скрипт на Python, який використовує бібліотеку mgrs для конвертації координат у Lat/Lon і створює файл situation.geojson. Присвой кожній точці властивість 'marker-color' (Blue для Friendly, Red для Enemy), щоб файл можна було відкрити на web-карті."
4. **Парсинг логістичного звіту (Regex)**
    - **Мета:** Автоматизований облік боєприпасів.
    - **Промпт:** "Напиши функцію парсингу на Python для тексту: 'Ammo Status: 155mm - 400rds, 120mm - 150rds, 7.62mm - 10000rds'. Функція має повертати словник {'155mm': 400, '120mm': 150,...}. Врахуй можливість різних роздільників (тире, двокрапка)."
5. **Створення графу логістичної мережі**
    - **Мета:** Підготовка даних для оптимізації маршрутів.
    - **Промпт:** "Використовуючи бібліотеку networkx, напиши код для створення графу постачання. Вузли: 'Main Hub', 'FOB Alpha', 'FOB Bravo'. Ребра: дороги з атрибутами 'distance_km' та 'risk_level' (0-10). Згенеруй приклад графу і покажи, як зберегти його у форматі GEXF або GraphML."
6. **Анонімізація особових даних (PII)**
    - **Мета:** Захист персональних даних при передачі звітів.
    - **Промпт:** "Напиши скрипт, який приймає JSON файл з даними про особовий склад (поля 'name', 'rank', 'unit'). Заміни поле 'name' на хеш SHA-256, щоб знеособити дані для аналітичного звіту, але зберегти унікальність для підрахунку статистики."
7. **Злиття даних (Data Merging) за ключем**
    - **Мета:** Об'єднання інформації з різних джерел (наприклад, штатний розклад та фактична наявність).
    - **Промпт:** "У мене є два списки словників у Python. Перший: [{'id': 'u1', 'authorized_strength': 100}]. Другий: [{'id': 'u1', 'current_strength': 85}]. Напиши код для об'єднання цих списків в один за ключем 'id' та розрахунку відсотка укомплектованості ('readiness_percent')."
8. **Перетворення дат у формат DTG NATO**
    - **Мета:** Стандартизація часових міток.
    - **Промпт:** "Напиши функцію на Python, яка приймає дату у форматі ISO 8601 ('2026-01-22T14:30:00') і конвертує її у військовий формат DTG ('221430ZJAN26'). Забезпеч коректну обробку часових поясів (переведення в Zulu/UTC)."
9. **Аналіз структури ACLED даних**
    - **Мета:** Робота з відкритими базами даних конфліктів.
    - **Промпт:** "На основі структури даних проекту ACLED (поля event_date, actor1,

actor2, event_type, location), напиши код на Python з використанням pandas для фільтрації подій. Мені потрібно вибрати всі події типу 'Explosions/Remote violence' за останній місяць у регіоні 'Donetsk'."

10. **Генерація синтетичних даних SITREP**
    - **Мета:** Створення тестового набору даних для навчання моделей.
    - **Промпт:** "Згенеруй Python-скрипт, який створює 50 фіктивних звітів SITREP. Кожен звіт має містити випадкову дату, координату MGRS у межах заданого квадрату, тип інциденту (обстріл, переміщення, БПЛА) та короткий опис тексту. Збережи результат у CSV для тестування аналітичної системи."

---

# Частина 2. Алгоритми пошуку в IAC

Наявність структурованих даних є необхідною, але недостатньою умовою для ефективної аналітики. Головне завдання аналітика — знайти "сигнал у шумі". В контексті IAC це означає застосування алгоритмів, здатних фільтрувати, ранжувати та зіставляти інформацію з величезною швидкістю та точністю.

## 2.1. Класифікація алгоритмів пошуку у військовому контексті

### 2.1.1. Текстовий пошук: Від ключових слів до Regex

Базовий рівень пошуку — це знаходження точних відповідностей. Однак у військових текстах часто використовуються абревіатури, кодові слова та нестандартні позначення.

- **Булевий пошук (Boolean Search):** Використання операторів AND, OR, NOT. Наприклад, запит (TANK OR ARMORED) AND (DESTROYED OR DAMAGED) NOT SIMULATION дозволяє відфільтрувати релевантні бойові звіти.
- **Регулярні вирази (Regex):** Дозволяють шукати за патерном. Це незамінний інструмент для пошуку серійних номерів техніки, позивних або специфічних форматів повідомлень, які можуть змінюватися в деталях, але зберігають структуру.

### 2.1.2. Нечіткий пошук (Fuzzy Search): Подолання "людського фактора"

Однією з найбільших проблем військових даних є помилки вводу. Назви населених пунктів (топоніми) часто транслітеруються по-різному (наприклад, "Kyiv", "Kiev", "Kyyiv"), а прізвища можуть бути записані з одруками. Точний пошук у таких випадках дасть нульовий результат, що може призвести до пропуску важливих розвідданих.

**Алгоритм відстані Левенштейна (Levenshtein Distance):** Цей алгоритм обчислює

мінімальну кількість операцій (вставка, видалення або заміна символу), необхідних для перетворення одного рядка в інший. В IAC він використовується для дедуплікації записів про особистості (Entity Resolution) та прив'язки подій до географічних назв.[22]

**Бібліотека fuzzywuzzy / thefuzz у Python:**

Дозволяє реалізувати "м'який" пошук, повертаючи ступінь схожості у відсотках.

Python

```
from fuzzywuzzy import process

# Список відомих цілей (правильні назви)
targets =

# Запит з радіоперехоплення (з помилкою)
input_query = "Amo Depot Alfa"

# Пошук найкращого співпадіння
match, score = process.extractOne(input_query, targets)
print(f"Match: '{match}' with score {score}%")
# Результат: Match: 'Ammo Depot Alpha' with score 89%
```

Якщо схожість перевищує певний поріг (наприклад, 80%), система може автоматично зв'язати подію з об'єктом; якщо менше — подати сигнал аналітику для ручної перевірки.[24]

## 2.1.3. Геопросторовий пошук (Geospatial Search)

Війна відбувається у просторі, тому значна частина запитів до IAC є просторовими.

- **Point-in-Polygon (Точка в полігоні):** Фундаментальний алгоритм для визначення належності об'єкта до зони.
  - *Застосування:* Чи знаходиться наш підрозділ у зоні ураження ворожої артилерії? Чи потрапила колона у замінований район?
  - *Реалізація:* Бібліотека shapely та geopandas використовують алгоритм Ray Casting (випускання променя) для перевірки належності точки полігону.[25]
- **Радіальний пошук (Within Distance):** Знаходження всіх об'єктів у радіусі $R$.
  - *Застосування:* Знайти всі пункти заправки в радіусі 50 км від маршруту руху.
- **Найближчий сусід (k-Nearest Neighbors - kNN):**
  - *Застосування:* Вибір найближчого пункту евакуації (MEDEVAC) для пораненого.[27]

### 2.1.4. Графові алгоритми та мережевий аналіз

Логістика та зв'язок моделюються як графи. Ефективність мережі визначається не лише її структурою, але й потоками, що проходять через неї.

- *Алгоритм Дейкстри (Dijkstra's Algorithm) та A:** Пошук найкоротшого шляху у зваженому графі.
  - *Військове застосування:* Планування маршруту колони. Вагою ребра може бути не лише відстань, але й час проходження (враховуючи стан дороги) або рівень ризику (ймовірність засідки). Алгоритм шукає шлях з мінімальною сумарною вагою ("найдешевший" шлях), а не обов'язково найкоротший геометрично.[28]
- **Аналіз центральності (Centrality Measures):**
  - *Betweenness Centrality:* Визначає вузли, через які проходить найбільша кількість найкоротших шляхів. У логістиці це "вузькі місця" (bottlenecks) — мости або перехрестя, знищення яких паралізує мережу. У розвідці (SNA - Social Network Analysis) — це ключові комунікатори в терористичній мережі.[29]

### 2.1.5. Семантичний пошук (Semantic Search)

Традиційний пошук за словами не розуміє контексту. Семантичний пошук використовує векторні представлення (embeddings), де текст перетворюється на числовий вектор. Близькість векторів (Cosine Similarity) означає близькість за змістом.

- *Застосування:* Запит "ворожа бронетехніка біля води" знайде повідомлення "танки противника скупчуються біля понтонної переправи", навіть якщо спільних слів немає. Це революціонізує роботу з архівами розвідданих.[30]

---

# 2.2. Практичні сценарії використання алгоритмів (Python Cases)

## Сценарій 1: Геопросторовий фільтр безпеки

Аналітик повинен автоматично перевіряти, чи не заходять дружні патрулі в зони заборони доступу (No-Go Areas).

Python

```
import geopandas as gpd
from shapely.geometry import Point, Polygon
```

```python
# 1. Визначення зони мінних полів (Полігон)
minefield_coords = [(0, 0), (0, 10), (10, 10), (10, 0)] # Умовні координати
minefield = Polygon(minefield_coords)

# 2. Дані трекінгу патрулів (Точки)
patrols = gpd.GeoDataFrame({
    'patrol_id': ['P-1', 'P-2', 'P-3'],
    'geometry': [Point(5, 5), Point(12, 12), Point(9, 1)]
})

# 3. Алгоритмічна перевірка: Point in Polygon
# Функція.within() повертає булеву маску (True/False)
danger_mask = patrols.within(minefield)
patrols_in_danger = patrols[danger_mask]

if not patrols_in_danger.empty:
    print(f"УВАГА! Патрулі у небезпечній зоні: {patrols_in_danger['patrol_id'].tolist()}")
else:
    print("Всі патрулі у безпеці.")
```

## Сценарій 2: Оптимізація логістики (NetworkX)

Планування евакуації техніки до ремонтного батальйону (Repair Depot) з урахуванням пропускної здатності доріг.

Python

```
import networkx as nx

# Створення графу доріг
G = nx.Graph()

# Додавання ребер (доріг). weight = час проїзду в хвилинах
G.add_edge("Frontline_Point", "Crossroad_A", weight=20)
G.add_edge("Frontline_Point", "Crossroad_B", weight=35)
G.add_edge("Crossroad_A", "Repair_Depot", weight=50) # Дорога погана
G.add_edge("Crossroad_B", "Repair_Depot", weight=25) # Траса

# Пошук найшвидшого маршруту (Алгоритм Дейкстри)
shortest_path = nx.dijkstra_path(G, source="Frontline_Point", target="Repair_Depot",
weight="weight")
```

```
total_time = nx.dijkstra_path_length(G, source="Frontline_Point", target="Repair_Depot",
weight="weight")

print(f"Маршрут евакуації: {' -> '.join(shortest_path)}")
print(f"Розрахунковий час: {total_time} хв")
# Алгоритм обере шлях через Crossroad_B (35+25=60), попри те, що через A він може здаватися
коротшим на карті, але довшим за часом (20+50=70).
```

---

# 2.3. Завдання для ChatGPT (Тема: Алгоритми пошуку)

Ці завдання спрямовані на те, щоб навчити аналітика формулювати складні алгоритмічні запити до ШІ.

1. **Реалізація нечіткого пошуку (Fuzzy Matching)**
   - **Мета:** Дедуплікація списків особового складу.
   - **Промпт:** "Напиши скрипт на Python, що використовує бібліотеку thefuzz. Порівняй два списки прізвищ: список А (офіційний реєстр) та список Б (імена зі звітів з можливими помилками). Виведи всі пари імен, які мають коефіцієнт схожості (similarity score) вище 85%, але не є ідентичними (100%)."
2. **Геопросторовий радіальний пошук**
   - **Мета:** Аналіз покриття засобами зв'язку або вогневої підтримки.
   - **Промпт:** "У мене є координата базової станції (Lat/Lon) і список координат абонентів. Напиши функцію на Python, яка використовує формулу Haversine для розрахунку відстані на сфері. Функція має відфільтрувати та повернути список всіх абонентів, що знаходяться в радіусі 15 км від станції."
3. **Пошук патернів у часових рядах (Time Series)**
   - **Мета:** Виявлення аномалій в активності ворога.
   - **Промпт:** "Я маю список часових міток (timestamps) ворожих обстрілів за місяць. Напиши скрипт на Python, який агрегує кількість подій по годинах доби (0-23) і будує гістограму. Також напиши алгоритм для виявлення годин, коли активність перевищує середнє значення на 2 стандартних відхилення (аномальні сплески)."
4. **Алгоритм Дейкстри з динамічними вагами**
   - **Мета:** Планування маршруту з урахуванням загроз.
   - **Промпт:** "Використовуючи networkx, створи граф, де ребра мають два атрибути: 'distance' і 'risk_factor' (множник від 1.0 до 5.0). Напиши код, який знаходить шлях між двома точками, мінімізуючи не просто відстань, а добуток відстані на фактор ризику (Custom Weight Function)."
5. **Кластеризація подій (K-Means)**
   - **Мета:** Виявлення центрів тяжіння активності противника.
   - **Промпт:** "У мене є набір координат (Lat/Lon) місць падіння снарядів.

Використовуй бібліотеку scikit-learn та алгоритм K-Means для групування цих точок у кластери. Визнач центроїди кластерів, щоб виявити ймовірні цілі або райони зосередження вогню."

6. **Парсинг та фільтрація JSON (Deep Search)**
   - **Мета:** Пошук у глибоко вкладених структурах даних.
   - **Промпт:** "Напиши рекурсивну функцію на Python для пошуку у складному JSON об'єкті (ORBAT). Функція має проходити через усі рівні вкладеності і знаходити всі підрозділи, у яких стан боєздатності ('status') дорівнює 'Critical', повертаючи список їхніх ID та батьківських підрозділів."
7. **Семантичний пошук (Концептуальний)**
   - **Мета:** Розуміння принципів роботи Vector Search.
   - **Промпт:** "Поясни принцип роботи косинусної подібності (Cosine Similarity) для пошуку текстів. Напиши приклад коду з використанням бібліотеки sentence-transformers, який перетворює речення 'Enemy tanks advancing' та 'Armored column moving forward' у вектори і обчислює ступінь їхньої схожості."
8. **Пріоритезація черги евакуації (Priority Queue)**
   - **Мета:** Алгоритмічне управління ресурсами MEDEVAC.
   - **Промпт:** "Напиши реалізацію черги з пріоритетом (Priority Queue) на Python для управління запитами на евакуацію. Запити мають категорії: 'Urgent' (пріоритет 1), 'Priority' (2), 'Routine' (3). Алгоритм має завжди видавати на обробку запит з найвищим пріоритетом, навіть якщо він надійшов пізніше за інші."
9. **Аналіз центральності у соціальній мережі**
   - **Мета:** Виявлення лідерів або ключових зв'язкових.
   - **Промпт:** "Створи граф комунікацій (хто з ким зв'язувався). Напиши код для розрахунку 'Betweenness Centrality' для кожного вузла. Поясни, як інтерпретувати результати: чому вузол з найвищим показником є критично важливим для зв'язності мережі, навіть якщо у нього мало прямих контактів."
10. **Бінарний пошук у логістичній базі**
    - **Мета:** Розуміння ефективності алгоритмів (Big O).
    - **Промпт:** "Маємо відсортований список з 1 мільйона серійних номерів гвинтівок. Напиши власну функцію бінарного пошуку (Binary Search) на Python, щоб знайти конкретний номер. Порівняй теоретичну кількість кроків для бінарного пошуку ( $O(\log N)$ ) та звичайного перебору ( $O(N)$ ) для цього обсягу даних."

---

# Висновки

Інтеграція потужних структур даних та алгоритмів пошуку у повсякденну діяльність військового аналітика є критичним фактором успіху в сучасній війні.

1. **Структурування як передумова аналізу:** Без перетворення хаотичних потоків інформації у формати JSON, GeoJSON або графи неможливе застосування передових методів аналітики. Використання стандартів (MGRS, класи постачання)

забезпечує сумісність та точність.

2. **Алгоритмічна перевага:** Здатність швидко виконувати нечіткий пошук, геопросторову фільтрацію або оптимізацію маршрутів надає командиру дорогоцінний час. У циклі OODA перемагає той, хто швидше проходить етап "Орієнтація" (Orientation), і саме тут алгоритми відіграють вирішальну роль.
3. **Роль Python та AI:** Сучасний аналітик не зобов'язаний бути професійним програмістом, але повинен вміти використовувати інструменти (LLM, бібліотеки Python) для вирішення ad-hoc задач. Наведені приклади та завдання демонструють, як за допомогою кількох рядків коду можна вирішувати складні операційні проблеми, які раніше вимагали годин ручної праці.

Опанування цими інструментами перетворює аналітика з "оператора Excel" на архітектора інформаційної переваги.

## Джерела


1. The Vital Planning and Analysis (ViPA) ORBAT Data Service Architecture and Design Overview - Defence Science and Technology, доступ отримано січня 22, 2026, https://www.dst.defence.gov.au/sites/default/files/publications/documents/DST-Group-TN-1539.pdf
2. Simulation-Supported Wargaming for Analysis of Plans - NATO, доступ отримано січня 22, 2026, https://publications.sto.nato.int/publications/STO%20Meeting%20Proceedings/STO-MP-MSG-133/MP-MSG-133-12.pdf
3. Understanding and converting MGRS coordinates in Python | dida blog, доступ отримано січня 22, 2026, https://dida.do/blog/understanding-mgrs-coordinates
4. pygeodesy.mgrs, доступ отримано січня 22, 2026, https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.mgrs-module.html
5. MILOBS Team Site (TS) Daily Situation Report (SITREP), доступ отримано січня 22, 2026, https://resourcehub01.blob.core.windows.net/training-files/Training%20Materials/021%20STM-UNMO/021-029%20Annex%20D%20-%20References%20(Formats%20-%20All%20Reports).pdf
6. ACLED Codebook, доступ отримано січня 22, 2026, https://acleddata.com/methodology/acled-codebook
7. Military Classes and Subclasses of Supply Page 1 of 2 - DAU, доступ отримано січня 22, 2026, https://www.dau.edu/sites/default/files/Migrated/CopDocuments/Military%20Classes%20of%20Supply.pdf
8. Classes of supply - Wikipedia, доступ отримано січня 22, 2026, https://en.wikipedia.org/wiki/Classes_of_supply
9. CJCSM 3150.23E, “Joint Reporting Structure Logistics Factors Report” - Joint Chiefs of Staff, доступ отримано січня 22, 2026, https://www.jcs.mil/Portals/36/Documents/Library/Manuals/CJCSM%203150.23E.

pdf
10. Python military symbols - PyPI, доступ отримано січня 22, 2026, https://pypi.org/project/military-symbol/
11. 14.2. JSON Data - Learning Data Science, доступ отримано січня 22, 2026, https://learningds.org/ch/14/web_json.html
12. orbat-mapper/orbat-mapper: Recreate historic battles and military scenarios in your browser - GitHub, доступ отримано січня 22, 2026, https://github.com/orbat-mapper/orbat-mapper
13. Pythonic way to parse a formatted text file - Stack Overflow, доступ отримано січня 22, 2026, https://stackoverflow.com/questions/31986429/pythonic-way-to-parse-a-formatted-text-file
14. Situation Report (SITREP) Template | The Persimmon Group, доступ отримано січня 22, 2026, https://thepersimmongroup.com/situation-report-sitrep-template/
15. IPB - GitHub, доступ отримано січня 22, 2026, https://cepdnaclk.github.io/e14-4yp-ipb/
16. 12 Essential Python Libraries for Geospatial Data Analysis (with Hands-On Examples), доступ отримано січня 22, 2026, https://www.geoapify.com/python-geospatial-data-analysis/
17. Contested Logistics: Modeling & Simulation in Challenged Environments - Systecon Group, доступ отримано січня 22, 2026, https://www.systecongroup.com/us/modeling-and-simulation-logistics-contested-environments
18. Optimization Approach to - LOGISTIC AND SUPPLY CHAIN - DAU, доступ отримано січня 22, 2026, https://www.dau.edu/sites/default/files/Migrate/ARJFiles/ARJ97/ARJ97_Jones.pdf
19. Introduction to TextFSM in Python - GeeksforGeeks, доступ отримано січня 22, 2026, https://www.geeksforgeeks.org/python/introduction-to-textfsm-in-python/
20. Parsing tmsh command output with Python TextFSM module and some Lessons learned, доступ отримано січня 22, 2026, https://community.f5.com/kb/technicalarticles/parsing-tmsh-command-output-with-python-textfsm-module-and-some-lessons-learned/330438
21. hobuinc/mgrs: Python MGRS library - GitHub, доступ отримано січня 22, 2026, https://github.com/hobuinc/mgrs
22. What is Fuzzy Matching? | Redis, доступ отримано січня 22, 2026, https://redis.io/blog/what-is-fuzzy-matching/
23. Fuzzy String Matching in Python Tutorial - DataCamp, доступ отримано січня 22, 2026, https://www.datacamp.com/tutorial/fuzzy-string-python
24. Fuzzy String Matching Using FuzzyWuzzy - HKUST Library, доступ отримано січня 22, 2026, https://library.hkust.edu.hk/sc/fuzzywuzzy/
25. geopandas.GeoSeries.within, доступ отримано січня 22, 2026, https://geopandas.org/en/stable/docs/reference/api/geopandas.GeoSeries.within.html
26. How to filter a geopandas dataframe for points that fall within a specified polygon ?, доступ отримано січня 22, 2026,

https://en.moonbooks.org/Articles/How-to-filter-a-geopandas-dataframe-for-points-that-fall-within-a-specified-polygon-/
27. Aeromedical Evacuation in NATO - Joint Air Power Competence Centre, доступ отримано січня 22, 2026, https://www.japcc.org/articles/aeromedical-evacuation-in-nato/
28. Navigating Networks with NetworkX: A Short Guide to Graphs in Python - Medium, доступ отримано січня 22, 2026, https://medium.com/data-science/navigating-networks-with-networkx-a-short-guide-to-graphs-in-python-c16cbafe8063
29. Finding Bottleneck Edges in a Graph in O(V+E) - Stack Overflow, доступ отримано січня 22, 2026, https://stackoverflow.com/questions/50178192/finding-bottleneck-edges-in-a-graph-in-ove
30. Introduction to Semantic Search with Python and OpenAI API - DEV Community, доступ отримано січня 22, 2026, https://dev.to/carolinamonte/introduction-to-semantic-search-with-python-and-openai-api-efg
31. Text Embeddings, Classification, and Semantic Search (w/ Python Code) - YouTube, доступ отримано січня 22, 2026, https://www.youtube.com/watch?v=sNa_uiqSlJo