МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук,
управління та адміністрування
Кафедра інформаційних технологій

Кваліфікаційна робота бакалавра

на тему: Розробка інструменту пошуку оптимального маршруту на основі цифрових моделей висот в ГІС

Виконав студент групи К-41
спеціальності 122 «Комп'ютерні науки»
Сулейманов Ілкін Агасадиг огли

Керівник к.геогр.н., доцент
Кузніченко Світлана Дмитрівна

Консультант _____

Рецензент к.техн.н., доцент
Гнатовська Ганна Арнольдівна

Одеса 2021

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ГІС	– географічна інформаційна система
ПЗ	– програмне забезпечення
СКБД	– система керування базами даних
ЦМР	– цифрова модель рельєфу
DEM	– Digital Elevation Model (цифрова модель рельєфу)
ESRI	– Environmental Systems Research Institute
GRASS	– Geographic Resources Analysis Support System
gvSIG	– Generalitat Valenciana, Sistema d'Informació Geogràfica

ВСТУП

В наш час геоінформаційні системи (ГІС) впроваджуються практично у всі сфери діяльності: нерухомість, транспорт, екологія, державний і муніципальний сектор, в промисловості тощо. З розвитком веб-технологій та хмарних сховищ даних відкрилася можливість наочно представляти користувачам різноманітну картографічну інформацію у зручній формі. Створена велика кількість інтернет ресурсів, з відкритими даними.

Можливості ГІС на сьогоднішній день добре використовуються у нашому повсякденному житті. За допомогою ГІС можна оцінити рельєф місцевості, знайти потрібні об’єкти та шлях до них. У зв’язки з тим, дуже актуальним є розробка інструментів для пошуку оптимального маршруту на основі цифрових моделей висот в ГІС. Подібні інструменти можуть бути корисні в різних ситуаціях, наприклад, для побудови маршрутів пошуково-рятувальних загонів.

Метою кваліфікаційної роботи бакалавра є створення набору інструментів геообробки просторової інформації на мові програмування Python для пошуку оптимального маршруту на основі цифрових моделей висот в ГІС.

Для досягнення цієї мети були поставлені наступні завдання:

- провести дослідження моделей та форматів для представлення просторової інформації;
- виконати порівняльний аналіз сучасних ГІС пакетів та їх функціональних можливостей;
- обгрунтувати вибір інструментальних засобів розробки інструментів геообробки просторової інформації;
- виконати етапи проектування та реалізації інструментів геообробки просторової інформації;
- розробити інструмент для проведення геообробки даних, зокрема для пошуку оптимального маршруту на основі цифрових моделей висот;
- виконати тестування інструменту.

Структура кваліфікаційної роботи бакалавра складається з вступу, трьох розділів, висновків, переліку посилань на 17 найменуваннь, додатку. Повний обсяг роботи становить 56 сторінок, містить 19 рисунків і 3 таблиці.

1 ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ ТА ВИКОРИСТАННЯ ГЕОІНФОРМАЦІЙНИХ СИСТЕМ

1.1 Поняття геоінформаційної системи та її структура

Геоінформаційна система (ГІС) – це апаратно-програмний та одночасно людино-машинний комплекс, призначений для збору, обробки, зберігання, відображення і поширення даних, а також отримання нової інформації і знань про просторово-координовані об'єкти і явища [1]. Особливістю даних систем є те, що всі моделюються в ГІС об'єкти і явища мають просторову прив'язку, яка дозволяє аналізувати їх у взаємозв'язкb з іншими просторово-визначеними об'єктами. Кардинальна відмінність ГІС від інших інформаційних систем, є те, що інформація в них наочно представлена у вигляді електронних карт.

З наукової точки зору ГІС – це засіб моделювання і пізнання природних і соціально-економічних систем. ГІС застосовується для дослідження природних, громадських і природно-суспільних об'єктів і явищ, які вивчають науки про Землю і суміжні з ними соціально-економічні науки, а також картографія, дистанційне зондування. У технологічному аспекті ГІС (ГІС-технологія) постає як засіб збору, зберігання, перетворення, відображення і поширення просторово - координованої географічної (геологічної, екологічної) інформації. З виробничої точки зору ГІС є комплексом апаратних пристроїв і програмних продуктів (ГІС - оболонок), призначених для забезпечення управління та прийняття рішень. Таким чином, ГІС може одночасно розглядатися як інструмент наукового дослідження, технологія і продукт ГІС-індустрії. [2].

Сучасні ГІС представляють собою новий тип інтегрованих інформаційних систем, які з одного боку, включають методи обробки даних багатьох автоматизованих систем, з іншого – мають специфіку в організації та обробці даних. Практично це визначає ГІС як багатоцільові, багатоаспектні системи [3].

Всі просторові дані організовуються пошарово: однотипні дані на зем-

ній поверхні групуються в шари. Сукупність усіх цих шарів утворює карту. Об'єкти одного шару повинні бути спільною природи походження (річки, будівлі, дороги) і мати однакову топологічну структуру і розмірність (опис об'єкта точками, лініями або полігонами).

Області застосування ГІС:

- управління земельними ресурсами, земельні кадастри;
- інвентаризація, облік, планування розміщення об'єктів розподіленої виробничої інфраструктури (наприклад, нафтогазовидобувні компанії, мережа бензоколонок, магазинів і т. п.) та управління ними;
- проектування, інженерні вишукування, планування в будівництві, архітектурі;
- тематичне картографування;
- управління наземним, повітряним та водним транспортом (розрахування завантаження, оптимальної траєкторії руху, часу прибуття і т. п.);
- управління природними ресурсами, природоохоронна діяльність та екологія;
- прогнозування надзвичайних ситуацій;
- військова справа та рішення широкого кола специфічних завдань, пов'язаних з розрахунком зон видимості, оптимальних маршрутів руху по пересіченій місцевості з урахуванням протидії тощо.
- сільське господарство, прогнозування врожайності і збільшення виробництва сільськогосподарської продукції, оптимізація її транспортування і збуту.

Геоінформаційна система складається з п'яти ключових складових: апаратні засоби, програмне забезпечення, дані, виконавці і методи, представлені на рис. 1.1 [3].

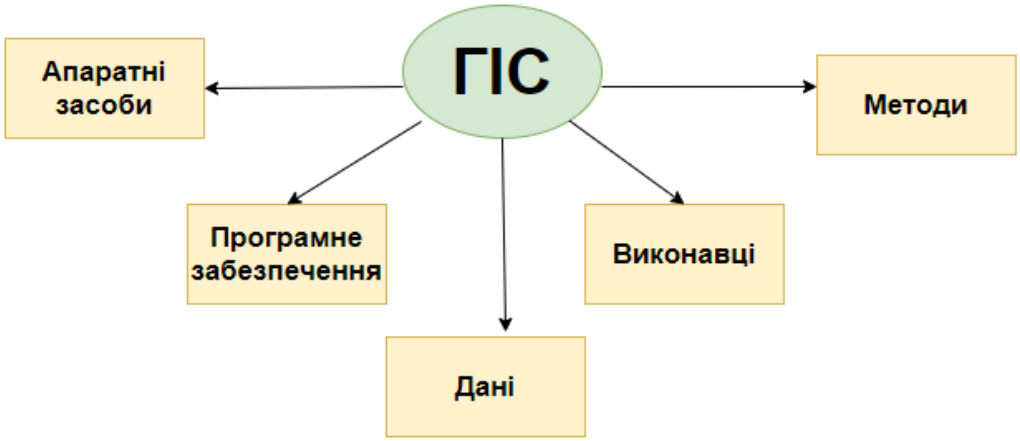

Рисунок 1.1 – Структура ГІС

Апаратні засоби. Це комп'ютер, на якому запущена ГІС. Зараз ГІС працюють на різних типах комп'ютерних платформ, від централізованих серверів до окремих або зв'язаних мережею настільних комп'ютерів.

Програмне забезпечення – це інструменти і функції, які необхідні для зберігання, аналізу і візуалізації географічної (просторової) інформації. Ключовими компонентами програмних продуктів є:

- інструменти для введення і оперування географічною інформацією система управління базою даних (DBMS або СКБД);
- інструменти підтримки просторових запитів, аналізу та візуалізації (відображення);
- графічний користувальницький інтерфейс (GUI або ГІП) для більш зручного доступу до інструментів і функцій.

Дані – найважливіший компонент. Дані про просторове положенні (географічні дані) і пов'язані з ними табличні дані збираються і готуються самим користувачем або купуються у постачальника. У процесі управління просторовими даними географічна інформаційна система об'єднує географічну інформацію з даними інших типів. Наприклад, з конкретним фрагментом електронної карти можуть бути пов'язані вже накопичені дані про населення, характер грунтів, близькості небезпечних об'єктів тощо (в залежності від завдання, яку доведеться вирішувати за допомогою ГІС). У складних, розподілених системах збору та обробки інформації часто з об'єктом на карті пов'я-

зують не існуючі дані, а їх джерело, що дозволяє в реальному часі відстежувати стан цих об'єктів. Цей підхід використовується для боротьби з надзвичайними ситуаціями, наприклад, лісових пожеж або повеней.

Виконавці – це люди, які працюють з програмними продуктами і розробляють плани їх використання при вирішенні поставлених завдань. Для ефективної роботи ГІС необхідне дотримання методів, передбачених розробниками, тому без підготовлених виконавців навіть найвдаліша розробка може втратити будь-який сенс.

Користувачами ГІС можуть бути як технічні фахівці, що розробляють і підтримують систему, так і кінцеві користувачі, яким ГІС допомагає вирішувати поставлені завдання.

Методи. Ефективність застосування ГІС багато в чому залежить від правильно розробленого плану і правил роботи, які складаються відповідно до специфіки завдань і роботи кожної організації.

Зазвичай структура ГІС включає чотири обов'язкові підсистеми:

- введення даних, що забезпечує введення і / або обробку просторових даних, отриманих з карт, матеріалів дистанційного зондування тощо;
- зберігання та пошуку, яка дозволяє оперативно отримувати дані для відповідного аналізу, актуалізувати і коригувати їх;
- обробки та аналізу, яка дає можливість оцінювати параметри, вирішувати розрахунково-аналітичні завдання;
- уявлення (видачі) даних в різному вигляді (карти, таблиці, зображення, блок-діаграми, цифрові моделі місцевості тощо).

Створення карт серед можливостей ГІС займає далеко не перше місце, так як для отримання копію карти абсолютно не потрібна велика частина функцій ГІС, або вони застосовуються опосередковано. Однак ГІС широко використовуються саме для підготовки карт до видання і, в меншій мірі, для аналітичної обробки просторових даних [1].

1.2 Класифікація ГІС

За апаратною платформою виділяють:

- ГІС професійного рівня;
- ГІС настільного типу.

До класичних ГІС професійного рівня відносяться широко відомі системи фірм Intergraph, ESRI, та ін. Системи створені спочатку для функціонування на робочих станціях і для мережевого використання. Вони підтримують численні застосунки, включають блоки векторизації картографічного матеріалу, роботу з великим числом зовнішніх пристроїв.

Настільні ГІС – програмні продукти, що володіють розширеним набором інструментів для роботи з просторовою інформацією. ГІС настільного типу орієнтовані на ПК і призначені для використання широким колом користувачів. Наприклад: AtlasGIS, MapInfo, ArcView, Microstation, WinGIS, Geograph / Geodraw, ПАРК і т. Д. Перераховані ГІС мають менший набором функцій. Вони мають низьку ціну, на їх базі організовуються робочі місця в великих ГІС-проектах, де ГІС будується як багаторівнева система.

За предметною областю моделювання розрізняють: міські (муніципальні), природоохоронні, земельні, геологічні.

За функціональним можливостям розрізняють ГІС:

- універсальні (інструментальні);
- спеціальні;
- ГІС-вьювери.

Універсальні ГІС характеризуються відкритістю, працюють з різними форматами даних, володіють достатньою потужним графічним редактором, мають засоби розробки і впровадження різних застосунків. Це найбільш широко використовуваний клас ГІС, оскільки дозволяють адаптувати різні завдання, збільшувати число вбудованих спеціалізованих модулів.

Спеціальні ГІС вирішують вузьке коло завдань на заданому наборі параметрів. Їх основне завдання – контроль протікання процесів і запобігання небажаних ситуацій, автоматизація документообігу.

ГІС-вьювери призначені для візуалізації просторової інформації, виведення на друк. Ці системи, як правило, не забезпечені апаратом для просторового аналізу і моделювання.

Основні поняття ієрархії інформаційної інтегрованої системи наведені на рис. 1.2 [4].

Рисунок 1.2 – Структура інтегрованої системи

Верхнім рівнем понять є інтегрована система – незалежний комплекс, в якому виконуються всі процеси обробки, обміну та подання інформації. Схема системи включає в себе системні рівні, підсистеми, процеси, завдання.

Система може бути неповною і повною.

Неповною називається система, яка здійснює часткову обробку даних, частковий введення даних або використовує інші системи в процесі обробки.

Повною називається система, яка в процесі роботи здійснює технологічний цикл, що включає наступні процеси [4]:

- введення (або можливість введення) всіх видів інформації даної предметної області для вирішення завдань, поставлених перед системою;
- обробку інформації із залученням набору існуючих засобів, що застосовуються для вирішення даного класу задач;
- виведення або подання даних в формах виведення відповідно до завдання без використання інших систем.

Структуру ГІС зазвичай представляють як набір інформаційних шарів. Шар – сукупність однотипних просторових об'єктів, що відносяться до однієї теми або класу об'єктів в межах певної території і в системі координат, загальних для набору шарів. Геоінформаційна структура даних в ГІС представлена на рис.1.3.

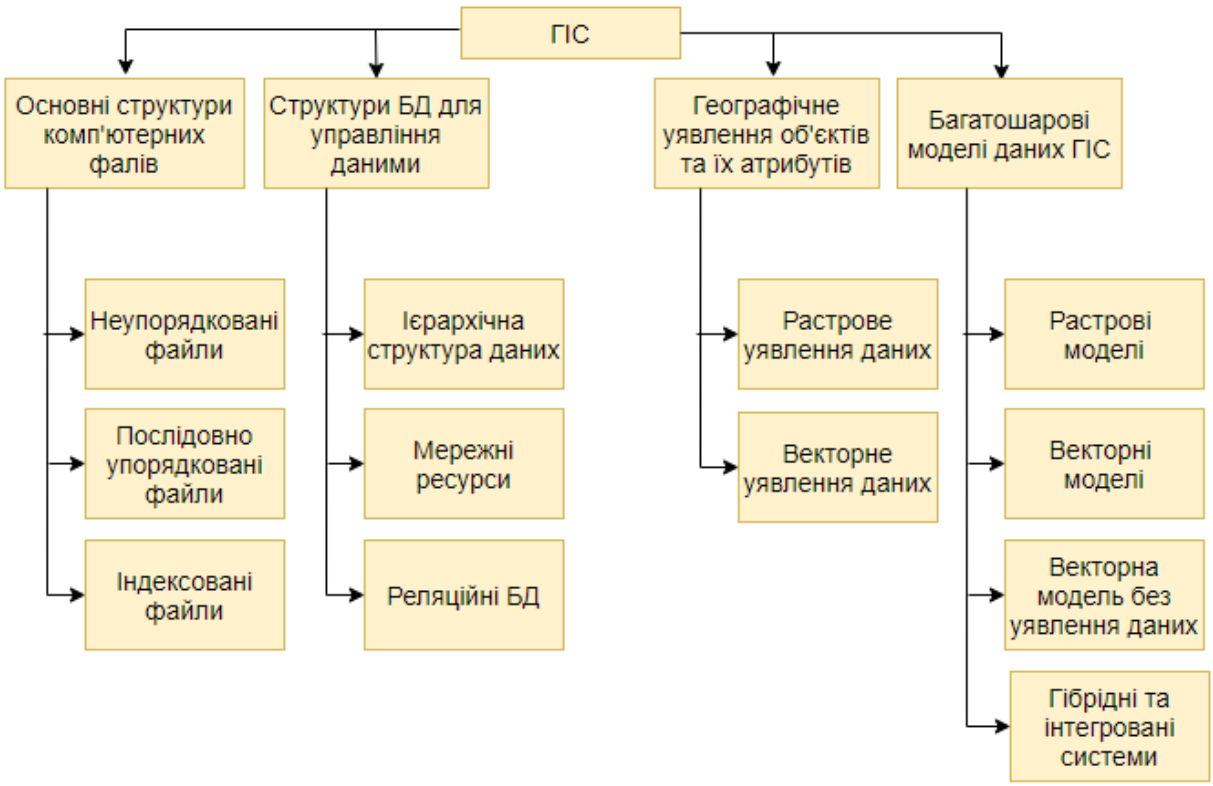

Рисунок 1.3 – Геоінформаційна структура даних в ГІС

Основу будь-якої ГІС становить автоматизована картографічна система – комплекс приладів і програмних засобів, що забезпечують створення і використання карт, яка складається з ряду підсистем, таких як підсистеми введення, обробки і виведення інформації. Функції ГІС представлені на рис.1.4.

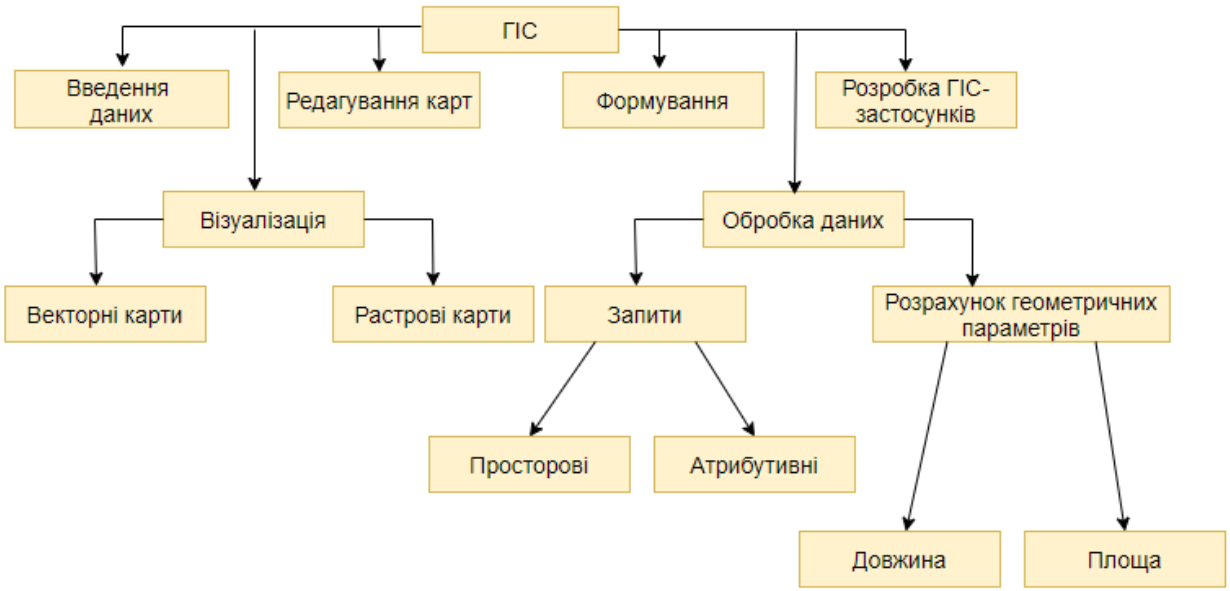

Рисунок 1.4 – Функції ГІС

В ГІС всі об'єкти представлені чотирма типами просторових об'єктів: точки, лінії, області і поверхні. Разом вони можуть представляти більшість природних і соціальних феноменів, які зустрічаються щодня. Точки, лінії і області можуть представлятися відповідними символами, поверхні же представляються або висотами точок, або контурами рельєфу або іншими комп'ютерними засобами. Картки призначені для того, щоб представляти не тільки об'єкти на її поверхні, але і форму Землі.

Глобус – традиційний спосіб відображення форми Землі. Картографи розробили набір методів, які називаються картографічними проекціями, які призначені для зображення з прийнятною точністю сферичної Землі на плоскому носії. Кожен з цих методів створює так зване сімейство проекцій. У ГІС найбільш широко поширеною системою проекцій і координат є універсальна

поперечна Меркатора. Вона використовується в більшості робіт з дистанційним зондуванням, підготовці топографічних карт, побудові баз даних природних ресурсів, так як вона забезпечує точні вимірювання в метричній системі, прийнятої в більшості країн і науковим співтовариством в цілому.

Для опису картографічної інформації недостатньо тільки метричних параметрів-координат. Тому для вказівки тематичних і часових характеристик застосовується атрибутивна інформація.

Атрибут – це елементарне дане, яке описує властивість будь-якого елементу моделі (об'єктами поняття). Атрибутами можуть бути символи (назви), числа (що відображають статистичні характеристики), графічні ознаки (колір, малюнок, графічна структура контуру тощо.). Зазвичай атрибути групують у вигляді спеціальних таблиць, що вельми зручно для організації взаємопов'язаного координатного і атрибутивного описів. Це обумовлено тим, що саме в таблиці можуть зберігатися як координати об'єктів (координатні дані), так і описові характеристики-атрибути. За допомогою атрибутів можна впорядковувати та типізувати дані, проводити аналіз баз даних з використанням різних алгоритмів. Таблиці виробляють суворе ранжування параметрів, що визначають різні ознаки об'єктів, оскільки кожному об'єкту відповідає рядок в таблиці, а кожному тематичною ознакою відводиться свій стовпець.

Точність обчислення в ГІС може бути дуже висока, тобто начно перевершувати точність самих даних. Тому важливе значення має приділятися отриманню первинних, вихідних даних – саме вони, перш за все, вимагають повноти та достовірності [4].

1.3 Процес розробки ГІС

Розробка ГІС складається з шести етапів (рис.1.5):

- аналіз вимог, що пред'являються до ГІС;
- визначення специфікацій;
- проектування системи;

- кодування;
- тестування;
- експлуатація та обслуговування.

Рисунок 1.5 – Затрати часу на реалізацію основних етапів розробки ГІС

Геоінформаційні системи являють собою системи керування базами даних (СКБД). Але є одна важлива відмінність – в ГІС спільно з атрибутивними даними обробляється і просторова (географічна) інформація. Тому при проектуванні ГІС фахівці використовують ті ж самі методики і техніки, що і при розробці звичайних СКБД.

Будь-яка БД містить інформацію про певну предметну область. Предметною областю називається певна сфера реального світу, яка становить інтерес для вивчення [5].

Перший етап проектування – це формалізація завдання, тобто на цьому етапі будують інфологічну модель предметної області. Створення інфологічної моделі включає в себе дослідження інформаційних потоків, характерних для конкретної предметної області, встановлення об'єктів і опис зв'язків, існуючих між ними.

Інфологічну модель використовують в якості фундаменту для будівництва датологіческой моделі БД, яка відображає логічні зв'язки між елементами даних незалежно від їх змісту і середовища зберігання. На даному етапі

необхідно враховувати різні обмеження, які накладають ПЗ, на структуру і функціональні особливості [6].

На наступному етапі створюється фізична модель бази даних, яка пов'язує датологіческой модель з конкретною середовищем зберігання. Це важливий етап, так як на ньому ведеться розробка елементів призначеного для користувача інтерфейсу, вирішуються питання цілісності даних і надійності системи, розподіляються права доступу і вибираються засоби і методи захисту від нелегального доступу.

Проектуючи географічні інформаційні системи, крім вищесказаного необхідно виконати наступні дії [6]:

- виробити вимоги, що стосуються вихідного картографічного матеріалу (потрібний масштаб, проекція, система координат);
- визначити розмірність географічних даних, з якими доведеться працювати (двовимірні 2D і / або тривимірні 3D), а також встановити модель представлення просторових даних (векторна і / або растрова);
- спроектувати пошаровий склад просторової інформації ГІС;
- встановити наявність цифрових карт потрібних територій.

Розглянемо деякі питання етапу кодування програмного забезпечення.

Програмне забезпечення (ПЗ, software) – сукупність програм системи і програмних документів, необхідних при експлуатації цих програм.

ПЗ ГІС підтримує той чи інший набір функціональних можливостей ГІС і включає спеціалізовані програмні засоби, такі як [6]:

- універсальні повнофункціональні ГІС (full GIS);
- інструментальні ГІС (GIS softwaretools);
- картографічні візуалізатори (mapviewer);
- картографічні браузери (mapbrowser);
- кошти настільного картографування (desktopmapping);
- інформаційно-довідкові системи (help-desksystem).

Крім того, існують спеціальні програмні засоби, обслуговуючі окремі функціональні групи:

- конвертування форматів;
- оцифровку;
- векторизацію;
- створення і обробку цифрових моделей рельєфу;
- взаємодія з системами супутникового позиціювання.

Комплект поставки програмного забезпечення ГІС може включати окремі функціональні модулі, які купуються і використовуються в наборі, що забезпечує вирішення завдань. У комплексі з ПЗ ГІС використовуються такі програмні продукти як:

- настільні видавничі пакети (AdobePageMaker, QuarkXpress, AdobeInDesign);
- пакети статистичного аналізу (Statistica);
- системи управління базами даних (MS Access, Oracle, DBase);
- системи автоматизованого проектування (AutoCAD);
- електронні таблиці (MS Excel);
- засоби цифрової обробки зображень (AdobePhotoshop).

1.4 Аналіз сучасних ГІС-платформ

В даний час в світі налічується велика кількість як комерційних, так і вільно розповсюджуваних ГІС-платформ. Розглянемо ряд найбільш використовуваних і популярних ГІС-платформ.

Платформа на базі ArcGIS- ESRI (США).

ESRI, Inc (Каліфорнія, США) – заснована в 1969 році, одна з найстаріших фірм в області розробки програмного забезпечення для ГІС (рис.1.6).

За допомогою програм ArcGIS можна побудувати ГІС першої, третьої і четвертої технологічних схем роботи з просторовими даними. Є можливість роботи як з власним форматом фалів даних, так і з сховищами просторових даних під керуванням SQL серверів Oracle, Microsoft, IBM DB2, PostgreSQL, Informix та ін. При цьому можливо як створення бази просторових даних у

внутрішньому форматі з використанням засобів ArcSDE (третє покоління), так і робота з власними розширеннями для зберігання просторових даних (четверте покоління) [7].

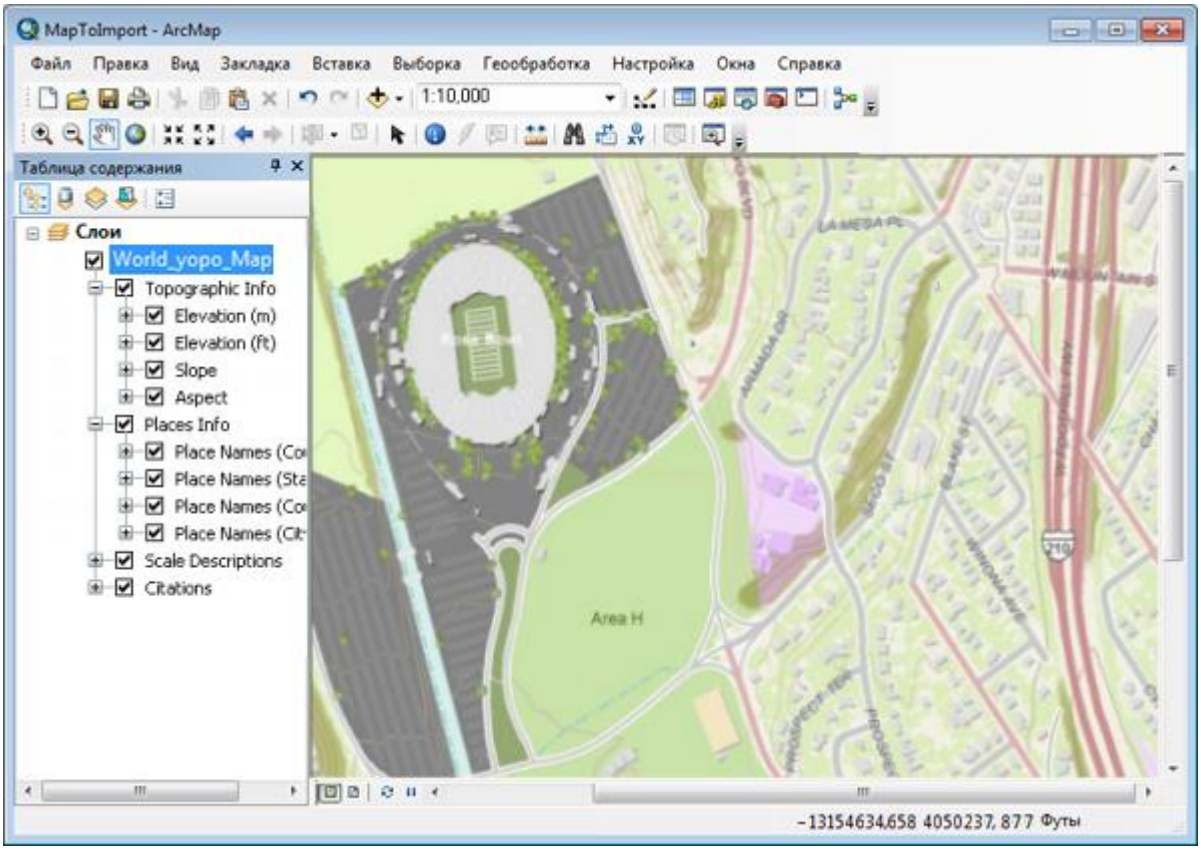

Рисунок 1.6 –Інтерфейс програми ArcMap платформи ArcGIS Desktop

Одна з небагатьох ГІС платформ, яка підтримує повноцінну роботу з топологічної моделлю представлення даних (вузлова і цепочноузловая моделі), а також зберігання, обробку та візуалізацію тривимірного представлення просторових даних. ArcGIS має гарну технічну підтримку різними мовами [8].

ArcGIS Pro – найсучасніший професійний настільний ГІС-застосунок від компанії Esri. У ArcGIS Pro можна вивчати, візуалізувати і аналізувати дані; створювати 2D-і 3D сцени; публікувати роботу на ArcGIS Online або на порталі ArcGIS Enterprise.

З недоліків платформи слід відмітити те, що вона є комерційним продуктом і для використання потребує наявності ліцензії.

Платформа MapINFO (США).

Вона дозволяє вирішувати всі основні завдання просторового аналізу статистичних і наукових даних в різних областях, проста в освоєнні і використанні, маючи при цьому більш низьку ціну ніж професійні ГІС, завдяки чому набула широкого поширення за кордоном.

MapInfo Pro підтримує всі поширені формати даних, включаючи офісні формати, такі як Microsoft Excel, Access, формати реляційних і просторових баз даних (Oracle, Microsoft SQL Server, PostGIS, SQLite), формати графічних даних (AutoCAD DXF / DWG, SHP, DGN) і багато інших (рис.1.7). Є можливість використовувати в роботі зображення практично будь-яких форматів (аерофотознімки, супутникові багатозональні знімки, скановані паперові карти і ін.). Крім того, MapInfo Pro має доступ до гібридних картах і знімкам Microsoft Bing.

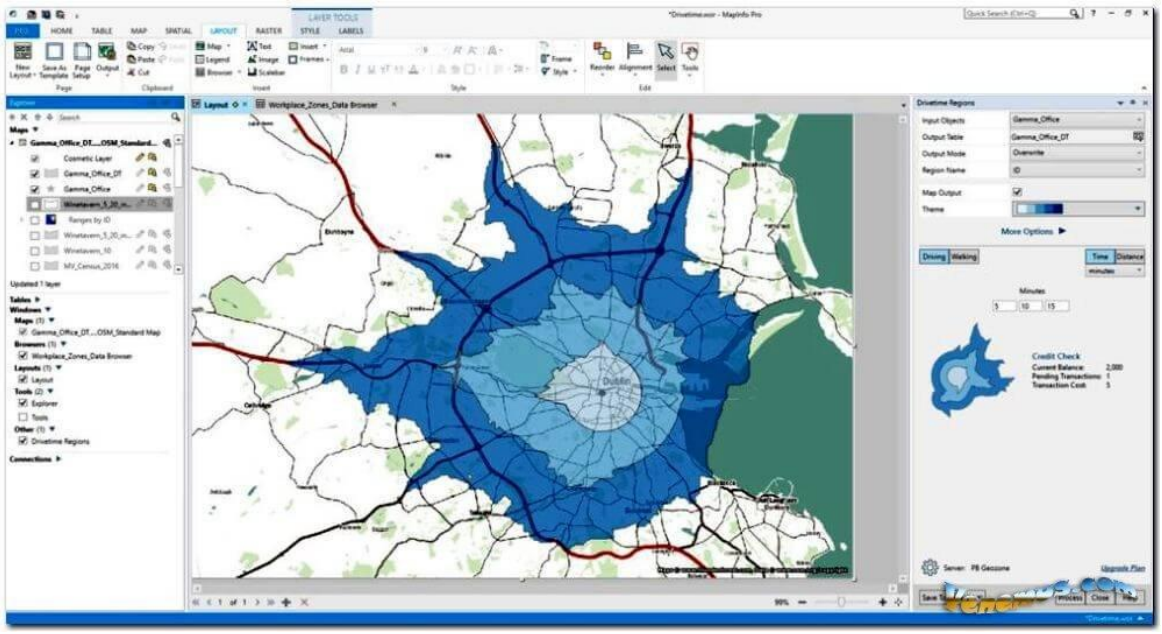

Рисунок 1.7 –Інтерфейс MapInfo Pro

Широкому поширенню також сприяли простий текстовий обмінний формат просторових даних, з практично усіма ГІС і CAD системами, а також наявність вбудованої системи розробки власних розширень на мові MapBasic, який був розширенням дуже популярного мови Basic, що використовується в більшості продуктів фірми Microsoft [9].

Відкрита ГІС QGIS.

Робота над QGIS була розпочата в травні 2002 року, а в 2007 році стала проектом Open Source Geospetial Foundation (OSGeo). Це настільна ГІС, ой динамічно розвивається основними перевагами якої є:

- безкоштовне розповсюдження;
- відкритий вихідний код;
- динамічний розвиток (нова версія виходить тричі на рік);
- широка документація;
- гнучкість взаємодії з різними апаратними базами, операційними системами і програмним забезпеченням (може бути встановлена на Windows, MacOS, Linux, Android).

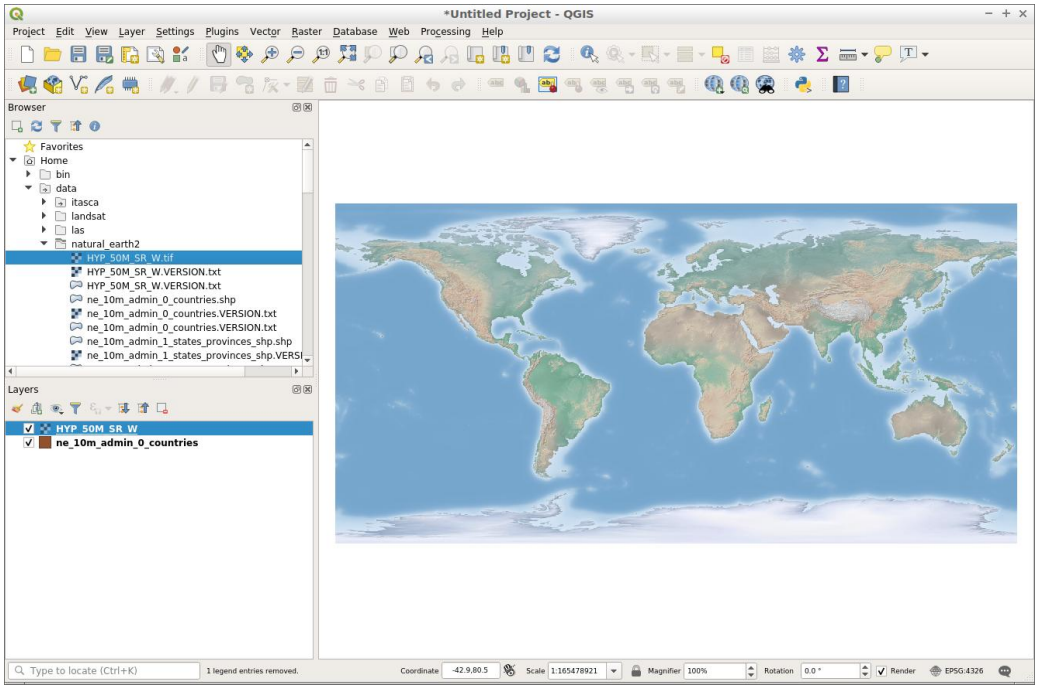

Рисунок 1.8 –Інтерфейс QGIS

Спочатку QGIS замислювалася, як переглядач просторових таблиць PostGIS, але з часом перетворилася на повну інструментальну ГІС, яка здатна вирішувати широкий спектр завдань.

Порівняльний аналіз розглянутих ГІС-платформ представлений а табл.1.1.

Таблиця 1.1 – Порівняльний аналіз ГІС-платформ

Критерій	ArcGIS	MapInfo	QGIS
Наявність архітектури клієнт/сервер	+	+	+
«Відкритість» архітектури	–	–	+
Повнофункціональність	+	+	+
Статистичний аналіз (наприклад, побудова графіків та діаграм)	+	+	–
Збереження картографічної та семантичної інформації на SQL-сервері	+	+	+
Можливість вбудування ГІС-ядра у зовнішні програмні системи	–	+	+
Публікація карт в Інтернеті	+	+	+
Вбудована мова для створення інструментів геообробки	Python	MapBasic	Python

Проведений аналіз програмних платформ, призначених для побудови геоінформаційних систем, дозволив зробити висновок, що для всіх подібних платформ характерна наявність повноцінної роботи з різними картографічними проекціями і системами координат. Забезпечена підтримка більшості популярних векторних і растрових форматів просторових даних, робота з тривимірним поданням просторових даних.

Велика кількість різних вбудованих прикладних і розрахункових підпрограм дозволяє обробляти просторові дані платформи. Підтримується велика кількість програмно-апаратних платформ. Платформи активно розвиваються, при цьому в розвитку різних компонентів бере участь безліч висококваліфікованих фахівців з усього світу. Можливе використання вільно

поширюваного ПО ГІС-платформ, що мінімізує витрати на придбання ліцензій на програмні компоненти платформи.

В кваліфікаційній роботі буде виконано розробку інструменту пошуку оптимального маршруту на основі цифрових моделей висот для ГІС-платформи ArcGIS. Вибір даної ГІС обумовлений гарною технічною підтримкою, наявністю докладного опису модулів та бібліотек для розробки власних інструментів геообробки просторових даних. Також на вибір вплинули особисті знання цієї платформи і досвід роботи, а також наявність ліцензії на використання ArcGIS в університеті.

2 ВИБІР ПРОГРАМНИХ ЗАСОБІВ РОЗРОБКИ ІНСТРУМЕНТІВ ГЕООБРОБКИ ПРОСТОРОВОЇ ІНФОРМАЦІЇ

Геообробка – це обробка географічної інформації, одна з основних функцій географічної інформаційної системи. Вона дає можливість створення нової інформації шляхом виконання операцій над існуючими даними. Будь-яка зміна або вилучення інформації при роботі з даними, є рішенням завдань геообробки. Це може бути, наприклад, завдання конвертації географічних даних в інший формат, або завдання, яке може полягати в послідовному виконанні декількох операцій, таких як вирізання, вибірка і подальший перетин наборів даних.

В ArcGIS є можливість виконувати завдання геообробки декількома способами, а саме [11]:

− запустити інструмент, скориставшись його діалоговим вікном;

− запустити інструменти з командного рядка;

− побудувати модель, послідовно використовуючи ряд інструментів геообробки, і запустити її;

− створити і запустити скрипт, який ініціює операції інструментів геообробки.

Кожен із системних інструментів, який встановлюється разом з ArcGIS, призначений для виконання однієї операції з географічними даними. Як правило, ці інструменти слід запускати послідовно, тобто результати роботи одного інструменту є вхідними даними для іншого інструмента. Перетворивши спеціалізовані моделі і скрипти у власні інструменти, з'являється можливість використовувати їх поряд з іншими системними інструментами при створенні послідовності завдань. Можна створити власну бібліотеку інструментів, яка будить вирішувати важливі для всього ГІС проекту індивідуальні завдання.

2.1 Створення інструментів за допомогою ModelBuilder

Конструктор моделей ArcGIS (Model Builder) – потужний засіб, що дозволяє, за допомогою побудови відповідних моделей більш ефективно управляти процесами обробки даних в ArcGIS. Елементом моделі може бути один з інструментів з ArcToolbox, скрипт на мові Python, VBScript та ін.

Модель в ModelBuilder – це відображення опису робочих процесів, які з'єднані один з одним, в послідовності інструментів геообробки, подаючи вихід одного інструменту в інший інструмент в якості входу. ModelBuilder можна також розглядати як візуальну мову програмування для побудови робочих потоків.

Основні переваги ModelBuilder [12]:

− ModelBuilder – це зручний в роботі додаток для створення і запуску робочих потоків, що містять послідовність інструментів;

− за допомогою ModelBuilder можна створювати власні інструменти. Інструменти, створені за допомогою ModelBuilder, можуть використовуватися в засобах підтримки скриптів Python і в інших моделях;

− ModelBuilder, поряд із засобами підтримки скриптів, надає можливість інтеграції ArcGIS з іншими додатками.

Вікно ModelBuilder складається з вікна відображення, в якому будується блок-схема моделі, головного меню і панелі інструментів, яку можна використовувати для роботи з елементами у блок-схемі моделі (рис. 2.1).

ModelBuilder має три набори параметрів, що використовуються в моделі:

1) Властивості моделі – ці властивості, які дозволяють задавати назву моделі, напис, опис, відносний шлях, властивості параметрів, змінні середовища моделі, довідку і кількість проходів.

2) Властивості діаграми – це властивості, що дозволяють змінювати розташування елементів на діаграмі, а також її колір і стиль.

3) Властивості відображення – це властивості, що дозволяють змінювати зовнішній вигляд і інші графічні властивості окремих елементів.

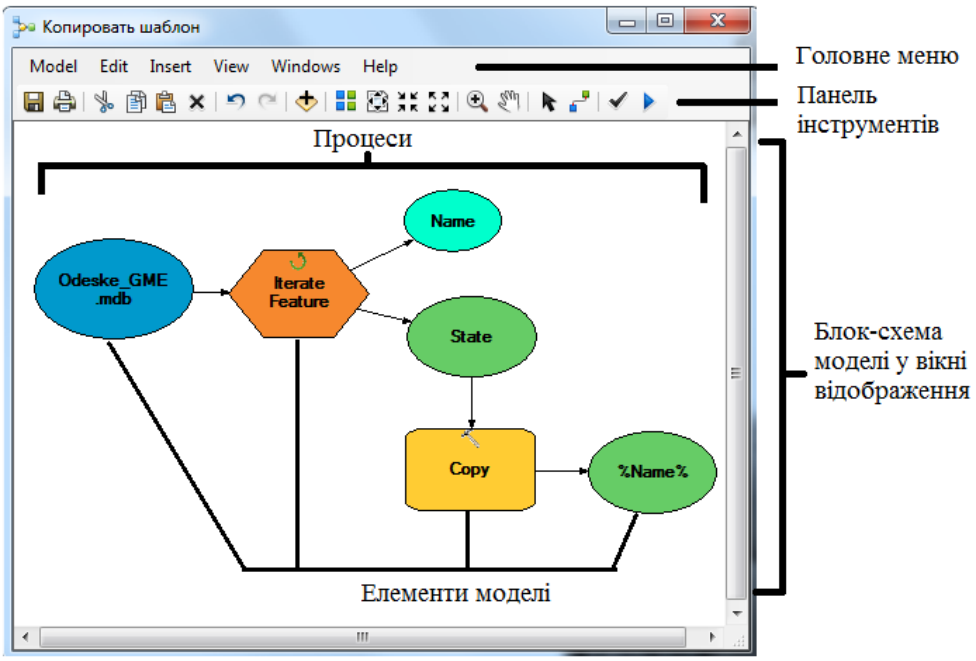

Рисунок 2.1 – Вигляд вікна ModelBuilder

Процес моделі складається з інструментів і всіх змінних, з'єднаних з ним (рис. 2.1). З'єднувачі показують послідовність виконання. У моделі часто буває кілька процесів, і вони можуть виконуватися по ланцюжку, так що вихідні дані одного процесу стають вхідними даними іншого. Кожен процес в моделі знаходиться в одному з чотирьох станів: «Не готовий до виконання (Not ready-to-run)», «Готовий до виконання (Ready-to-run)», «Виконується (Running)» або «Було виконано (Has-been-run)».

Моделі можна запускати з ModelBuilder або з діалогового вікна інструменту моделі, вікна Python, або в автономному скрипті. При запуску процесів з ModelBuilder їх виконання можна відстежувати в діалоговому вікні ходу геообробки.

ModelBuilder дозволяє створювати прості або складні моделі за допомогою ряду методів, які дозволяють ефективно управляти логічною послідо-

вністю виконання моделі, її даними, а також швидше здійснювати завдання, що повторюються. В групі інструментів Ітератори (Iterator) ModelBuilder містить дванадцять ітераторів, які повторюють процес або набір процесів для наборів вхідних значень (табл.2.1).

Таблиця 2.1 – Опис групи інструментів Ітератори (Iterator)

Ітератор	Опис
For	Виконує ітерації з початкового до кінцевого значення даної змінної. Він працює так само, як і оператор For в будь-якій мові програмування, виконуючи ітерацію задану кількість разів.
While	Виконується, поки умова є істиною.
Iterate Feature Selection	Виконує ітерації над об'єктами в класі просторових об'єктів.
Iterate Row Selection	Виконує ітерації для рядків в таблиці.
Iterate Field Values	Виконує ітерації для кожного значення в поле.
Iterate Multivalue	Виконує ітерації для списку значень
Iterate Datasets	Виконує ітерації для кожного набору даних в Робочій області (Workspace) або в Наборі класів об'єктів (Feature Dataset).
Iterate Feature Classes	Виконує ітерації для класів просторових об'єктів в робочій області або в наборі класів об'єктів.
Iterate Files	Виконує ітерації над файлами в папці.
Iterate Rasters	Виконує ітерації для растрів в Робочій області (Workspace) або в Каталозі растрів (Raster Catalog).
Iterate Tables	Виконує ітерації для таблиць в робочій області.
Iterate Workspaces	Виконує ітерації з робочими областями в папці.

Існує сім інструментів геообробки, які підтримують додаткові функції в ModelBuilder (табл.2.2). Ці інструменти можна використовувати в діалогових вікнах і скриптах.

Таблиця 2.2 – Опис інструментів тільки моделей (Model Only)

Інструменти	Опис
Calculate Value	Інструмент видає результуюче значення на основі зазначеного виразу Python
Collect Values	Інструмент розроблений для того, щоб збирати вихідні дані ітератора або конвертувати список декількох значень в один набір вихідних даних. Вихідні дані Collect Values можна використовувати як вхідні дані для таких інструментів як Злиття, Додати і Мозаїка
Get Field Value	Інструмент отримує значення першого рядка заданого поля з таблиці
Merge Branch	Інструмент об'єднує два або більше логічних гілок в один вихід
Parse Path	Інструмент отримує з набору вхідних даних його файл, шлях, тільки ім'я файлу і тільки розширення. Вихідні дані можуть використовуватися як вбудовані змінні в імені вихідних даних інших інструментів.
Select Data	Інструмент вибирає дані в батьківському елементі даних, наприклад, папці, базі геоданих, наборі класів об'єктів або покритті.
Stop	Виводить модель з циклу ітерації, якщо вхідний значення встановлено в True або False. Він функціонально схожий на ітератор While, але корисний для зупинки моделі в тому випадку, якщо в моделі є один ітератор While і не можна додати додаткові ітератори.

Інструменти моделі повністю інтегровані в середовище геообробки і працювати з ними можна так само, як з системними інструментами або інструментами скриптів. Як і всі інші інструменти геообробки, інструменти моделей можна запускати з діалогового вікна, через скрипти Python, а також додавати і запускати їх в іншої моделі. Скрипти Python і зовнішні програми, що запускаються за допомогою скрипта, можна інтегрувати в модель так само, як і системні інструменти, що додаються і запускаються в моделі [13].

2.2 Створення інструментів геообробки за допомогою Python

Мова Python є незалежною, міжплатформною, відкритою мовою програмування, швидкою, потужною і легкою в освоєнні. Вона широко використовується і підтримується [14].

Python з'явився в ArcGIS версії 9.0. З тих пір він використовувався в якості однієї з мов для написання скриптів, що містять процеси геообработки; область його застосування продовжує розширюватися. Кожен випуск розширював можливості Python і робив його використання все більш зручним.

ESRI остаточно впровадив Python в ArcGIS та розглядає цю мову в якості основного засобу, який задовольняє всі потреби користувачів. Перелічимо деякі переваги Python:

− легкий у вивченні, він ідеально підходить для початківців, залишаючись при цьому відмінним засобом для досвідчених користувачів;

− відмінно масштабований, він підходить як для великих проектів, так і для маленьких одноразових програм, відомих як скрипти;

− портативність і міжплатформність;

− вбудованість (написання скриптів в ArcGIS);

− стабільна і впевнена робота;

− велике співтовариство користувачів.

Python поширюється на всю систему ArcGIS, перетворюючись в мову аналізу, перетворення даних, автоматизації картографічних процесів, і дозволяє збільшити продуктивність цих робіт.

Нижче наведені основні терміни, пов’язані з геообробкою за допомогою Python [14]:

− ArcPy – бібліотека ArcPy забезпечує доступ з Python до всіх інструментів геообробки, включаючи додаткові модулі, а також пропонує велику кількість корисних функцій і класів для роботи з даними ГІС. Використовуючи Python і ArcPy, можна розробляти велику кількість зручних програм для роботи з географічними даними.

− Модулі ArcPy – являє собою файл Python, що містить функції і класи. ArcPy підтримується різними модулями, включаючи модуль доступу до даних (arcpy.da), модуль картографії (arcpy.mapping), додатковий модуль ArcGIS Spatial Analyst (arcpy.sa) і додатковий модуль ArcGIS Network Analyst (arcpy.na).

− Класи ArcPy – класи можна використовувати для створення об'єктів, що їх називають екземплярами. Такі класи ArcPy як SpatialReference і Extent часто використовуються як ярлики для завдання параметрів для інструментів геообробки, які інакше довелося б ставити у вигляді складних рядків.

− Функції ArcPy – частина програми, що виконує певне завдання. Функція може включатися в більш велику програму. У ArcPy всі функції геообробки представлені у вигляді функцій, проте не всі функції є інструментами геообробки. Крім інструментів в ArcPy є кілька функцій для поліпшення робочих процесів геообробки з використанням Python. Функції (часто звані методами) можуть використовуватися для створення списків певних наборів даних, вилучення властивостей набору даних, перевірки імені таблиці перед її додаванням в базу геоданих, а також для виконання багатьох інших корисних завдань геообробки.

– Автономний скрипт Python – це виконуваний файл з розширенням .py, який можна запустити з командного рядка, з інтегрованої середовища розробки Python (IDE) або двічі клацнувши файл .py в провіднику Windows.

– Інструмент-скрипт Python – це скрипт Python, який доданий в набір інструментів геообробки. Після додавання в якості інструменту-скрипта, він починає вести себе як будь-який інший інструмент геообробки – його можна відкрити і запустити з діалогового вікна, використовувати у вікні Python або ModelBuilder, а також викликати з інших скриптів та інструментів-скриптів.

– Вікно Python. За допомогою вікна Python можна легко і швидко скористатися Python безпосередньо з ArcGIS для інтерактивного запуску інструментів геообробки і функцій, а також скористатися перевагами, які дає використання модулів і бібліотек Python. Це вікно також дозволяє користувачам вивчати Python. Вікно Python може використовуватися для виконання окремих рядків коду на Python з висновком повідомлень про результат в цьому ж вікні. Це корисно для експериментів з синтаксисом і роботи з кодом невеликої довжини, а також дає можливість тестування завдань поза скрипта.

– Надбудова Python (add-in) – призначена для користувача функціональність, яка була реалізована на мові Python, наприклад, набір інструментів на панелі інструментів, включених в додаток ArcGIS for Desktop, призначена для забезпечення додаткових функціональних можливостей із супроводу призначених для користувача завдань. Для ясності процесу розробки надбудов Python слід завантажити і використовувати майстер надбудов Python для оголошення типу настройки. Майстер згенерує всі необхідні для роботи надбудови файли.

– Набори інструментів Python – набори інструментів геообробки, створені повністю в Python. Набір інструментів Python і інструменти, що містяться в ньому, виглядають і діють так само, як інструменти та набори інструментів, створені будь-яким іншим способом. Набір інструментів Python (.pyt) являє собою файл ASCII, що визначає набір інструментів і один або кілька інструментів.

ArcPy – це пакет, який заснований на успішному модулі arcgisscripting. Його метою є створення основи для успішного і продуктивного виконання аналізу географічних даних, конвертації даних, управління даними і автоматизації карти в Python.

Додатки та скрипти ArcGIS написані з використанням ArcPy дозволяють отримати доступ до численних модулів Python, розроблених користувачами ГІС і програмістами, що працюють в різних галузях. Ще одна перевага використання ArcPy в середовищі Python полягає в тому, що Python є універсальною мовою програмування, яка дозволяє працювати в режимі інтерпретації, що дає можливість швидко моделювати і перевіряти скрипти в інтерактивному середовищі, а також підтримує можливість написання великих додатків.

З технічної точки зору інструменти геообробки представляють собою функції, які можна викликати з arcpy, як і будь-яку іншу "рідну" функцію цього модуля. Проте, щоб уникнути плутанини між інструментами і відмінними від них функціями, такими як утиліта ListFeatureClasses(), існують чіткі відмінності:

− інструменти документуються інакше, ніж функції. У кожного інструменту є довідкова сторінка в довідковій системі ArcGIS Desktop. Функції документуються в документації ArcPy;

− інструменти на відміну від функцій повертають об'єкт result;

− інструменти створюють повідомлення, до яких можна звертатися за допомогою багатьох функцій, таких як GetMessages(). Функції не створюють повідомлень;

− інструменти ліцензуються на рівні продукту (ArcGIS for Desktop Basic, Standard або Advanced) або на рівні додаткових модулів ArcGIS Network Analyst, ArcGIS Spatial Analyst та ін. Функції не потребують ліцензування, вони встановлюються разом з ArcPy.

У наступному прикладі використовуються інструменти з наборів Data Management і Conversion. До вхідного класу об'єктів, який містить список ву-

лиць, додається поле, потім воно обчислюється, а клас об'єктів завантажується в базу геоданих ArcSDE.

```
>>> import arcpy
>>> Arcpy.AddField_management ("c:/data/Odesska.mdb/
streets", "LENGTH_MILES", "TEXT")
>>> Arcpy.CalculateField_management ("c:/data/Odesska.mdb/
streets", "LENGTH_MILES", "!shape.length@miles!", "PYTHON_9.3")
>>> Arcpy.FeatureClassToFeatureClass_conversion ("c:/data/
Odesska.mdb/streets", "Database Connections /MySDE.sde/
OdesskaDataset", "streets")
```

При виконанні інструменту геообробки результат його роботи повертається у вигляді об'єкта result. Зазвичай це шлях до вихідного набору даних, який був створений або оновлений за допомогою інструменту. В інших випадках це можуть бути інші типи значень, такі як число або логічне значення. Якщо вихідні дані є багатозначним параметром, значення можуть бути повернуті як список в списку.

У наступних прикладах коду показано, як захоплюються повернуті значення і якими вони можуть бути.

Повертається шлях вихідного класу просторових об'єктів. Результат можна використовувати як вхідні дані для іншої функції.

```
>>> Result = arcpy.Buffer_analysis ( "rivers", "riverBuf",
"50 METERS")
>>> Print result
C:\data\Odesska.mdb\riverBuf
>>> Arcpy.Clip_analysis("streets", result, "streets_50m")
```

Повертається кількість просторових об'єктів.

```
>>> Result = arcpy.GetCount_management( "streets_50m")
>>> Print result.getOutput (0)
54
```

Повертається список індексів просторової сітки за замовчуванням для класу просторових об'єктів.

```
>>> Result = arcpy.CalculateDefaultGridIndex_management(
"streets_50m")
>>> For i in range (0, result.outputCount):
... Print result.getOutput (i)
```

```
...
560
200
0
```

Параметри середовища геообробки можна розглядати як додаткові параметри, які впливають на результат роботи інструменту. Вони відрізняються від звичайних параметрів інструментів тим, що вони задаються окремо від інструменту і використовуються інструментом під час його роботи. Такі параметри середовища, як область інтересу, система координат вихідного набору даних і розмір комірки нового набору растрових даних, можна задати заздалегідь, потім вони будуть використані під час роботи інструменту.

Параметри середовища доступні у вигляді властивостей класу env. Ці властивості можна використовувати для отримання поточних значень параметрів середовища і для їх завдання. Нижче наведені приклади використання значень параметрів середовища.

Задаються параметри середовища робочої області.

```
>>> Arcpy.env.workspace = "c:/data/Odesska.mdb"
>>> Arcpy.Buffer_analysis( "streets", "streetBuf", "500
METERS")
```

Задається індекс просторової сітки для значення, що повертається інструментом.

```
>>> Arcpy.env.spatialGrid1 =
arcpy.CalculateDefaultSpatialGridIndex_management("streets").
GetOutput (0)
```

Повертаються поточні настройки розміру комірки растра і перевіряється, чи використовується це значення в якості вихідного.

```
if arcpy.env.cellSize! = 30:
arcpy.env.cellSize = 30
```

Крім інструментів ArcPy містить велику кількість функцій для поліпшення підтримки процесу геообработки. Функції можна використовувати для створення списків певних наборів даних, отримання властивостей наборів даних, перевірки наявності даних, перевірки імен таблиць перед додаванням

їх в базу геоданих або для виконання безлічі інших завдань, що вирішуються за допомогою скриптів.

У наступному прикладі коду показано отримання властивостей даних і перевірка додаткового модуля.

```
import arcpy
# Prints True
print arcpy.Exists("c: /data/Odesska.mdb/streets")
# Prints NAD_1983_StatePlane_Oregon_North_FIPS_3601_Feet
sr = arcpy.Describe("c: /data/Odesska.mdb/streets").
spatialReference
print sr.name
# Prints Available
print arcpy.CheckExtension ( "spatial")
arcpy.CheckOutExtension ( "spatial")
```

3 РОЗРОБКА ІНСТРУМЕНТУ ПОШУКУ ОПТИМАЛЬНОГО МАРШРУТУ НА ОСНОВІ ЦМР

Розрахунок маршрутів руху є геопросторовою функцією, що найбільш часто використовується користувачами. Як правило, ці алгоритми обчислюють найкоротший шлях між точками А і В, вони також можуть враховувати обмеження швидкості дороги або навіть поточні умови руху, щоб вибрати маршрут оптимальний за часом.

Однак, якщо необхідно побудувати нову дорогу, прокласти маршрут для пошуково-рятувального загону або лінії електропередачі чи трубопровід через віддалений район, то найкоротший шлях на місцевості може перетнути гору або пройти через озеро. У цьому випадку потрібно враховувати перешкоди та уникати їх, якщо це можливо. Але якщо незначна перешкода відводить нас занадто далеко від маршруту, то вартість його реалізації може бути дорожче, ніж просто переїзд через гору.

Мета моделювання в цьому випадку полягає в аналізі тільки найбільш важливих просторових умов і факторів, що призводять до істотного збільшення довжини маршруту і ігнорування менш значних умов, які аналізуються на стадії проектування. При такій постановці можна використовувати алгоритми побудови на картах шляхів мінімальної вартості, в англомовній літературі отримали назву least-cost path analysis (LCPA). Алгоритм дозволяє знайти маршрут, який є найкращим компромісом відстані порівняно з вартістю слідування цим маршрутом.

В основі методів вирішення завдань LCPA лежить теорія алгоритмів пошуку оптимальних шляхів на графах. В літературі вона отримала назву All-Pairs Shortest Pathproblem (APSP) або Pathfinding. Завдання пошуку найкоротшого шляху між двома вершинами графа складається в знаходженні ланцюга з джерела в стік, що мінімізує вартість проходження потоку заданої величини по даному ланцюгу. Загальна постановка задачі вперше приведена в [15]. Для вирішення цього завдання розроблено безліч алгоритмів. Найбільшу популя-

рність здобули алгоритми, які можна розділити на евристичні (беруть початок від широко відомого алгоритму A *, A-star [16]) і алгоритми пошуку в ширину (алгоритм Дейкстри (Dijkstra's algorithm) [17]). Алгоритмом Дейкстри обчислює найкоротший шлях в мережі, наприклад, дорожній мережі. Метод A* може це зробити, але він краще підходить для переміщення по матричній моделі висот у вигляді сітки.

При пошуку маршрутів на карті довільної пересіченої місцевості в якості графа зазвичай використовують регулярну прямокутну сітку – растрову модель території (DEM), в якій задана можливість переходу в сусідні комірки по вертикалі, горизонталі та діагоналях (рис. 3.1). Деякі комірки можуть бути позначені як непрохідні (на рис. 3.1 показані темним кольором). Даний підхід, зокрема, поширений в комп'ютерних іграх.

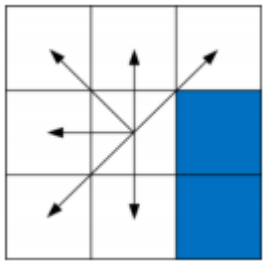

Рисунок 3.1 – Уявлення графа в растровій моделі просторових даних

3.1 Моделювання алгоритму пошуку оптимального маршруту на тестовій сітці мінімальної вартості

Виконаємо моделювання алгоритму на сітці 5x5 з згенерованими випадковими значеннями висот. Алгоритм створення тестової сітки висот в цьому випадку буде складатися з наступних кроків:

1) Створити сітку з випадковим значеннями псевдовисот від 1 до 16.
2) Визначити початкове місце в лівому нижньому куті сітки.
3) Визначити кінцеву точку у правому верхньому куті сітки.

4) Створіть сітку мінімальної вартості, яка містить висоту кожної комірки, плюс відстань комірки до фінішу.
5) Переглянути кожну сусідню клітинку та обрати ту, яка має найнижчу вартість.
6) Повторюйте оцінку, використовуючи вибрану поточну комірку, поки не дійдемо до кінця.
7) Повернути набір вибраних комірок як шлях із найменшими витратами.
8) Налаштувати тестову сітку.

Розглянемо скрипт, що створює сітку мінімальної вартості для який реалізації цього сценарія. Спочатку скрипт формує сітку штучного рельєфу як випадково сформований масив NumPy з умовними значеннями висот від 1 до 16. Далі створюється сітка відстаней, яка обчислює відстань від кожної комірки до комірки призначення. Сума висоти кожної комірки і її відстані до фінішу – вартість кожної комірки. Блок-схема даного алгоритму наведена на рис.3.2.

3.2 Моделювання алгоритму A* на тестовій сітці мінімальної вартості

Реалізуємо алгоритм пошуку A* для тестової сітці мінімальної вартості. Алгоритм заливки стартує із заданої комірки і починає перевірку сусідніх комірок. Сусідня комірка, що має найменшу вартість додається до маршруту і позначається для перевірки її сусіда, поки не буде перевірена вся регулярна сітка (grid). Тобто при проході через кожну комірку виконується перевірка її сусідів. Будь-яка з цих комірок, яку треба перевірити додається у множину open_set(), вже перевірені комірки – у множину closet_set(), комірки, що пройшли перевірку і можуть бути додані до маршруту – у множину path(). Множини використовуються, щоб уникнути рекурсії та дублювання перевірок комірок.

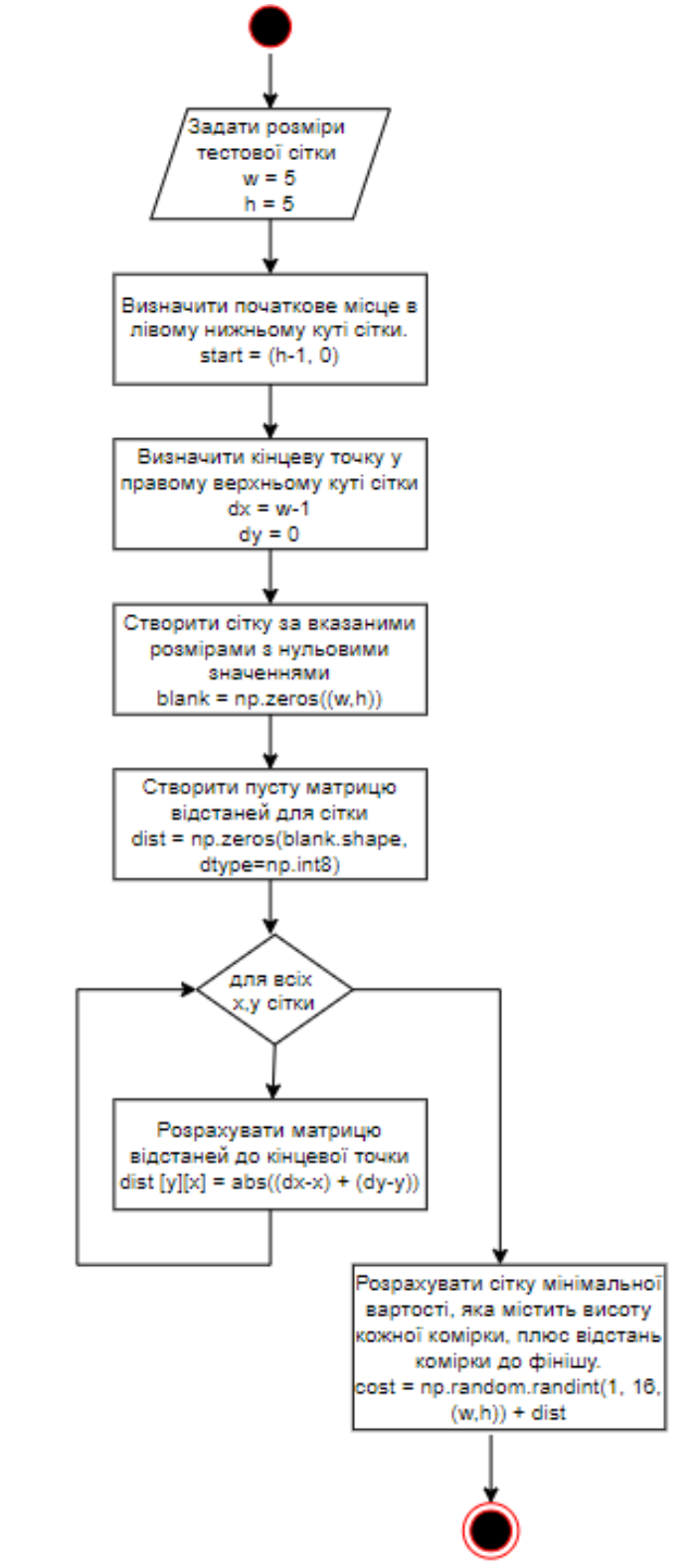

Рисунок 3.2 – Блок-схема побудови тестової сітки мінімальної вартості

Блок-схема алгоритму A* для тестової сітці мінімальної вартості наведена на рис.3.3.

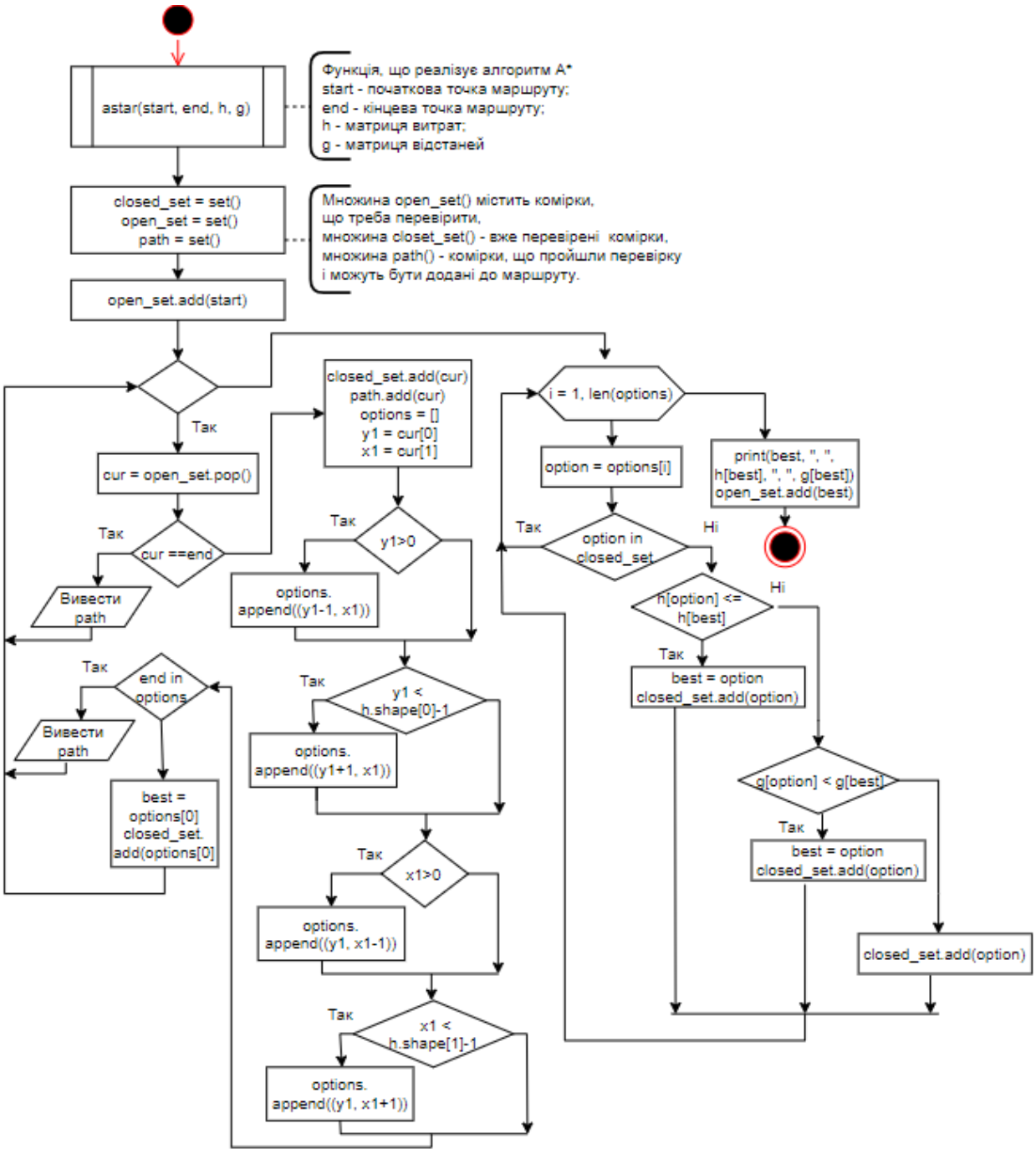

Рисунок 3.3 – Блок-схема алгоритму A* для тестової сітці мінімальної вартості

Перевірити результати моделювання на тестовій сітці можна шляхом запуску створеної функції, задав її вхідні параметри:

```
path = astar(start, (dy, dx), cost, dist)
```

Можна вивести числову згенеровану сітку, задавши точки знайденого оптимального маршруту як 1:

```
path_grid = np.zeros(cost.shape, dtype=np.uint8)
for y, x in path:
   path_grid[y][x] = 1
path_grid[dy][dx] = 1
print("PATH GRID: 1=path")
print(path_grid)
```

Результати чисельного моделювання наведені на рис. 3.4

У цій реалізації використовується Манхеттенська відстань, що означає, що відстань не використовує діагональні лінії – вимірювання лише ліворуч, праворуч, вгору та вниз. Пошук також не рухається по діагоналі.

```
COST GRID
[[13 10 5 15 9]
 [15 13 16 5 16]
 [17 8 9 9 17]
 [ 4 1 11 6 12]
 [ 2 7 7 11 8]]

(Y,X),
(3, 0) , 4 , 1
(3, 1) , 1 , 0
(2, 1) , 8 , 1
(2, 2) , 9 , 0
(2, 3) , 9 , 1
(1, 3) , 5 , 0
(0, 3) , 15 , 1

PATH GRID:
[[0 0 0 1 1]
 [0 0 0 1 0]
 [0 1 1 1 0]
 [1 1 0 0 0]
 [1 0 0 0 0]]
```

Рисунок 3.4 – Результати моделювання на тестовій сітці

3.3 Моделювання алгоритму A* на основі ЦМР

Виконаємо моделювання алгоритму A* з використанням цифрової моделі рельєфу. Це може бути здійснено за допомогою отриманих раніше скриптів.

Вигляд фрагменту ЦМР та разташованих на рельєфі точок початка та кінця маршруту у програмі ArcMap 10.5 наведено на рис.3.5. Мета моделювання – рухатися від початку до кінцевої точки з мінімальною вартістю.

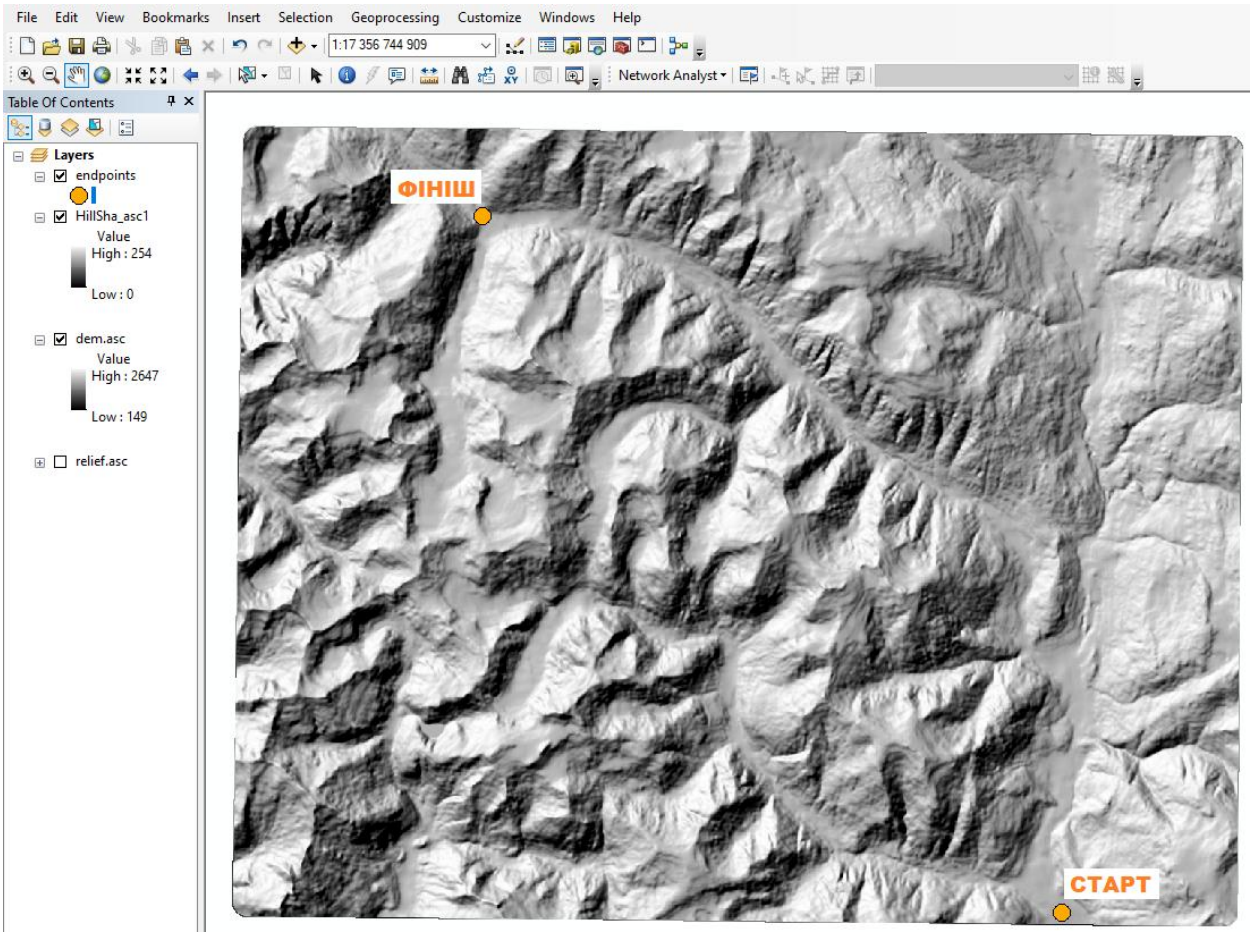

Рисунок 3.5 – ЦМР із заданими точками початку та кінця маршруту

Якщо дивитись лише на місцевість, є дві стежки, які йдуть по маршрутах з низькими висотами без особливих змін напрямку. Ці два маршрути проілюстровані на рис.3.6.

Можна припустити, що у випадку використання алгоритм A*, побудовані маршрути будуть близькими до маршрутів на рис.3.6. Але треба враховувати, що алгоритм дивиться лише в безпосередній близькості, тому він не може розглядати ціле зображення і вносити корективи на початку маршруту на основі відомої перешкоди, що існує попереду.

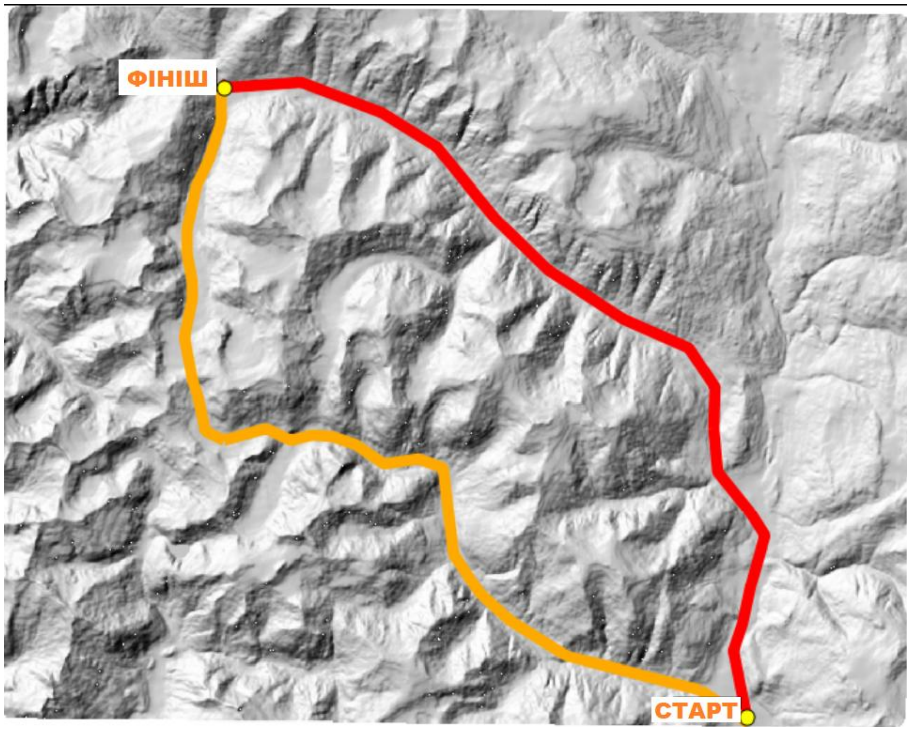

Рисунок 3.6 – Маршрути побудовані за низькими висотами ЦМР

В остаточній реалізації скрипта будемо використовувати Евклідову відстань, а також здійснювати пошук у восьми напрямках замість чотирьох. Результатом буде растр із значеннями шляху, де комірки, що належать маршруту, будуть мати значення 1, а інші значення – нуль.

Опишимо етапи створення скрипту пошуку оптимального маршруту на основі ЦМР.

Спочатку завантажується сітка в масив NumPy з сітки ASCII. Джерелом є файл ЦМР source = dem.asc. Формат файлу ASCII має заголовок, в якому зазначено кількість рядків і стовпців матриці, розмір комірки растра і інша інформація, далі йде числова матриця зі значенням висот скопійована з ЦМР.

Далі треба створити сітку target = path.asc в якій буде зберігатися знайдений шлях, а також задати початкову та кінцеву комірки маршруту.

Алгоритм завантаження матриці висот ЦМР представлений на рис.3.7.

Для роботи скрипта треба завантажити додаткові спеціальні бібліотеки:

```
import numpy as np
```

```
import math
from linecache import getline
import pickle
```

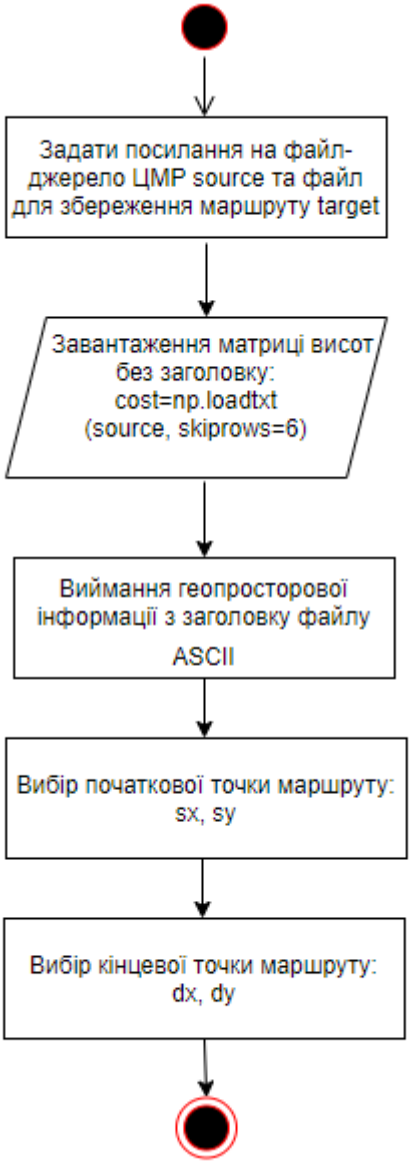

Рисунок 3.7 – Алгоритм завантаження матриці висот ЦМР

Скрипт завантажує сітку, пропускаючи заголовок:

```
cost = np.loadtxt(source, skiprows=6)
```

Далі витягується геопросторова інформація з заголовка файлу ASCII, яка додаткова аналізується:

```
hdr = [getline(source, i) for i in range(1, 7)]
values = [float(ln.split(" ")[-1].strip()) for ln in hdr]
```

```
cols, rows, lx, ly, cell, nd = values
```

Нарешті, визначаються початкове (sx, sy) та кінцеве (dx,dy) розташування точок маршруту.

Для інструменту геообробки були створені три функції для того, щоб рухатись по місцевості. Перше – це алгоритм A*, а два інших допоміжні алгоритми у виборі наступного кроку. Стисло розглянемо їх.

1) Функція розрахунку Евклідової відстані (e_dist), яка повертає пряму відстань між двома точками в одиницях карти:

```
def e_dist(p1, p2):
x1, y1 = p1
x2, y2 = p2
distance = math.sqrt((x1-x2)**2+(y1-y2)**2)
return int(distance)
```

2) Функція weighted_score, яка повертає зважену оцінку для сусідньої комірки на основі зміни висоти між сусідньою та поточною коміркою, а також відстані до пункту призначення, тобто вагова функція оцінює кожен вузол на придатність для пересування.

Ця функція зменшує ймовірність зв’язку між двома комірками, що полегшує уникнення зворотного відстеження. В оцінці враховуються: висота поточного вузла cur_h = h [cur]; відстань від поточного вузла до кінцевого cur_g = e_dist (cur, end); відстань від поточного вузла до початкового cur_d = e_dist (cur, start). Виходячі з отриманої інформації розраховується оцінка вузла. Алгоритм вагової функції наведено на рис.3.8.

За допомогою даних двох даткових функцій та на основі вдосконаленого алгоритму A* був створений інструмент розрахунку маршруту в ГІС, який наведено у додатку А.

Результати пошуку маршруту найменшої вартості для завантаженої ЦМР наведені на рис. 3.9. Результати пошук за алгоритмом A* був дуже близьким до одного з найдених вручну маршрутів на рис.3.6. У кількох випадках алгоритм вирішив пересікти певну місцевість, замість того, щоб нама-

гатись її об'їхати. Це добре можна побачити у наближеному фрагменті верхній правої ділянки маршруту (рис.3.10).

Скрипт використовує лише два значення: рельєф та відстань. Але є можливість додати сотні факторів, таких як тип грунту, водойми та існуючі дороги. Треба просто змінити функцію розрахунку оцінки, щоб врахувати будь-які додаткові фактори. Проте, чим більше факторів, тим складніше простежити, що враховувала реалізація A *, коли обирала маршрут.

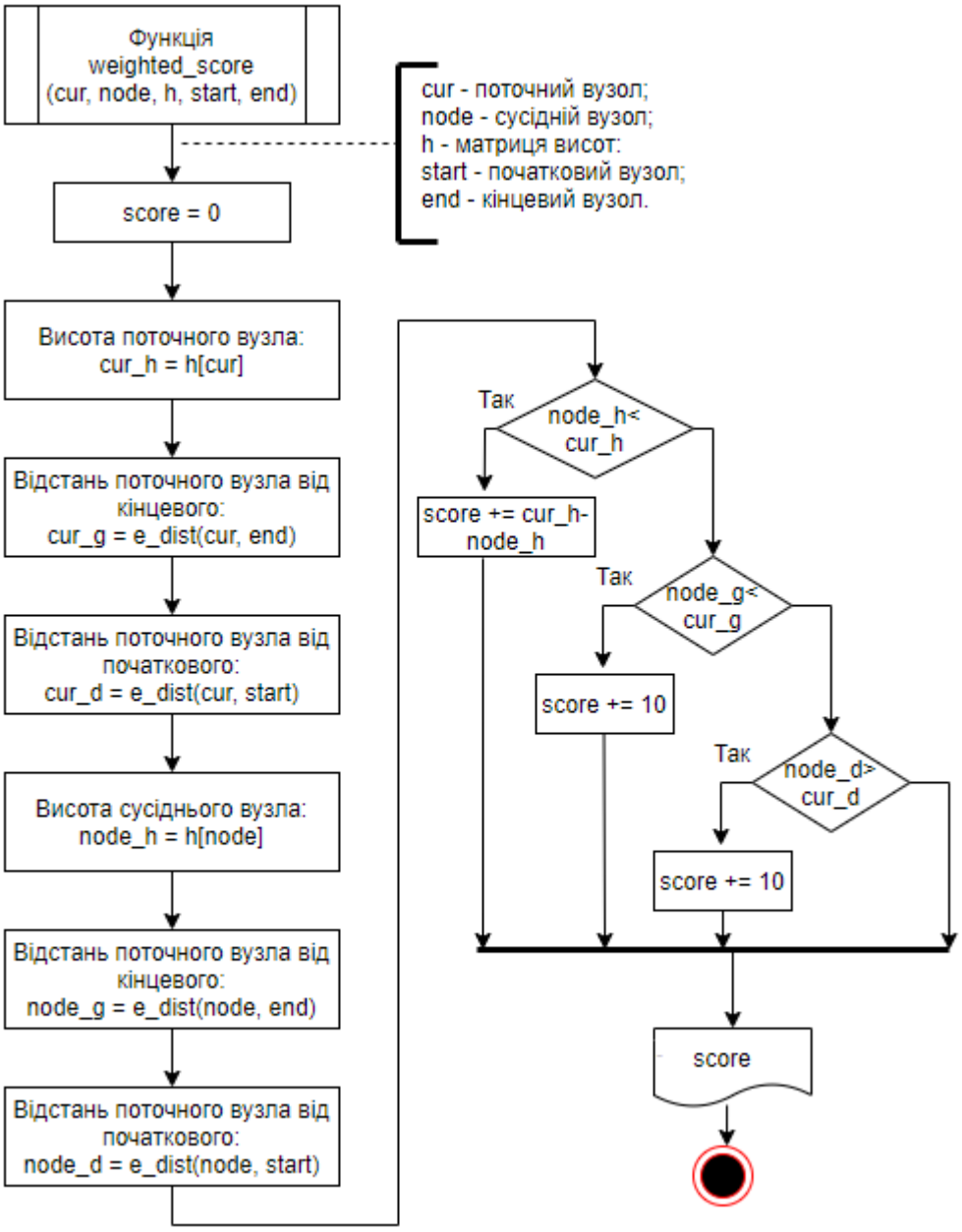

Рисунок 3.8 – Блок-схема алгоритму функції розрахунку вагової оцінки вузла маршруту

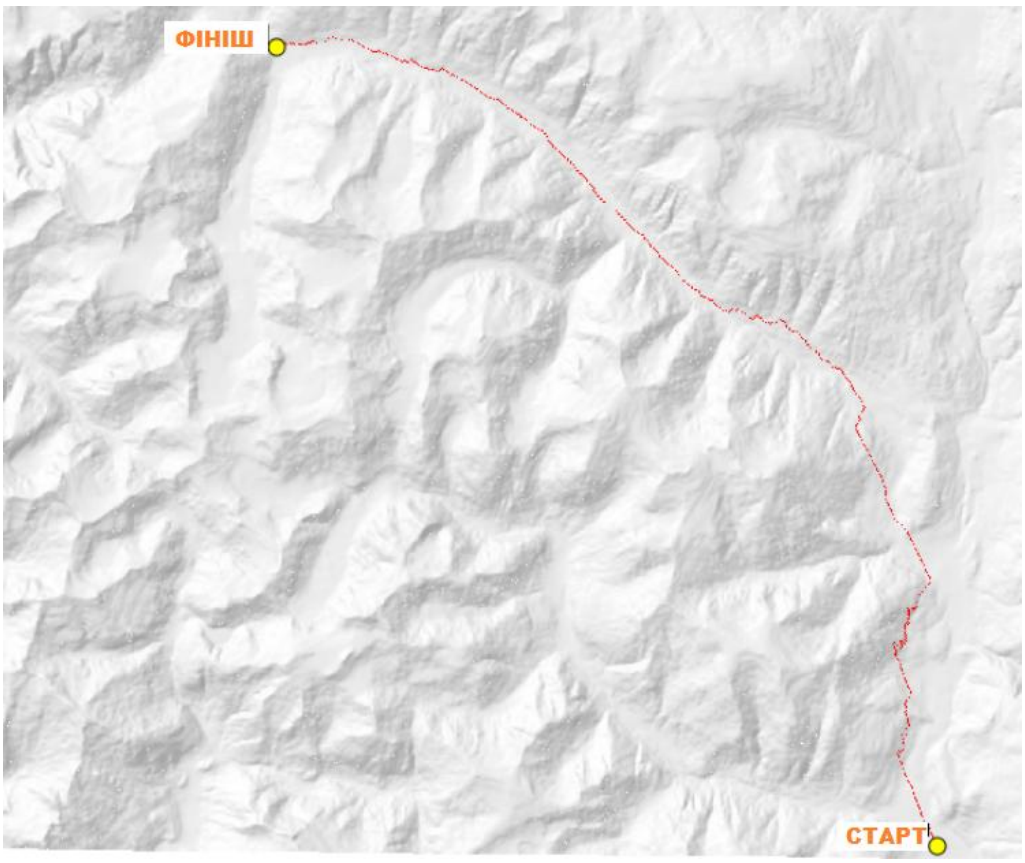

Рисунок 3.9 – Маршрут побудований для ЦМР за алгоритмом A*

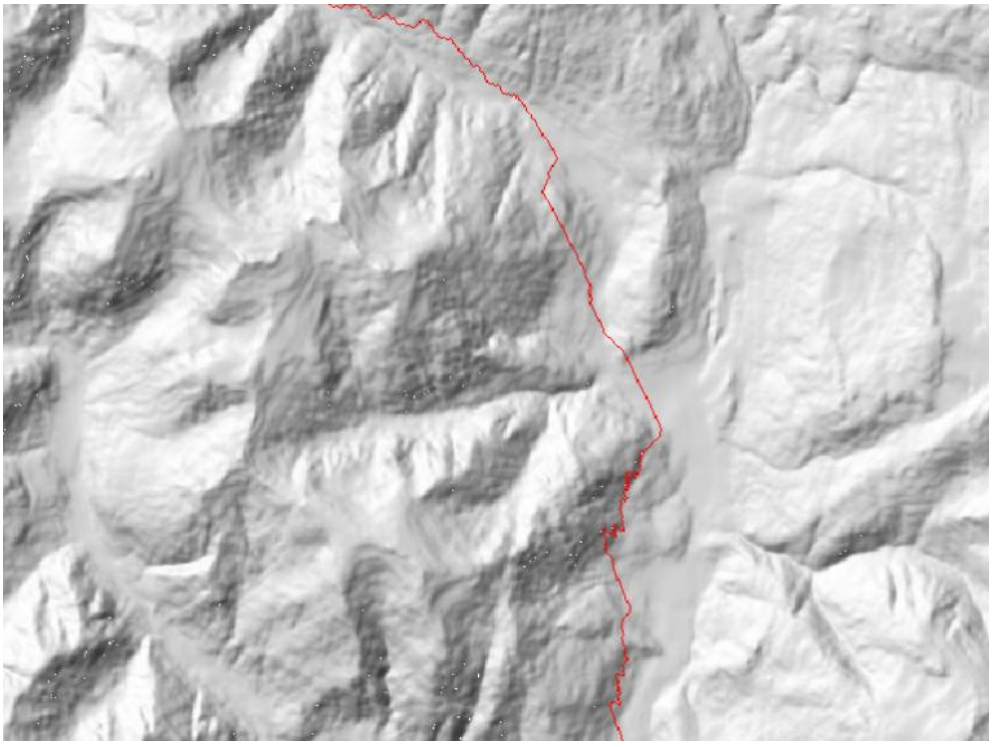

Рисунок 3.10 – Наближений фрагмент верхній правої ділянки маршруту

Майбутнім напрямком цього аналізу буде створення векторної версії цього маршруту як лінії. Процес включає відображення кожної комірки в точку, а потім використання аналізу найближчого сусіда для правильного впорядкування точок, перед тим як зберегти його як шейп-файл або файл GeoJSON.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи був проведений аналіз сучасних програмних засобів геообробки просторової інформації. Показано, що найбільш зручним засобом для роботи з векторними та растровими даним є створення власних скриптів Python, які можуть виконуватися автономно чи запускатися з середовища геоінформаційної системи.

В роботі був розроблений власний інструменти геообробки для пошуку оптимального маршруту на основі цифрових моделей висот в ГІС. Інструмент може бути корисним, якщо необхідно побудувати нову дорогу, прокласти маршрут для пошуково-рятувального загону або лінії електропередачі чи трубопровід через віддалений район.

Для розрахунку маршруту використовувався алгоритм A*. В якості вхідних даних – цифрові моделі рельєфу (файли DEM).

Розрахунок вартості вузлів маршруту відбувався за двома факторами: висота рельєфу і відстать. Але є можливість додати більше факторів, наприклад, тип грунту, водойми та існуючі дороги. Для цього треба змінити функцію розрахунку вартості, щоб врахувати додаткові фактори.

Геоінструмент написаний на мові програмування Python 3.0. Тестування скрипта відбувалося в пакеті ГІС ESRI ArcGIS 10.5 і показало, що він успішно реалізує вихідний алгоритм.

Майбутнім напрямком цього аналізу буде створення векторної версії маршруту. Процес включає відображення кожної комірки в точку, а потім використання аналізу найближчого сусіда для правильного впорядкування точок, перед тим як зберегти його як шейп-файл або файл GeoJSON.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Сербенюк С. Н. Картография и геоинформатика – их взаимодействие. М.: МГУ, 1990. 160 с.
2. Берлянт А. М. Геоинформационное картографирование. М .: МГУ, 1997. 64 с.
3. Цветков В. Я. Геоинформационные системы и технологии. М .: Финансы и статистика, 1998. 288 с.
4. Васильев, В. Н. Обзор существующих ГИС. Молодой ученый. 2016. № 14 (118). С. 62-66. URL: https://moluch.ru/archive/118/32677/ (дата звернення: 28.04.2021).
5. Шипулин В. Д. Основные принципы геоинформационных систем: учебное пособие .Харьковская национальная академия городского хозяйства.Х.: ХНАГХ, 2010. 337. с.
6. Воробьева, А.А. Учебное пособие по курсу геоинформационные системы территориального управления. Санкт-Петербург: Изд-во «Питер», 2012.
7. Офіційний сайт компанії ESRI Inc. URL: www.esri.com (дата звернення 21.04.2021)
8. Сайт справочної системи ArcGIS. URL: webhelp.esri.com/arcgisdesktop (дата звернення 21.04.2021)
9. MapInfo Professional 9.0. URL: http://base.dnsgb.com.ua/files/book/MapInfo-9.0.pdf (дата звернення 21.04.2021)
10. Офіційний сайт QGIS. URL: https://www.qgis.org/ru/site/ (дата звернення 21.04.2021)
11. Гурьянова Л.В. Аппаратно–программные средства ГИС: компьютерный практикум для студентов часть 2. Полоцк: ПГУ, 2011. 120 с.
12. David W. Allen Getting to Know ArcGIS ModelBuilder URL: http://esripress.esri.com/bookresources/modelbuilder.pdf (дата звернення 21.04.2021)

13. Офіційний сайт Python. URL: http://www.python.org (дата звернення 21.04.2021)
14. Tateosian L. Python for ArcGIS. Springer International Publishing Switzerland, 2015. 544 p.
15. Phillips D.T., Garsia-Diaz A. Fundamentals of Network Analysis. Prentice-Hall, Inc.: Englewood Cliffs, NJ, 1981.
16. Hart P.E., Nilsson N.J., Raphael B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths // IEEE Trans. Syst. Sci. Cybernet. 1968. No. 4. P. 100–107.
17. Dijkstra, E.W. A Note on Two Problems in Connexion with Graphs. Numerische Mathematik, 1959, no. 1, pp. 269– 271.

Додаток А

Програмний код

```
import numpy as np
import math
from linecache import getline
import pickle

# Our terrain data
source = "dem.asc"

# Output file name
# for the path raster
target = "path.asc"

print("Opening %s..." % source)
cost = np.loadtxt(source, skiprows=6)
print("Opened %s." % source)

# Parse the header
hdr = [getline(source, i) for i in range(1, 7)]
values = [float(ln.split(" ")[-1].strip()) for ln in hdr]
cols, rows, lx, ly, cell, nd = values

# Starting column, row
sx = 1006
sy = 954

# Ending column, row
dx = 303
dy = 109


def e_dist(p1, p2):

    x1, y1 = p1
    x2, y2 = p2
    distance = math.sqrt((x1-x2)**2+(y1-y2)**2)
    return int(distance)


def weighted_score(cur, node, h, start, end):

    score = 0
    # current node elevation
    cur_h = h[cur]
    # current node distance from end
    cur_g = e_dist(cur, end)
    # current node distance from start
    cur_d = e_dist(cur, start)
```

```
    # neighbor node elevation
    node_h = h[node]
    # neighbor node distance from end
    node_g = e_dist(node, end)
    # neighbor node distance from start
    node_d = e_dist(node, start)
    # Compare values with the heighest
    # weight given to terrain followed
    # by progress towards the goal.
    if node_h < cur_h:
        score += cur_h-node_h
    if node_g < cur_g:
        score += 10
    if node_d > cur_d:
        score += 8
    return score


def astar(start, end, h):

    # Closed set of nodes to avoid
    closed_set = set()
    # Open set of nodes to evaluate
    open_set = set()
    # Output set of path nodes
    path = []
    # Add the starting point to
    # to begin processing
    open_set.add(start)
    while open_set:
        # Grab the next node
        cur = open_set.pop()
        # Return if we're at the end
        if cur == end:
            return path
        # Close off this node to future
        # processing
        closed_set.add(cur)
        # The current node is always
        # a path node by definition
        path.append(cur)
        # List to hold neighboring
        # nodes for processing
        options = []
        # Grab all of the neighbors
        y1 = cur[0]
        x1 = cur[1]
        if y1 > 0:
            options.append((y1-1, x1))
        if y1 < h.shape[0]-1:
            options.append((y1+1, x1))
        if x1 > 0:
            options.append((y1, x1-1))
```

```
            if x1 < h.shape[1]-1:
                options.append((y1, x1+1))
            if x1 > 0 and y1 > 0:
                options.append((y1-1, x1-1))
            if y1 < h.shape[0]-1 and x1 < h.shape[1]-1:
                options.append((y1+1, x1+1))
            if y1 < h.shape[0]-1 and x1 > 0:
                options.append((y1+1, x1-1))
            if y1 > 0 and x1 < h.shape[1]-1:
                options.append((y1-1, x1+1))
            # If the end is a neighbor, return
            if end in options:
                return path
            # Store the best known node
            best = options[0]
            # Begin scoring neighbors
            best_score = weighted_score(cur, best, h, start, end)
            # process the other 7 neighbors
            for i in range(1, len(options)):
                option = options[i]
                # Make sure the node is new
                if option in closed_set:
                    continue
                else:
                    # Score the option and compare
                    # it to the best known
                    option_score = weighted_score(cur, option,
                                                  h, start, end)
                    if option_score > best_score:
                        best = option
                        best_score = option_score
                    else:
                        # If the node isn't better seal it off
                        closed_set.add(option)
                    # Uncomment this print statement to watch
                    # the path develop in real time:
                    # print(best, e_dist(best, end))
            # Add the best node to the open set
            open_set.add(best)
        return []

print("Searching for path...")
p = astar((sy, sx), (dy, dx), cost)
print("Path found.")
print("Creating path grid...")
path = np.zeros(cost.shape)
print("Plotting path...")
for y, x in p:
    path[y][x] = 1
path[dy][dx] = 1

print("Path plotted.")
```

```
print("Saving %s..." % target)
header = ""
for i in range(6):
    header += hdr[i]

# Open the output file, add the hdr, save the array
with open(target, "wb") as f:
    f.write(bytes(header, 'UTF-8'))
    np.savetxt(f, path, fmt="%4i")

print("Saving path data...")
with open("path.p", "wb") as pathFile:
    pickle.dump(p, pathFile)

print("Done!")
```