МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет
«Запорізька політехніка»

МЕТОДИЧНІ ВКАЗІВКИ

до виконання самостійної роботи
з дисципліни

“ Алгоритми та структури даних ”

для студентів спеціальностей
121 “Інженерія програмного забезпечення” та
122 “Комп’ютерні науки”
(всіх форм навчання)

2023

Методичні вказівки до виконання самостійної роботи з дисципліни “Алгоритми та структури даних” для студентів спеціальностей 121 “Інженерія програмного забезпечення” та 122 “Комп’ютерні науки” (всіх форм навчання) / В.М. Льовкін. – Запоріжжя : НУ «Запорізька політехніка», 2023. – 20 с.

Автор: В.М. Льовкін, канд. техн. наук, доцент

Рецензент: А.О. Олійник, д. т. н., професор

Відповідальний
за випуск: С.О. Субботін, д.т.н., професор

Затверджено
на засіданні кафедри
програмних засобів

Протокол № 8
від “30” березня 2023 р.

ЗМІСТ

ВСТУП

Дане видання призначене для вивчення студентами денної форми навчання алгоритмів та структур даних та практичного засвоєння вміння використовувати дані алгоритми в процесі розв'язання практичних завдань шляхом розроблення програмного забезпечення.

Відповідно до графіка студенти перед виконанням роботи повинні ознайомитися з конспектом лекцій та рекомендованою літературою. Дані методичні вказівки містять тільки основні, базові теоретичні відомості, необхідні для виконання роботи, тому для виконання роботи та при підготовці до її захисту необхідно ознайомитись з конспектом лекцій та опрацювати весь необхідний матеріал, наведений в переліку рекомендованої літератури, використовуючи також статті в інтернет-виданнях та актуальних наукових журналах з комп'ютерних наук.

Для одержання заліку студент повинен у відповідності зі всіма наведеними вимогами розробити програмне забезпечення та оформити звіт, після чого продемонструвати на комп'ютері розроблене програмне забезпечення з виконанням всіх запропонованих викладачем тестів.

Звіт виконують на білому папері формату А4 (210×297 мм). Текст розміщують тільки з однієї сторони листа. Поля сторінки з усіх боків – 20 мм. Аркуші вміщують у канцелярський файл.

Усі завдання повинні виконуватись студентами індивідуально і не містити плагіату як в оформленому звіті так і в розробленому програмному забезпеченні. Під час співбесіди при захисті роботи студент повинен виявити знання щодо мети роботи, теоретичного матеріалу, методів виконання кожного етапу роботи, змісту основних розділів звіту з демонстрацією результатів на конкретних прикладах, практичних прийомів використання теоретичного матеріалу в розробленому програмному забезпеченні. Студент повинен вміти обґрунтувати всі прийняті ним рішення та правильно аналізувати і використовувати на практиці отримані результати. Для базової самоперевірки при підготовці до виконання і захисту роботи студент повинен відповісти на контрольні запитання, наведені наприкінці опису відповідної роботи.

1 ОСНОВНІ ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1 Структури даних

1.1.1 Купи та черги з пріоритетами. Пірамідальне сортування

Пірамідальне сортування ґрунтується на використанні спеціалізованих структур даних, що дозволяють керувати інформацією в процесі виконання алгоритму.

Існує багато різних структур даних, які краще використовувати в тих чи інших випадках. Для одного з найпоширеніших випадків – сортування – зручно використовувати такі структури даних, як черги з пріоритетами та купи.

Черга з пріоритетами – структура даних, яка зберігає множину елементів, у якій кожен елемент v має певний пріоритет із значенням key(v).

Двійкова купа (binary heap) або піраміда – легка для реалізації структура даних, що дозволяє швидко вставляти та вилучати елемент з максимальним пріоритетом (наприклад, максимальний за значенням).

Двійкова купа представляє собою майже повне бінарне дерево, для якого виконується основна властивість купи: пріоритет кожної вершини більше пріоритетів її нащадків. В найпростішому випадку пріоритет кожної вершини можна вважати рівним її значенню. В такому випадку структура називається max-heap, оскільки корінь піддерева є максимумом зі значень елементів піддерева.

Алгоритм пірамідального сортування ґрунтується на використанні структури даних піраміда та складається з двох етапів:

– побудова піраміди;

– видалення найбільших елементів.

При підготовці матеріалів даного пункту використано джерела [1]-[4], [6]-[8], [12].

1.1.2 Геш-таблиці. Гешування

Гешування ґрунтується на ідеї розподілу ключів у одномірному масиві H [0..m-1], який називається геш-таблицею. Розподіл

виконується шляхом обчислення для кожного ключа значення деякої наперед визначеної функції h, яка називається геш-функцією.

Геш-функція призначає кожному з ключів геш-адресу, яка представляє собою ціле число від 0 до *m*-1. У загальному випадку геш-функція має задовольняти двом суперечливим вимогам:

– геш-функція має розподіляти ключі за комірками геш-таблиці як можна більш рівномірно;

– геш-функція повинна легко обчислюватися.

Методи побудови геш-функцій:

– метод ділення;

– метод множення;

– універсальне гешування.

Колізія – ситуація, коли два або більше ключів гешуються в одно й ту саму комірку геш-таблиці. Будь-яка схема гешування повинна мати механізм розв'язання колізій. Існує два основні підходи до оброблення колізій: з застосуванням ланцюжків і з застосуванням адресації.

При підготовці матеріалів даного пункту використано джерела [1], [3]-[8], [11], [12].

1.1.3 Бінарні дерева пошуку. B-дерева

Бінарне дерево – скінченна множина вузлів, яка може бути або порожньою, або складатися з кореня та двох бінарних дерев, які не перетинаються та називаються лівим та правим піддеревами кореня.

Існують три алгоритми обходу бінарних дерев:

– у прямому порядку;

– симетричний;

– центрований.

B-дерева є узагальненням бінарних дерев пошуку. Вони представляють собою збалансовані дерева пошуку, призначені для ефективної роботи з дисковою пам'яттю. Вузли B-дерев можуть мати багато дочірніх вузлів.

У B-деревах всі записи даних або ключі зберігаються в листі у зростаючому порядку ключів, а батьківські вузли використовують для індексування.

У процесі пошуку в B-дереві, починаючи з кореня слідують

ланцюжком покажчиків до листа, який може містити шуканий ключ. Після цього пошук відбувається серед ключів даного листа.

При підготовці матеріалів даного пункту використано джерела [1], [3]-[8], [11], [12].

1.2 Удосконалені методи розроблення та аналізу
1.2.1 Жадібні алгоритми

Жадібний алгоритм – метод розв'язання оптимізаційних задач, який ґрунтується на тому, що процес прийняття рішень можна розбити на елементарні кроки, на кожному з яких приймається окреме рішення. Рішення, що приймається на кожному кроці, повинно бути оптимальним тільки на поточному кроці і повинно прийматися без урахування попередніх або наступних рішень. Тобто на кожному кроці обирається найкращий варіант, вважаючи при цьому, що підсумкове рішення буде оптимальним.

Існують задачі, для яких послідовність таких жадібних виборів призведе до оптимального рішення для будь-якого екземпляра задачі, що розглядається. Однак для інших задач дане твердження не виконується: для розв'язання таких задач жадібний алгоритм може використовуватися у випадку, якщо прийнятним є приблизне рішення.

Коди Хаффмана широко розповсюджені для стиснення даних. Це доволі ефективний алгоритм, який дозволяє зазвичай зекономити від 20 до 90 % даних.

Алгоритм Хаффмана належить до жадібних алгоритмів і дозволяє закодувати текст, який побудований на основі n-символьного алфавіту. Кодування ґрунтується на принципі, за яким більш короткі коди призначаються символам, що зустрічаються частіше, а більш довгі – символам, що зустрічаються рідше. Тобто коди символів мають змінну довжину. Такий підхід не дозволяє визначити, скільки бітів кодованого тексту представляють кожний символ. Тому в кодах Хаффмана для розв'язання даної проблеми використовуються префіксні коди.

При підготовці матеріалів даного пункту використано джерела [1]-[4], [9], [11], [12].

1.2.2 Динамічне програмування

За своєю суттю динамічне програмування є методом проектування алгоритмів, які дозволяють розв'язувати задачі з підзадачами, які перекриваються. Такі підзадачі зазвичай виникають з рекурентних співвідношень, які зв'язують рішення даної задачі з рішеннями менших підзадач того ж виду.

Замість того, щоб розв'язувати підзадачі, що перекриваються, знову і знову, динамічне програмування дозволяє розв'язати кожну з менших підзадач тільки один раз, записуючи при цьому результат розв'язання в таблицю, з якої потім можна отримати розв'язання початкової задачі.

При підготовці матеріалів даного пункту використано джерела [1], [3], [4], [9], [11], [12].

1.3 Алгоритми для роботи з графами

1.3.1 Алгоритми обходу графів

Граф – це сукупність двох скінченних множин: множини точок та множини ліній, що попарно з'єднують деякі з цих точок.

Множина точок формує вершини (вузли) графа. Множина ліній, що з'єднують вершини графа, формує ребра (дуги) графа.

Приклад графічного зображення графа наведено на рис. 1.1.

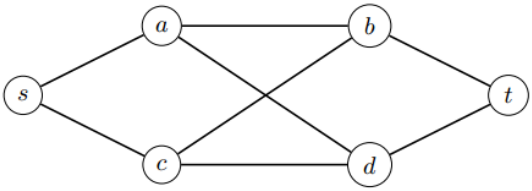

Рисунок 1.1 – Приклад графа

Найбільш розповсюдженими способами зображення графів є:
– списки суміжних вершин;
– матриця суміжності.
Для обходу графів існують два основні алгоритми:
– пошук в глибину;

– пошук в ширину.

Алгоритм пошуку в ширину в процесі обходу графа виконує обхід всіх вершин на відстані k перед тим, які перейти до пошуку вершин на відстані k+1.

Для відслідковування роботи алгоритму пошук в ширину розфарбовує вершини графа в кольори: білий, сірий та чорний. Спочатку вершини мають білий колір. Після того, як під час пошуку вершина досягається в перший раз, вона фарбується у сірий колір. Далі в процесі пошуку вершина фарбується в чорний колір.

Алгоритм пошуку в глибину орієнтований на те, щоб під час пошуку заглиблюватися в граф на стільки, на скільки це можливо. Пошук в глибину досліджує всі ребра, що виходять з вершини, яка була відкрита останньою, і залишає вершину тільки тоді, коли не залишається недосліджених ребер – в такому випадку пошук повертається у вершину, з якої була досягнута дана вершина.

Аналогічно пошуку в ширину пошук в глибину розфарбовує вершини графа в білий, сірий та чорний кольори.

Окрім безпосередньо обходу ребер графа та відвідування всіх вершин алгоритми пошуку в глибину та ширину корисні під час дослідження ряду важливих властивостей графів.

Сильно зв’язним компонентом орієнтованого графа називають максимальну множину вершин даного графа, для кожної пари яких справедливо, що вони досяжні одна з одної.

Для виконання пошуку сильно зв’язних компонент необхідно транспонувати заданий граф, тобто перетворити напрям ребер графа на протилежний.

Сам алгоритм пошуку сильно зв’язних компонент складається з двох пошуків у глибину:

– у початковому графі для обчислення часу завершення роботи з кожною вершиною;

– у транспонованому, де вершини розглядаються в порядку зменшення отриманих під час першого пошуку значень часу завершення роботи з кожною вершиною.

У такому випадку кожне дерево лісу пошуку в глибину, отримане під час другого пошуку, буде окремим сильно зв’язним компонентом.

При підготовці матеріалів даного пункту використано джерела [1], [3]-[6], [8], [9], [11], [12].

1.3.2 Алгоритми пошуку найкоротших шляхів

Алгоритм пошуку в ширину є алгоритмом пошуку найкоротшого шляху в незваженому графі, тобто в графі, кожному ребру якого відповідає одинична вага. Алгоритми Дейкстри, Беллмана-Форда та Флойда-Воршелла дозволяють працювати з графами, ребра яких мають не одиничну вагу.

Алгоритми пошуку найкоротших шляхів зазвичай ґрунтуються на твердженні, що найкоротший шлях між двома вершинами містить в собі інші найкоротші шляхи.

Алгоритм Дейкстри дозволяє виконувати пошук найкоротших шляхів з однієї вершини до всіх інших вершин у зваженому орієнтованому графі. Даний алгоритм знаходить найкоротші шляхи до вершин графа в порядку їх віддаленості від вихідної вершини, тобто спочатку знаходиться найкоротший шлях від вихідної вершини до найближчої, потім до другої найближчої тощо. Таким чином, перед початком i-ої ітерації алгоритм визначає найкоротші шляхи до $(i - 1)$-ої вершин, найближчих до вихідної. Дані вершини, вихідна вершина та ребра найкоротших шляхів утворюють піддерево даного графа. Чергова найближча до вихідної вершина може бути знайдена серед вершин, суміжних з отриманим піддеревом: для кожної суміжної вершини обчислюється сума відстаней до найближчої вершини дерева, після чого обирається вершина з найменшою сумою.

Алгоритм Беллмана-Форда дозволяє виконувати пошук найкоротших шляхів з однієї вершини до всіх інших вершин у випадку, коли вага кожного з ребер може бути від'ємною. Алгоритм повертає логічне значення, яке вказує, чи міститься в графі цикл з від'ємною вагою, що досягається з джерела. Якщо такий цикл існує, то алгоритм вказує на те, що рішення не існує. Якщо таких циклів не існує, то алгоритм повертає найкоротші шляхи та їх ваги.

Алгоритм Флойда-Воршелла призначений для пошуку найкоротшого шляху між всіма парами вершин. Даний алгоритм може бути реалізовано за допомогою обчислення матриці відстаней зваженого графа шляхом послідовних обчислень. Кожна матриця на відповідному

кроці містить довжини найкоротших шляхів між будь-якими двома вершинами, при чому кількість проміжних вершин у них не може перевищувати відповідний номер поточного кроку.

При підготовці матеріалів даного пункту використано джерела [1]-[3], [5], [6], [8], [9], [11], [12].

1.3.3 Моделювання транспортної мережі за допомогою графів

Під час моделювання транспортної мережі за допомогою графів кожне орієнтоване ребро можна розглядати як деякий канал, яким пересувається деякий продукт. Кожний канал має задану пропускну здатність, що характеризує максимальну швидкість пересування продуктів каналом. Вершини є точками перетину каналів. Через вершини, відмінні від джерела та стоку, продукт просувається, не накопичуючись.

Задача про максимальний потік полягає у знаходженні такого потоку за транспортною мережею, що сума потоків з витоку (до стоку) є максимальною.

Метод Форда-Фалкерсона дозволяє розв’язати задачу про максимальний потік шляхом ітеративного збільшення значення потоку. Спочатку потік обнуляється, а далі на кожній ітерації величина потоку збільшується шляхом пошуку збільшуючого шляху.

При підготовці матеріалів даного пункту використано джерела [1], [3], [8], [9].

2 ПОРЯДОК ВИКОНАННЯ САМОСТІЙНОЇ РОБОТИ

2.1 Узгодити з викладачем індивідуальне завдання для виконання всіх наступних пунктів.

2.2 Розробити програмне забезпечення з графічним інтерфейсом, призначене для аналізу та дослідження використання основних комп'ютерних алгоритмів, що складається з форми, на якій розташовано наступні вкладки:

– Пірамідальне сортування;
– Структури даних;
– Жадібні алгоритми;
– Динамічне програмування;
– Алгоритми обходу графів;
– Алгоритми пошуку найкоротших шляхів;
– Моделювання транспортної мережі.

Робота кожної вкладки має забезпечуватися за допомогою відповідних програмних модулів. Обов'язкові вимоги до програмних модулів та графічних інтерфейсів користувача для кожної вкладки описані нижче.

2.3 Для реалізації вкладки «Пірамідальне сортування» розв'язати індивідуальне завдання за допомогою пірамідального сортування, яке повинно забезпечуватися за допомогою:

– програмного модуля, що містить клас, який реалізує купу і має дозволяти виконувати наступні операції на основі окремих методів: вставлення елементу, сортування елементів, побудова купи з невпорядкованого масиву, видалення елементу, сортування елементів із використанням купи;

– програмного модуля, який реалізує графічний інтерфейс відповідної вкладки і дозволяє додавати нові елементи до купи на основі полів, що відповідають індивідуальному завданню, та на основі підключення файлів з масивами даних, вилучати існуючі елементи, виконувати пірамідальне сортування та два інші алгоритми сортування, визначені індивідуальним завданням, з виведенням результатів наочним способом (отриманого порядку елементів та часу, витраченого на сортування).

2.4 Для реалізації вкладки «Структури даних» розв’язати індивідуальне завдання за допомогою гешування та B-дерев, що повинно забезпечуватися за допомогою:

– програмного модуля, що містить окремі класи, які реалізують структури даних геш-таблиця, що дозволяє виконувати вставлення елементу, видалення елементу, пошук елементу на основі використання параметрів, обраних у відповідності з індивідуальним завданням, та B-дерево, що дозволяє виконувати створення порожнього дерева, пошук у дереві, вставлення ключа, видалення ключа;

– програмного модуля, що реалізує графічний інтерфейс відповідної вкладки і дозволяє виконувати формування геш-таблиці та B-дерева, додавання і видалення елементів, пошук, оброблення результатів, виведення їх у відповідні поля для виконання індивідуального завдання, при цьому розв’язуючи одне з завдань за допомогою обох структур даних.

2.5 Для реалізації вкладки «Жадібні алгоритми» розв’язати індивідуальне завдання за допомогою жадібного алгоритму, що повинно забезпечуватися за допомогою:

– програмного модуля, що містить клас, який дозволяє задавати початкові дані, вводити нові параметри, коригувати та видаляти існуючі, розв’язувати задачу;

– програмного модуля, який реалізує використання алгоритму Хаффмана для стискання даних текстового файлу у вигляді класу. Клас повинен мати методи, які дозволяють задати файл з даними, виконувати стискання даних, визначити параметри виконаного стискання та зворотне перетворення, записувати результати кодування/декодування в файл;

– програмного модуля, який реалізує графічний інтерфейс відповідної вкладки і дозволяє задавати параметри, що характеризують задачу, за допомогою відповідних полів, модифікувати їх та розв’язувати задачу за допомогою жадібного алгоритму, виконувати кодування та декодування файлу, задаючи всі необхідні параметри в графічному вигляді.

2.6 Для реалізації вкладки «Динамічне програмування» розв’язати індивідуальне завдання за допомогою принципів

динамічного програмування, що повинно забезпечуватися за допомогою:

– програмного модуля, що містить клас, який дозволяє задавати початкові дані, вводити нові параметри, коригувати та видаляти існуючі, розв'язувати відповідну задачу динамічного програмування;

– програмного модуля, який реалізує графічний інтерфейс відповідної вкладки і дозволяє задавати параметри, що характеризують задачу, за допомогою відповідних полів, модифікувати їх та розв'язувати задачу динамічного програмування.

2.7 Для реалізації вкладки «Алгоритми обходу графів» реалізувати алгоритми пошуку в ширину та в глибину за допомогою:

– програмного модуля, який реалізує алгоритм обходу графу на основі пошуку в глибину як для орієнтованого, так і для неорієнтованого графа. У процесі пошуку має бути сформовано ліс пошуку в глибину. Для реалізації має використовуватися стек. Модуль має бути побудовано на основі відповідного класу, який повинен дозволяти визначати граф, виконувати пошук в глибину, виводити побудований ліс пошуку в глибину, виводити результат обходу тощо;

– програмного модуля, який реалізує алгоритм обходу графу на основі пошуку в ширину як для орієнтованого, так і для неорієнтованого графа. У процесі пошуку має бути сформовано дерево пошуку в ширину. Для реалізації має використовуватися черга. Модуль має бути побудовано на основі відповідного класу, який повинен дозволяти визначати граф, виконувати пошук в ширину, виводити побудоване дерево пошуку в ширину, виводити результат обходу тощо;

– програмного модуля, який реалізує графічний інтерфейс відповідної вкладки і дозволяє задавати граф, виводити результати обходу за допомогою обраного користувачем алгоритму, а також демонструвати покрокове виконання обраного алгоритму за допомогою графіки.

2.8 Для реалізації вкладки «Алгоритми пошуку найкоротших шляхів» реалізувати алгоритми пошуку найкоротших шляхів за допомогою:

– програмних модулів, які реалізують алгоритми Дейкстри, Флойда-Уоршелла, Беллмана-Форда на основі виділення відповідних класів для виконання всіх необхідних обчислень, визначення

параметрів (в тому числі безпосередньо визначення графа) та отримання результатів;

– програмного модуля, який реалізує графічний інтерфейс відповідної вкладки і дозволяє задавати граф, виводити результати пошуку за допомогою обраного користувачем алгоритму пошуку найкоротшого шляху, а також демонструвати покрокове виконання обраного алгоритму за допомогою графіки.

2.9 Для реалізації вкладки «Моделювання транспортної мережі» реалізувати алгоритм Форда-Фалкерсона за допомогою:

– програмного модуля, який реалізує алгоритм Форда-Фалкерсона на основі виділення відповідного класу для виконання всіх необхідних обчислень, визначення параметрів (в тому числі безпосередньо визначення графа) та отримання результатів;

– програмного модуля, який реалізує графічний інтерфейс відповідної вкладки і дозволяє задавати граф, виводити результати виконання алгоритму Форда-Фалкерсона, а також демонструвати покрокове виконання обраного алгоритму за допомогою графіки.

2.10 Виконати тестування розробленого програмного забезпечення.

2.11 Довести, що жадібний вибір для задачі, визначеної індивідуальним завданням, є оптимальним рішенням або принаймні є частиною деякого оптимального рішення.

2.12 Виконати аналіз розроблених алгоритмів пірамідального сортування, динамічного програмування, жадібних алгоритмів, алгоритмів роботи з графами для розв’язання індивідуального завдання щодо часу їх роботи та кількості використаної пам’яті.

2.13 Порівняти ефективність використання пірамідального сортування та двох інших алгоритмів сортування, визначених індивідуальним завданням, щодо швидкості виконання.

2.14 Порівняти ефективність використання геш-таблиць та B-дерев для розв’язання одного й того самого завдання з п. 2.3.4 на основі часу, витраченого на виконання пошуку, вставлення елементу та видалення елементу.

2.15 Виконати аналіз впливу параметрів гешування на швидкість виконання гешування.

2.16 Оформити пояснювальну записку.

3 ВИМОГИ ДО ЗМІСТУ ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Пояснювальна записка до самостійної роботи оформлюється у відповідності з вимогами ДСТУ 3008:2015 [13].

При підготовці матеріалів даного розділу використано джерело [13].

3.1 Вступ

Вступ розташовують на окремій сторінці. У вступі коротко викладають: оцінку сучасного стану проблеми, відмічаючи практично розв'язані задачі, прогалини знань, що існують у даній галузі, провідні фірми та провідних вчених і фахівців даної галузі; світові тенденції розв'язання поставлених задач; актуальність даної роботи та підставу для її виконання; мету роботи та галузь застосування; взаємозв'язок з іншими роботами.

3.2 Основна частина

Основна частина пояснювальної записки складається з розділів, підрозділів, пунктів і підпунктів і містить наступні розділи:

– структури даних;

– удосконалені методи розроблення та аналізу;

– алгоритми роботи з графами.

Розділ зі структур даних повинен включати наступну інформацію:

– короткий опис алгоритму пірамідального сортування, індивідуальне завдання, особливості застосування алгоритму для розв'язання індивідуального завдання, опис програмного забезпечення в частині реалізації даного алгоритму з наведенням відповідних екранних форм, результати розв'язання завдання, характеристики роботи алгоритму, порівняння ефективності використання трьох алгоритмів сортування;

– короткий опис гешування та B-дерев як структур даних і відповідних їм алгоритмів, індивідуальне завдання, особливості застосування алгоритму для розв'язання індивідуального завдання, опис програмного забезпечення в частині реалізації геш-таблиць та B-

дерев з наведенням відповідних екранних форм, результати розв'язання завдання, характеристики роботи алгоритмів та порівняння структур даних щодо ефективності розв'язання завдання.

Розділ з удосконалених методів розроблення та аналізу повинен містити наступні дані:

– індивідуальне завдання, особливості використання жадібного алгоритму та зокрема алгоритму Хаффмана для розв'язання завдання, опис програмного забезпечення в частині реалізації даного алгоритму з наведенням відповідних екранних форм, результати розв'язання завдання, характеристики роботи алгоритму;

– індивідуальне завдання, особливості використання динамічного програмування для розв'язання завдання, опис програмного забезпечення в частині реалізації даного алгоритму з наведенням відповідних екранних форм, результати розв'язання завдання, характеристики роботи алгоритму.

Розділ з алгоритмів роботи з графами повинен включати наступну інформацію:

– короткий опис алгоритмів обходу графів, особливості власного застосування алгоритмів, опис програмного забезпечення в частині реалізації даних алгоритмів з наведенням відповідних екранних форм, характеристики роботи алгоритмів;

– короткий опис алгоритмів пошуку найкоротших шляхів, особливості власного застосування алгоритмів, опис програмного забезпечення в частині реалізації даних алгоритмів з наведенням відповідних екранних форм, характеристики роботи алгоритмів;

– короткий опис алгоритму Форда-Фалкерсона, особливості власного застосування алгоритмів, опис програмного забезпечення в частині реалізації даного алгоритму з наведенням відповідних екранних форм, характеристики роботи алгоритму.

3.3 Висновки

Висновки вміщують безпосередньо після викладення суті записки, починаючи з нової сторінки. У висновках наводять найбільш важливі результати, одержані в самостійній роботі, виконують аналіз і оцінку одержаних результатів роботи з урахуванням світових тенденцій вирішення поставленої задачі; можливі галузі використання

результатів роботи; народногосподарську, наукову, соціальну значущість роботи.

У висновках необхідно наголосити на якісних та кількісних показниках отриманих власноруч результатів, обґрунтувати достовірність результатів, викласти рекомендації щодо їх використання.

Висновки повинні займати не менше однієї повної сторінки.

3.4 Додатки

Додатки можуть містити матеріал, який є необхідним для повноти пояснювальної записки, але включення його до основної частини записки може змінити впорядковане й логічне уявлення про роботу; не може бути послідовно розміщений в основній частині записки через великий обсяг або способи відтворення; може бути вилучений для широкого кола читачів, але є необхідним для фахівців даної галузі.

У додатки можуть бути включені: додаткові ілюстрації або таблиці; матеріали, які через великий обсяг, специфіку викладення або форму подання не можуть бути внесені до основної частини (оригінали фотографій, проміжні математичні докази, формули, розрахунки; протоколи випробувань; копія технічного завдання, інструкції, методики, опис комп'ютерних програм, розроблених у процесі виконання роботи та ін.).

У одному з додатків обов'язково має бути наведено повний текст розробленого програмного забезпечення.

Якщо додатки оформлюють на наступних сторінках записки, кожний такий додаток повинен починатися з нової сторінки. Додаток повинен мати заголовок, надрукований вгорі малими літерами з першої великої симетрично відносно тексту сторінки. Посередині рядка над заголовком малими літерами з першої великої повинно бути надруковано слово "Додаток ___" і велика літера, що позначає додаток.

ЛІТЕРАТУРА

1. Кормен, Т. Вступ до алгоритмів [Текст] / Томас Кормен, Чарльз Лейзерсон, Рональд Рівест. – К. : К. І. С., 2019. – 1288 с.

2. Кормен, Т. Г. Алгоритми доступно [Текст] / Т. Г. Кормен. – Київ : К. І. С., 2021. – 194 с.

3. Introduction to Algorithms, fourth edition [Текст] / Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. – The MIT Press, 2022. – 1312 p.

4. Ришковець, Ю. В. Алгоритмізація та програмування. Частина 2 [Текст] : навчальний посібник / Ю. В. Ришковець, В. А. Висоцька. – Львів: Видавництво «Новий Світ-2000», 2020. – 320 с.

5. Мелешко, Є. В. Алгоритми та структури даних [Текст] : Навчальний посібник для студентів технічних спеціальностей денної та заочної форми навчання / Є. В. Мелешко, М. С. Якименко, Л. І. Поліщук. – Кропивницький: Видавець – Лисенко В.Ф., 2019. – 156 с.

6. Креневич, А.П. Алгоритми і структури даних. Підручник [Текст] / А.П. Креневич. – К. : ВПЦ "Київський Університет", 2021. – 200 с.

7. Knuth, D. The Art of Computer Programming, Vol. 1 [Текст] : Fundamental Algorithms / D. Knuth. – Addison-Wesley Professional, 1997. – 672 p.

8. Sedgewick, R. Algorithms (4th Edition) [Текст] / Robert Sedgewick, Kevin Wayne. – Addison-Wesley Professional, 2011. – 976 p.

9. Erickson, J. Algorithms [Текст] / J. Erickson. – 2019. – 472 p. – ISBN : 978-1-792-64483-2.

10. Louridas, P. Algorithms [Текст] : The MIT Press Essential Knowledge series / P. Louridas. – Cambridge, Massachusetts : The MIT Press, 2020. – 312 p.

11. Bhargava, A. Y. Grokking Algorithms [Текст] : An illustrated guide for programmers and other curious people / A. Y. Bhargava. – New York : Manning Publications, 2016. – 256 p.

12. Mehlhorn, K. Algorithms and Data Structures [Текст] : The Basic Toolbox / Kurt Mehlhorn, Peter Sanders. – Berlin : Springer, 2008. – 300 p.

13. ДСТУ 3008:2015 Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання [Текст]. – К. : ДП «УкрНДНЦ», 2016. – 31 с.