АЛГОРИТМИ ТА МОДЕЛІ ОБЧИСЛЕНЬ: ТЕЗИ ДО ПРОГРАМИ КУРСУ

Н.Б. Козак

Національний університет «Львівська політехніка»,
кафедра електронних обчислювальних машин
E-mail: nazar.b.kozak@lpnu.ua

Подано опис програми курсу «Алгоритми та моделі обчислень» та висвітлено його зв'язок з іншими дисциплінами спеціальності «Комп'ютерна інженерія».

Ключові слова: теорія алгоритмів, теорія обчислюваності, теорія складності обчислень, моделі обчислень, методи розробки алгоритмів.

Вступ

Відповідність спеціальності «Комп'ютерна інженерія». Курс «Алгоритми та моделі обчислень» є точнішою назвою дисципліни «Алгоритми та методи обчислень», у якій словосполучення «методи обчислень» часто помилково тлумачиться як «чисельні методи» *(наприклад, чисельне диференціювання, інтегрування, обчислення за допомогою рядів)*, що є частиною прикладної математики.

У прикладній математиці модель обчислень (прикладна математична модель) – це система рівнянь або функцій, які описують певне явище або процес *(наприклад, зміну параметра в часі t відносно початкового моменту t_0)*. У комп'ютерній інженерії модель обчислень – це формальна алгоритмічна система, яка реалізує алгоритм, який, у свою чергу, відповідає обчисленням у прикладній математичній моделі. Тобто алгоритм, зокрема, може бути засобом реалізації прикладної математичної моделі, а модель обчислень у контексті комп'ютерної інженерії – засіб реалізації самого алгоритму. Якщо в курсі «Архітектура комп'ютера» розглядається апаратна реалізація обчислень, то курс «Алгоритми та моделі обчислень» присвячений формалізації алгоритмів [1], яка передує їхній апаратній реалізації.

Курс є розвитком дисципліни професора Черкаського М.В. «Алгоритми та методи обчислень» [2], яка викладалась на кафедрі ЕОМ протягом багатьох років.

Оновлений курс зосереджується на формуванні у студентів навичок програмування різними мовами, умінні проектувати алгоритми з використанням різних алгоритмічних стратегій, а також на глибокому розумінні різних моделей обчислень.

Загальна структура курсу «Алгоритми та моделі обчислень». Курс розроблено на основі класичних літературних джерел [3–11]. Його структура охоплює такі основні компоненти:

- Теорія алгоритмів *(англ. Theory of Computation)* [5]:
  - теорія автоматів *(англ. Automata theory)*;
  - теорія формальних мов *(англ. Formal language theory)*;
  - теорія обчислюваності *(англ. Computability theory)*;
  - теорія складності обчислень *(англ. Computational complexity theory)*;
- Моделі обчислень *(англ. Models of Computation)* [6, 7];
- Методи проектування алгоритмів (алгоритмічні стратегії) *(англ. Algorithm Design Paradigm, Algorithmic Paradigm, Algorithmic Technique, Algorithmic Strategy)* [8].

Базова частина курсу сформована з урахуванням програм провідних університетів, зокрема курсів MIT: «Automata, Computability, and Complexity» [12] та «Theory of Computation» [13]. Особливу увагу в курсі приділено функційній і реактивній моделям обчислень, які підтримуються відповідними мовами програмування та бібліотеками. Це дає змогу доповнити знання студентів, які раніше знайомились з процедурною та об'єктно-орієнтованою парадигмами.

У розділі про функційні моделі обчислень розглядається лямбда-числення і комбінаторна логіка, які разом з знаннями про комбінаційну логіку *(яку студенти вивчають у курсі «Комп'ютерна логіка»)* формують у студентів цілісне уявлення про даний тип обчислень. Також вводяться базові елементи теорії категорій та теорії моделей, які особливо важливі для інтерпретації формальних мов як моделей обчислень – аспект, який часто ігнорується в подібних курсах.

Відповідність єдиному фаховому вступному випробуванню з інформаційних технологій. Курс «Алгоритми та моделі обчислень» (АМО) забезпечує підготовку студентів до єдиного фахового вступного випробування (ЄФВВ) з інформаційних технологій у частині «Алгоритми та обчислювальна складність». Відповідність окремих розділів структури системи знань за курсом *(додаток А)* вимогам ЄФВВ подано у таблиці 1.

Таблиця 1

ЄФВВ		АМО
№	Питання	№
1.1.1	Поняття алгоритму. Визначення його часової та просторової (за обсягом пам'яті) складності	1.1.2 2.1.1 2.1.2
1.1.2	Поняття абстрактного типу даних. Абстрактні типи даних: стеки, списки, вектори, словники, множини, мультимножини, черги, черги з пріоритетами	3.2.6.1 3.2.6.3 3.2.6.5 3.2.6.2 3.2.6.7 3.2.6.6 3.2.6.4
1.1.3	Кортежі, множини, словники, одно- та двобічнозв'язні списки. Реалізація абстрактних типів даних з оцінюванням складності операцій	3.2.8.1 3.2.8.3 3.2.8.4 3.2.8.5 3.2.8 (3.2.8.9)
1.1.4	Базові алгоритми та їх складність: пошук, сортування (прості сортування вибором, вставками, обмінами та удосконалені сортування деревом, сортування Шелла, швидке сортування)	4.1 4.2.1 4.2.2 4.2.3 4.2.8 4.2.5 4.2.6
1.1.5	Алгоритми на графах та їх складність: пошук в ширину і глибину; пошук зв'язних компонентів; побудова кістякового дерева; побудова найкоротших шляхів з виділеної вершини; побудова найкоротших шляхів між двома вершинами	4.5 (4.5.17) 4.5.1 4.5.4 4.5.2 4.5.8 4.5.9
1.2.1	Стратегія «розділяй та володарюй» та приклади застосування	3.3.2.3
1.2.2	Стратегія балансування та приклади застосування	3.3.2.9
1.2.3	Динамічне програмування та приклади застосування	3.3.2.5
1.2.4	Оцінювання складності алгоритму під час застосування кожної стратегії	3.3.2.10
1.3.1	Імперативний та декларативний підходи до програмування	6.4
1.3.2	Розв'язні, напівроз'язні та нерозв'язні проблеми. Проблема зупинки	6.2.2.2 6.2.2.3

Детальна структура курсу «Алгоритми та моделі обчислень»

Розширену детальну структуру системи знань за курсом «Алгоритми та моделі обчислень» наведено в додатку А. Відповідно до цієї структури сформовано такі розділи програми курсу:

Тема 1. Вступ до теорії алгоритмів. Розділ починається з неформального тлумачення поняття алгоритму та переходить до його формалізації за допомогою класичних формальних алгоритмічних систем, яким ставляться у відповідність формальні мови та їхні граматики згідно з ієрархією Хомського. Особлива увага приділяється скінченним детермінованим та недетермінованим автоматам, які розпізнають регулярні мови, оскільки на кожному наступному рівні ієрархії пристрій, що розпізнає відповідну формальну мову, є розширенням скінченного автомату. Окремо розглядаються взаємні перетворення регулярних виразів та скінченних автоматів.

Тема 2. Основи аналізу алгоритмів. Це класична частина курсу, яка охоплює аналіз обчислювальної складності за часом виконання та обсягом пам'яті. Окремо виділено тему структурної складності алгоритмів *(з урахуванням напрацювань проф. Черкаського М.В.)* та ієрархії класів складності, що рідко висвітлюються в подібних курсах.

Тема 3. Методи відображення та синтез алгоритмів. У багатьох класичних джерелах [3–10] розгляд відображення алгоритмів опускається, оскільки модель алгоритму часто розглядається як його відображення. Однак у цьому курсі прийнято рішення виділити це питання окремо. Окрім класичних блок-схем, розглядаються й інші способи відображення алгоритмів. Також, через відсутність у навчальному плані дисципліни «Програмування, частина 3. Структури даних та алгоритми», у цей розділ включено тему абстрактних типів даних *(АТД)* та відповідних структур даних. Крім того, вивчаються алгоритмічні стратегії *(методи розробки алгоритмів)*.

Тема 4. Базові алгоритми обробки інформації. Розділ присвячено базовим алгоритмам обробки інформації: алгоритмам пошуку, сортування, порівняння зі взірцем, алгоритмам на графах та іншим. Такі алгоритми часто використовуються як компоненти в більш складних прикладних алгоритмах і є обов'язковими у подібних курсах.

Тема 5. Бібліотеки базових алгоритмів обробки інформації для популярних мов програмування. Розглянуто засоби узагальненого програмування мовами C++, C# та Java. Такий підхід є базовим у сучасній розробці. Хоча студенти частково ознайомлені зі стандартною бібліотекою шаблонів (STL) у C++, дуже важливо проаналізувати аналогічні інструменти в C# і Java з можливістю покрокового порівняння. У розділ також включено огляд деяких прикладних бібліотек.

Тема 6. Моделі обчислень. Спершу розділ доповнює розгляд класичних формальних алгоритмічних систем та відповідних мов теорією моделей, зокрема інтерпретацією формальних мов. Далі викладається теорія обчислюваності – ще один класичний компонент подібних курсів [12]. Новим елементом є включення теорії категорій, яка забезпечує семантичну інтерпретацію лямбда-числення *(зокрема через декартово замкнені категорії)* при вивченні функційної парадигми. Також розглядається бібліотека ReactiveX як приклад засобу реактивного програмування.

Взаємозв'язок курсу з іншими дисциплінами

Курсовий проєкт «Розробка системних програмних модулів та компонент систем програмування». Теорія автоматів і формальних мов, яка найповніше викладається у курсі

«Алгоритми та моделі обчислень», є теоретичною основою для розробки компіляторів та інтерпретаторів у цьому курсовому проєкті.

Курси «Комп'ютерна логіка» та «Комп'ютерна схемотехніка». Ці курси розглядають автомати Мура та Мілі з практичного погляду. Курс «Алгоритми та моделі обчислень» продовжує вивчення автоматів, зосереджуючись на формальній стороні. Якщо курс «Комп'ютерна логіка» розглядати як адаптацію першої частини курсу «Математична логіка та теорія алгоритмів», то «Алгоритми та моделі обчислень» є його продовженням у частині теорії алгоритмів.

Врахування відсутності курсу «Програмування, частина 3. Алгоритми та структури даних». У зв'язку з тим, що цей курс не викладається для студентів спеціальності «Комп'ютерна інженерія», до програми курсу «Алгоритми та моделі обчислень» включено теми про абстрактні типи даних *(АТД)* та відповідні структури даних.

Література

1. Параметричні моделі алгоритмів [Електронний ресурс] / М. В. Черкаський, А. О. Саченко // Электротехнические и компьютерные системы. - 2011. - № 4. - С. 162-167. - Режим доступу: http://nbuv.gov.ua/UJRN/etks_2011_4_30

2. Мельник А. О. Теорія алгоритмів і методи обчислень: новий курс / А. О. Мельник, М. В. Черкаський // Вісник Національного університету «Львівська політехніка». – 2001. – № 437 : Комп'ютерні системи та мережі. – С. 99–105.

3. Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford. Introduction to Algorithms. — 3rd. — MIT Press, 2009. — ISBN 0-262-03384-4.

4. Donald E. Knuth.The Art of Computer Programming, Volumes 1-4A Boxed Set. Third Edition (Reading, Massachusetts: Addison-Wesley, 2011), 3168pp. ISBN 978-0-321-75104-1, 0-321-75104-3.

5. Michael Sipser (2013). Introduction to the Theory of Computation. 3rd. Cengage Learning. ISBN 978-1-133-18779-0

6. Savage, John E. (1998). Models Of Computation: Exploring the Power of Computing. ISBN 978-0-201-89539-1

7. Fernandez, Maribel (2009). Models of Computation: An Introduction to Computability Theory. Undergraduate Topics in Computer Science. Springer. ISBN 978-1-84882-433-1.

8. Anany Levitin (2012). Introduction to the design & analysis of algorithms. 3rd. ISBN-13: 978-0-13-231681-1

9. Aho, A. V., Hopcroft, J. E., & Ullman, J. D. (1983). Data Structures and Algorithms. Addison-Wesley. ISBN 978-0-201-00023-8.

10. Aho, A. V., Hopcroft, J. E., & Ullman, J. D. (1974). The Design and Analysis of Computer Algorithms. Addison-Wesley. ISBN 978-0-201-00029-0.

11. Hopcroft, J. E. & Ullman, J. D., Introduction to Automata Theory, Languages and Computation, Reading, Mass.:. Addison-Wesley, 1979. ISBN 978-0321455369.

12. https://ocw.mit.edu/courses/6-045j-automata-computability-and-complexity-spring-2011/pages/lecture-notes/

13. https://ocw.mit.edu/courses/18-404j-theory-of-computation-fall-2020/pages/lecture-notes/

Додаток А

Частина 1. Тема №1. Вступ до теорії алгоритмів.

1.1 Неформальне тлумачення алгоритму

1.1.1 Історія тлумачення алгоритму

1.1.2 **Поняття алгоритму**

1.1.3 Основні властивості алгоритму

1.1.4 Параметри алгоритму

1.1.5 Базові структури алгоритмів *(алгоритмічні конструкції)*. Теорема Бьома-Якопіні.

1.1.6 Рекурсивні алгоритми

1.1.7 Паралельні алгоритми

1.1.8 Недетерміновані алгоритми

1.1.9 Імовірнісні алгоритми *(увипадковлені алгоритми)*

1.2 Формалізація поняття алгоритму

(продовження в частині курсу, яка присвячена моделям обчислень(тема №6))

1.2.1 Введення в теорію алгоритмів

1.2.2 Абстрактні моделі алгоритму

1.2.3 Формальні алгоритмічні системи *(ФАС)*

1.2.4 Скінченний автомат

1.2.4.1 Детермінований скінченний автомат *(DFA)*

1.2.4.2 Недетермінований скінченний автомат *(NFA)* та імовірнісний автомат*(PA).*

1.2.4.3 ω-автомат

1.2.4.4 Автомат з однобуквеними переходами. Формування автомату з однобуквеними переходами за заданим недетермінованим автоматом.

1.2.4.5 Видалення непродуктивних та недосяжних станів скінченного автомату

1.3.4.6 Видалення λ-переходів та ε-переходів у недетермінованому скінченному автоматі

1.3.4.7 Детермінізація квазідетермінованого скінченного автомату

1.3.4.8 Мінімізація скінченого детермінованого автомату

1.2.5 Перетворювачі *(трансдуктори)* на основі детермінованого скінченного автомату

1.2.5.1 Автомат Мура *(Moore machine)*

1.2.5.2 Автомат Мілі *(Mealy machine)*

1.2.6 Автомат з магазинною пам'яттю *(МП-автомат, PDA)*

1.2.7 Машина Тюрінга та її варіанти

1.2.8 Числення Поста

1.2.9 Нормальні алгоритми Маркова

1.2.10 Регістрова машина

1.2.11 РАМ-машина

1.2.12 ПРАМ-машина та варіанти пам’яті із впорядкованим доступом

1.3 Формальні граматики та формальні мови

1.3.1 Формальні граматики, мови та ієрархія Хомського

1.3.2 Регулярні мови та вирази

1.3.2.1 Властивості граматик регулярних мов. Автоматні граматики. Доповнення автоматної мови.

1.3.2.2 Лема про накачку для регулярних мов

1.3.2.3 Знаходження мови для заданої регулярної граматики

1.3.2.4 Регулярні вирази

1.3.2.5 Формування регулярного виразу для заданого недетермінованого скінченного автомату

1.3.2.6 Формування недетермінованого скінченного автомату для заданої регулярної граматики

1.3.2.7 Формування недетермінованого скінченного автомату для заданого регулярного виразу

1.3.3 КВ-мови *(контекстно-вільні мови)* та нотації БНФ

1.3.3.1 Властивості граматик КВ-мов

1.3.3.2 Лема про накачку для КВ-мов

1.3.3.3 Дерева розбору КВ-граматик

1.3.3.4 Створення МП-автомату для заданої КВ-граматики

1.3.3.5 Нотації БНФ та РБНФ

1.3.3.6 Нормальна форма Хомського для КВ-граматик

1.3.3.7 Алгоритм Кока-Касамі-Янгера для КВ-граматик у нормальній формі Хомського

1.4 Формалізація поняття алгоритму (доповнення до 1.2)

1.4.1 Детермінізація недетермінованого скінченного автомату *(доповнення до 1.2.4)*

Частина 1. Тема 2. Основи аналізу алгоритмів

2.1 Обчислювальна складність

2.1.1 **Складність по часу виконання алгоритму**

2.1.1.1 Оцінка розміру вхідних даних

2.1.1.2 Залежність часу виконання від розміру вхідних даних

2.1.1.3 Аналіз нерекурсивних алгоритмів
2.1.1.4 Аналіз рекурсивних алгоритмів
2.1.1.5 Аналіз алгоритму відповідно до варіантів його виконання: для найкращого випадку, для середнього випадку та для найгіршого випадку
2.1.1.6 Асимптотичний аналіз
2.1.1.6.1 Загальні визначення
2.1.1.6.2 Нотація Бахмана-Ландау *(група асимптотичних нотацій)*: O-, o-, Ω-, ω-, Θ- нотації
2.1.1.7 Детермінований (DTIME) та недетермінований (NTIME) ресурси часу виконання
2.1.1.8 Здійсненні класи складності
2.1.1.8.1 DLONGTIME-клас складності *(логарифмічний)*
2.1.1.8.2 PolylogTIME-клас складності *(полілогарифмічний)*
2.1.1.8.3 P-клас складності (поліноміальний)
2.1.1.8.4 P-повні задачі
2.1.1.8.5 RP-клас та coRP-клас складності
2.1.1.8.6 ZPP-клас складності
2.1.1.8.7 BPP-клас складності
2.1.1.8.8 BQP-клас складності
2.1.1.9 Проблематичні класи складності
2.1.1.9.1 NP- та coNP- класи складності
2.1.1.9.2 Співвідношення між P- та NP- класами складності
2.1.1.9.3 NP- та coNP- повні задачі. Теорема Кука-Левіна. Стандартні NP-повні задачі.
2.1.1.9.4 NP- складні, еквівалентні, проміжні та прості задачі
2.1.1.9.5 PP-клас складності
2.1.1.9.6 UP-клас складності
2.1.1.9.7 #P-клас *(вимовляється як «шарп П»)* складності та #P-повні задачі
2.1.1.9.8 ⊕P-клас *(вимовляється як «паритет П»)* складності
2.1.1.10 Нездійсненні класи складності
2.1.1.10.1 EXPTIME-клас складності *(експоненціальний клас складності)*
2.1.1.10.2 NEXPTIME-клас складності
2.1.1.10.3 2-EXPTIME-клас складності
2.1.1.10.4 ELEMENTARY-клас складності
2.1.1.10.5 R-клас складності
2.1.1.10.6 PR-клас складності
2.1.1.10.7 RE-клас складності та coRE-клас складності
2.1.1.10.8 ALL-клас складності
2.1.1.11 Аналіз паралельних алгоритмів
2.1.1.11.1 Особливості аналізу паралельних алгоритмів
2.1.1.11.2 Теорема Брента. Закон Густавсона–Барсіса. Закон Амдала.
2.1.1.11.3 NC-клас складності

2.1.2 **Ємнісна (просторова) складність алгоритму *(складність по об'єму пам'яті)***
2.1.2.1 Визначення просторової складності алгоритму
2.1.2.2 Детермінований (DSPACE) та недетермінований (NSPACE) просторові (ємнісні) ресурси
2.1.2.3 Теорема Севіча
2.1.2.4 Здійсненні класи складності
2.1.2.4.1 L-клас складності
2.1.2.4.2 PolyL-клас складності *(полілогарифмічний)*
2.1.2.4.3 SL-клас складності
2.1.2.4.4 NL-клас складності
2.1.2.4.5 NL-повні задачі
2.1.2.4.6 Еквівалентність класів NL та coNL
2.1.2.4.7 RL-клас складності
2.1.2.5 Проблематичні класи складності
2.1.2.5.1 PSPACE-клас складності
2.1.2.5.2 PSPACE-повні задачі
2.1.2.6 Нездійсненні класи складності
2.1.2.6.1 EXPSPACE-клас складності *(експоненціальний клас складності по пам'яті)*

2.2. Структурна складність
2.2.1 Цикломатична складність
2.2.1.1 Цикломатичне число
2.2.1.2 Цикломатична складність графу потоку керування та графу алгоритму
2.2.2 Структурна складність обчислень поданих структурною матрицею потокового графу алгоритму
2.2.3 AC^i-класи складності
2.2.4 ACC^0-клас складності

2.2.5 TC^i-класи складності
2.2.6 CC-клас складності
2.3 Ієрархії класів складності
2.3.1 Теорема про ієрархії класів часової складності
2.3.2 Теорема про ієрархії класів просторової складності
2.3.3 Поліноміальна ієрархія та PH-клас складності
2.3.4 Експоненціальна ієрархія
2.3.5 Ієрархія Гжегорчика
2.3.6 Арифметична ієрархія
2.3.7 Булева ієрархія

Частина 1. Тема 3. Методи відображення та синтез алгоритмів
3.1 Методи відображення алгоритмів
3.1.1 Вербальне та аналітичне подання алгоритму
3.1.2 Подання алгоритму псевдокодом або з використанням формальних мов
3.1.3 Схематичне подання алгоритму
3.1.3.1 Просте графічне подання алгоритму
3.1.3.1.1 Блок-схема алгоритму
3.1.3.1.2 Граф потоку керування
3.1.3.1.3 Граф алгоритму
3.1.3.1.4 Потоковий граф алгоритму
3.1.3.2 Структурограма
3.1.3.2.1 Діаграма Нассі-Шнайдермана
3.1.3.3 Діаграми UML
3.1.3.3.1 Множина структурних та поведінкових діаграм UML
3.1.3.3.2 Подання задачі поведінковими діаграмами
3.1.3.3.2.1 Діаграма прецедентів *(діаграма варіантів використання)*
3.1.3.3.3 Подання обчислень поведінковими діаграмами
3.1.3.3.3.1 Діаграма послідовності
3.1.3.3.3.2 Діаграма комунікації
3.1.3.3.3.3 Узагальнена діаграма взаємодій
3.1.3.3.3.4 Діаграма стану
3.1.3.3.3.5 Діаграма діяльності
3.1.3.3.4 Структурні діаграми
3.1.3.3.4.1 Діаграма класів
3.1.3.3.4.2 Діаграма компонентів
3.1.3.3.4.3 Діаграма розгортання
3.1.4 Подання алгоритму структурною матрицею потокового графу алгоритму
3.2 Типи та структури даних
3.2.1 Представлення даних в пам’яті комп’ютера
3.2.2 Класифікація типів даних
3.2.3 Базові типи даних
3.2.4 Похідні типи даних
3.2.5 Перетворення типів
3.2.6 **Абстрактні типи даних (АТД)**
3.2.6.1 **Поняття абстрактного типу даних**. Контейнери та колекції
3.2.6.2 **Вектор**
3.2.6.3 **Стек**
3.2.6.4 **Черга**. **Черга з пріоритетом**. Двобічна черга та двобічна черга з пріоритетом.
3.2.6.5 **Список**
3.2.6.6 **Множина**. **Мультимножина**.
3.2.6.7 Асоціативний масив (**словник**). Мультисловник
3.2.6.8 Граф
3.2.6.9 Дерево
3.2.7 Класифікація структур даних
3.2.8 **Реалізації абстрактних типів даних**
3.2.8.1 **Кортеж**
3.2.8.2 Геш-таблиця
3.2.8.3 Реалізція **множин** за допомогою геш-таблиці
3.2.8.4 Реалізція **словників** за допомогою геш-таблиці
3.2.8.5 **Одно- та двобічно зв'язні списки**. Список з пропусками. Розгорнутий зв'язаний список.
3.2.8.6 Бінарне дерево пошуку
3.2.8.6.1 Збалансоване дерево

3.2.8.6.2 АВЛ-дерево
3.2.8.6.3 Червоно-чорне дерево
3.2.8.7 Б-дерево *(англ. B-tree)* та R-дерево
3.2.8.8 Купа
3.2.8.8.1. Двійкова купа
3.2.8.8.2. Біноміальна купа
3.2.8.8.3. Фібоначчієва купа
3.2.8.9 **Оцінювання складності операцій при реалізації АТД**

3.3 Синтез алгоритмів

3.3.1 Покрокове проектування алгоритмів
3.3.2 Підходи при синтезі алгоритмів *(алгоритмічні стратегії)*
3.3.2.1 Повний перебір та приклади застосування
3.3.2.2 Метод зменшення розміру задачі та приклади застосування
3.3.2.3 Метод декомпозиції *(«**розділяй та володарюй**»)* **та приклади застосування**
3.3.2.4 Метод перетворень та приклади застосування
3.3.2.5 **Динамічне програмування та приклади застосування**
3.3.2.6 Дерево розв'язків та його застосування при проектуванні алгоритмів
3.3.2.6.1 Дерево розв'язків *(дерево рішень)*
3.3.2.6.2 Бектрекінг *(перебір з поверненням)*
3.3.2.6.3 Метод гілок і границь
3.3.2.6.4 Альфа-бета відсікання
3.3.2.7 Евристичні алгоритми
3.3.2.7.1 Особливості евристичних алгоритмів
3.3.2.7.2 Метод спроб і помилок *(trial and error)*
3.3.2.7.3. Скупі *(жадібні)* алгоритми та локальний пошук
3.3.2.7.3.1 Особливості скупих алгоритмів
3.3.2.7.3.2 Локальний пошук
3.3.2.8 Ітераційне вдосконалення алгоритму
3.3.2.9 Просторово-часовий компроміс *(**стратегія балансування**)* при проектуванні алгоритмів **та приклади застосування**
3.3.2.10 **Оцінювання складності алгоритму під час застосування кожної стратегії**

Частина 1. Тема 4. Базові алгоритми обробки інформації

4.1 Алгоритми пошуку

4.1.1 Послідовний пошук та оцінка його складності
4.1.2 Послідовний пошук з бар'єром та оцінка його складності
4.1.3 Бінарний пошук та оцінка його складності
4.1.4 Порозрядний пошук та оцінка його складності
4.1.5 Зовнішній пошук
4.1.6 Застосування геш-таблиць для пошуку. Розв'язання колізій при гешуванні відкритою адресацією та методом ланцюжків.

4.2 Алгоритми сортування даних

4.2.1 **Сортування вибором** та оцінка його складності
4.2.2 **Сортування вставками** та оцінка його складності
4.2.3 **Сортування обміном** та оцінка його складності
4.2.4 Сортування злиттям та оцінка його складності
4.2.5 **Сортування Шелла** та оцінка його складності
4.2.6 **Швидке сортування** та оцінка його складності
4.2.7 Пірамідальне сортування та оцінка його складності
4.2.8 **Сортування деревом** та оцінка його складності
4.2.9 Порозрядне сортування та оцінка його складності
4.2.10 Мережі сортування
4.2.11 Зовнішнє сортування

4.3 Алгоритми порівняння зі взірцем

4.3.1 Примітивний алгоритм пошуку підрядка та оцінка його складності
4.3.2 Алгоритм Рабіна-Карпа та оцінка його складності
4.3.3 Алгоритм Кнута-Морріса-Пратта та оцінка його складності
4.3.4 Алгоритм Бойєра-Мура та оцінка його складності
4.3.5 Пошук підрядків за допомогою скінчених автоматів
4.3.6 Наближене порівняння рядків

4.4 Чисельні алгоритми

(більш глибоко тему розкрито в додатковій частині курсу, яка присвячена чисельним методам(тема №10))

4.4.1 Матриці та дії з ними. Алгоритм Копперсміта-Віноградa та алгоритм Штрассена.
4.4.2 Робота з довгими числами

4.4.3 Многочлени та швидке перетворення Фур'є
4.4.4 Системи алгебраїчних рівнянь
4.4.5 Розв'язання систем лінійних рівнянь
4.4.6 Розв'язання нелінійних рівнянь
4.4.7 Алгоритми апроксимації і інтерполяція чисельних функцій

4.5 Алгоритми на графах та мережеві алгоритми

4.5.1 Пошук у графі. **Пошук в ширину і глибину**.
4.5.2 **Побудова кістякового дерева** *(каркасу графа)*
4.5.3 Каркас мінімальної ваги. Метод Дж. Крускала. Метод Р. Пріма.
4.5.4 Досяжність. Визначення зв'язності. **Пошук зв'язних компонентів**. Двозв'язність.
4.5.5 Ейлерові цикли
4.5.6 Гамільтонові цикли
4.5.7 Фундаментальна множина циклів
4.5.8 **Побудова найкоротших шляхів з виділеної вершини**. Алгоритм Дейкстри. Алгоритм Беллмана-Форда.
4.5.9 **Побудова найкоротших шляхів між двома вершинами**
4.5.10 Алгоритм Флойда-Воршелла
4.5.11 Метод генерації всіх максимальних незалежних множин графа. Задача про найменше покриття.
4.5.12 Задача про найменше розбиття.
4.5.13 Розфарбування графа. Пошук мінімального розфарбування вершин графа.
4.5.14 Потоки в мережах
4.5.15 Метод побудови максимального потоку в мережі
4.5.16 Методи наближеного рішення задачі комівояжера: алгоритм Ейлера, алгоритм Крістофідеса, застосування локальної оптимізації
4.5.17 **Складність алгоритмів на графах**

4.6 Паралельні та розподілені алгоритми

4.6.1 Методи паралельного виконання програми за допомогою спільної пам'яті або за допомогою передачі повідомлень
4.6.2 Організація паралельних обчислень відповідно до принципу консенсусу і на основі вибору
4.6.3 Методи визначення завершення паралельних обчислень
4.6.4 Паралельний пошук, паралельне сортування, паралельні чисельні алгоритми, паралельні алгоритми на графах

Частина 1. Тема 5. Бібліотеки базових алгоритмів обробки інформації для популярних мов програмування

5.1 Застосування базових алгоритмів при узагальненому програмуванні на C++ засобами STL (Standard Template Library)

5.1.1 Огляд бібліотеки
5.1.2 Огляд базових типів бібліотеки
5.1.3 Засоби бібліотеки для роботи з стрічками та вводом/виводом
5.1.4 Контейнерні класи бібліотеки
    5.1.4.1 Послідовні контейнери бібліотеки
    5.1.4.2 Асоціативні контейнери бібліотеки
5.1.5 Ітератори
5.1.6 Функціональні об'єкти
    5.1.6.1 Арифметичні функціональні об'єкти
    5.1.6.2 Предикати
    5.1.6.3 Адаптери
        5.1.6.3.1 Заперечувачі
        5.1.6.3.2 Зв'язувачі
        5.1.6.3.3 Адаптери вказівників на функції
        5.1.6.3.4 Адаптери методів
5.1.7 Алгоритми бібліотеки
    5.1.7.1 Алгоритми сортування та пошуку
    5.1.7.2 Чисельні алгоритми
    5.1.7.3 Інші немодифікуючі алгоритми
    5.1.7.4 Інші модифікуючі алгоритми
5.1.8 Розподільники пам'яті для контейнерних класів бібліотеки

5.2 Застосування базових алгоритмів при узагальненому програмуванні на C++ засобами Boost

5.2.1 Огляд набору бібліотек Boost
5.2.2 Контейнери та алгоритми
5.2.3 Стрічкові алгоритми
5.2.4 Окремі засоби набору бібліотек Boost
    5.2.4.1 Застосування boost::any
    5.2.4.2 Застосування boost::assign

5.2.4.3 Застосування boost::function
5.2.4.4 Застосування boost::bind
5.2.4.5 Застосування boost::optional
5.2.4.6 Застосування boost::variant
5.2.4.7 Застосування boost::lexical_cast
5.2.4.8 Застосування boost::spirit
5.2.4.9 Застосування boost::filesystem
5.2.4.10 Застосування boost::asio
5.2.4.11 Застосування boost::static_assert
5.2.5. Метапрограмування за допомогою boost::mpl

5.3 Застосування базових алгоритмів при узагальненому програмуванні на Java засобами JCL (Java Class Library)

5.3.1 Загальний огляд мови Java. Засоби для узагальненого програмування на Java.
5.3.2 Огляд бібліотеки
5.3.3 Поняття ітератора в контексті застосування бібліотеки
5.3.3.1 Застарілий інтерфейс Enumeration
5.3.3.2 Інтерфейс Iterator
5.3.3.3 Інтерфейс Iterable
5.3.4 Інтерфейс Collection
5.3.5 Інтерфейс Set та його реалізації
5.3.5.1 Інтерфейси Set, SortedSet та NavigableSet
5.3.5.2 Класи HashSet, LinkedHashSet та TreeSet
5.3.6 Інтерфейс Queue та його реалізації
5.3.6.1 Інтерфейси Queue та Deque
5.3.6.2 Класи LinkedList, ArrayDeque та PrioryQueue
5.3.7 Інтерфейс List та його реалізації
5.3.7.1 Інтерфейс List
5.3.7.2 Класи Vector, Stack, ArrayList та LinkedList
5.3.7.3 Інтерфейс ListIterator.
5.3.8 Інтерфейс Map та його реалізації
5.3.8.1 Інтерфейси Map, SortedMap та NavigableMap
5.3.8.2 Класи HashTable, TreeMap, HashMap, LinkedHashMap, ArrayList та WeakHashMap
5.3.9 Алгоритми з допоміжного класу Collections
5.3.10 Використання бібліотеки для конкурентних обчислень
5.3.10.1 Застосування ключового слова synchronized
5.3.10.2 Огляд засобів пакету java.until.concurrent

5.4 Застосування базових алгоритмів при узагальненому програмуванні на C# засобами FCL (Framework Class Library)

5.4.1 Загальний огляд мови C#. Засоби для узагальненого програмування на C#.
5.4.2 Огляд узагальнених та неузагальнених засобів бібліотеки
5.4.3 Поняття ітератора в контексті застосування бібліотеки
5.4.3.1 Узагальнений та неузагальнений інтерфейси IEnumerator
5.4.3.2 Узагальнений та неузагальнений інтерфейси IEnumerable
5.4.4 Узагальнений та неузагальнений інтерфейси ICollection
5.4.5 Узагальнений та неузагальнений інтерфейси IList та їх реалізації
5.4.5.1 Узагальнений та неузагальнений інтерфейси IList
5.4.5.2 Узагальнені класи HashSet, List та Collection
5.4.5.3 Неузагальнені класи ArrayList та Array
5.4.6 Узагальнений та неузагальнений класи Stack
5.4.7 Узагальнений та неузагальнений класи Queue
5.4.8 Неузагальнений клас BitArray
5.4.9 Узагальнений та неузагальнений інтерфейси IDictonary та їх реалізації
5.4.9.1 Узагальнений та неузагальнений інтерфейси IDictonary
5.4.9.2 Узагальнені класи Dictonary, SortedDictonary та SortedList
5.4.9.3 Неузагальнені класи ListDictonary, HashTable та SortedList
5.4.10 Використання бібліотеки для конкурентних обчислень
5.4.10.1 Властивості ICollection.IsSynchronized та ICollection.SyncRoot
5.4.10.2. Огляд засобів простору імен System.Collections.Concurrent

5.5 Застосування алгоритмів лінійної алгебри при програмуванні на C++ за допомогою стандарту BLAS

5.5.1 Огляд стандарту
5.5.2 Три рівні функціональності стандарту
5.5.3 Огляд бібліотек-реалізації стандарту
5.5.4 Бібліотека-реалізація uBLAS з набору бібліотек Boost

5.5.5 Застосування uBLAS для мови C++

5.6 Застосування алгоритмів обробки сигналів при програмуванні на C++ та Python засобами OpenCV (Open Source Computer Vision Library)

5.6.1 Огляд бібліотеки. Доступність бібліотеки різним мовам програмування.
5.6.2 Основні модулі бібліотеки
5.6.3 Типове застосування бібліотеки
5.6.4 Застосування OpenCV для мови C++
5.6.5 Застосування OpenCV для мови Python

Частина 2. Тема 6. Моделі обчислень

Тут не розглядаються формальні мови та відповідні їм моделі автоматів. Формальні мови та відповідні їм моделі автоматів розглядаються в темі №1. Тут також не розглядаються великі мовні моделі (LLM). LLM (англ. large language model) розглядаються в темі №10 (додаткова частина курсу).

6.1 Елементи теорії моделей

6.1.1 Поняття теорії моделей
6.1.2 Моделі та структури в теорії моделей
6.1.3 Інтерпретації формальних мов
6.1.4 Теорія повноти *(Completeness Theorem) (для логіки першого порядку)*
6.1.5 Принцип перенесення *(Transfer Principle)* моделей *(для логіки першого порядку)*
6.1.6 Визначуваність *(Definability)*

6.2 Теорія обчислюваності

6.2.1 Основи обчислюваності *(Computability)*
6.2.1.1 Визначення теорії обчислюваності
6.2.1.2 Поняття обчислюваної функції *(Computable Function)*
6.2.1.3 Примітивно-рекурсивні функції *(Primitive Recursive Functions)*
6.2.1.4 Частково-рекурсивні функції *(Partial Recursive Functions)*
6.2.1.5 Теза Черча-Тюрінга *(Church–Turing Thesis)*
6.2.2 Розв'язність та розпізнаваність
6.2.2.1 Задача про прийняття рішень *(Decision problem)*
6.2.2.2 **Розв'язні, напіврозв'язні та нерозв'язні проблеми**
6.2.2.2.1 Розв'язні *(Decidable)* проблеми
6.2.2.2.2 Напіврозв'язні *(Semidecidability/Recursively enumerable/Recognizability)* проблеми *(розпізнавані проблеми)*
6.2.2.2.3 Нерозв'язні *(Undecidable)* проблеми
6.2.2.3 **Проблема зупинки** машини Тюрінга *(Halting Problem)*
6.2.2.4 Нерозв'язувана *(Undecidable)* задача про прийняття рішень та степінь Тюрінга *(Turing Degrees)*
6.2.3 Необчислюваність *(Uncomputability)*
6.2.3.1 Поняття необчислюваної функції *(Uncomputable function)*
6.2.3.2 Теореми Геделя про неповноту *(Gödel's Incompleteness Theorems)*
6.2.3.3 Обчислення з оракулом *(Computation with Oracle)*
6.2.4 Нумерації та універсальність *(Gödel Numbering, Enumeration, Universality)*
6.2.4.1 Нумерації алгоритмів
6.2.4.2 Нумерація Геделя
6.2.4.3 Головні універсальні функції та множини
6.2.4.4 Канторові нумерації кортежів натуральних чисел
6.2.4.5 Нумерації Кліні та Поста
6.2.4.6 Нумерація машин Тюрінга та частково-рекурсивних функцій
6.2.4.7 S_n^m-теорема *(Smn-теорема)* та теорема Кліні про нерухому точку
6.2.5 Зведення задач *(Problem Reductions)*
6.2.5.1 Зведення однієї задачі до іншої за поліноміальний час
6.2.5.2 Зведення Карпа *(Karp reductions, many-one reductions)*
6.2.5.3 Зведення Кука *(Cook reduction)*
6.2.5.4 Зведення Левіна *(Levin reduction)*
6.2.5.5 Зведення через таблицю істинності *(truth-table reduction)*
6.2.5.6 Зведення Тюрінга *(Turing reduction)*

6.3 Елементи теорії категорій

6.3.1 Поняття теорії категорій
6.3.2 Приклади категорій
6.3.3 Дуальна категорія *(Dual category)*
6.3.4 Морфізми в категоріях *(Morphisms)*
6.3.5 Початкові та термінальні об'єкти *(Initial and terminal objects)*
6.3.6 Функтори в теорії категорій

6.3.7 Натуральні перетворення *(Natural transformations)*
6.3.8 Категорії множин *(Set category)*
6.3.9 Категорії типів *(Type categories)*
6.3.10 Картезіансько-замкнені *(декартово замкнені)* категорії *(Cartesian closed categories)*
6.3.11 Моноїдальні *(тензорні)* категорії *(Monoidal categories)*

6.4 Імперативний та декларативний підходи до програмування

6.4.1 Класифікація парадигм програмування
6.4.2 Імперативне програмування *(Imperative programming)*: структурне програмування *(Structured programming)*, процедурне програмування *(Procedural programming)* та об'єктно-орієнтоване програмування *(Object-Oriented Programming, OOP)*
6.4.3 Декларативне програмування (Declarative programming): функційне програмування *(Functional programming)*, логічне програмування *(Logic programming)* та символьне і правилове програмування *(Rule-based and symbolic programming)*
6.4.4 Семантична відмінність: опис «як зробити» проти «що зробити»
6.4.5 Сучасні тренди у парадигмах програмування: реактивне програмування *(Reactive programming)*, потокове програмування *(Stream processing)*, асинхронне та подійно-орієнтоване програмування

6.5 Функційні моделі обчислень та парадигма функційного програмування. Комбінаторна логіка.

6.5.1 Функційні моделі обчислень
6.5.1.1 Лямбда числення
6.5.1.1.1 Вступ до лямбда числення
6.5.1.1.2 Підстановки та перетворення при застосуванні лямбда числення
6.5.1.1.3 Розширення чистого лямбда числення
6.5.1.1.4 Теорема про нерухому точку
6.5.1.1.5 Редекси і нормальна форма
6.5.1.1.6 Теорема Черча-Россера
6.5.1.1.7 Редукція термів в лямбда численні
6.5.1.1.8 Типізоване лямбда числення
6.5.1.1.8.1 Поняття типу в типізованому лямбда численні
6.5.1.1.8.2 Просте типізоване лямбда числення
6.5.1.1.8.3 Підстановка типу та уніфікація
6.5.1.2 Комбінаторна логіка *(як варіант лямбда числення)*
6.5.1.3 Комбінаційна логіка *(як функційна модель обчислень)*
6.5.1.4 Абстрактна система переписувань *(Abstract rewriting system)*
6.5.2 Парадигма функційного програмування
6.5.3 Функційне програмування за допомогою сучасних мов програмування
6.5.3.1 Мова функційного програмування Haskell
6.5.3.1.1 Програмування на Haskell
6.5.3.1.2 Базові типи в Haskell
6.5.3.1.3 Системи модулів в Haskell
6.5.3.1.4 Списки в Haskell
6.5.3.1.5 Види поліморфізму. Параметричний та спеціальний поліморфізм в Haskell.
6.5.3.1.6 Класи типів в Haskell
6.5.3.1.7 Стандартні класи типів в Haskell
6.5.3.1.8 Реалізації класів типів в Haskell
6.5.3.1.9 Згортки в Haskell
6.5.3.1.10 Моноїди в Haskell
6.5.3.1.11 Клас типів Foldable
6.5.3.1.12 Функтори в Haskell
6.5.3.1.13 Клас типів Pointed
6.5.3.1.14 Аплікативні функтори
6.5.3.1.15 Клас типів Traversable
6.5.3.1.16 Клас типів монад
6.5.3.1.17 Монада Maybe
6.5.3.1.18 Монада IO
6.5.3.1.19 Монада Reader та монада Writer
6.5.3.1.20 Монада State
6.5.3.2. Загальний огляд засобів функційного програмування на C++
6.5.3.3. Загальний огляд мов функційного програмування Erlang та Elixir
6.5.3.4. Загальний огляд інших засобів функційного програмування

6.6 Паралельні моделі обчислень та парадигма реактивного програмування. Паралельне програмування.

6.6.1 Паралельні моделі обчислень
6.6.1.1 Мережа процесів Кана
6.6.1.2 Мережа Петрі

6.6.1.3 Мережа взаємодій
6.6.1.4 Синхронний потік даних
6.6.2 Парадигма реактивного програмування
6.6.3 Функційне реактивне програмування
6.6.4 Реактивне програмування за допомогою бібліотеки ReactiveX
6.6.4.1 Шаблон спостерігача *(Observer pattern)* та його розширення у ReactiveX
6.6.4.2 Сутності для спостереження *(Observable)*
6.6.4.3 Оператори бібліотеки ReactiveX
6.6.4.4 Сутність Single
6.6.4.5 Сутність суб'єкту *(Subject)*
6.6.4.6 Планувальник *(Scheduler)* рушія бібліотеки ReactiveX
6.6.4.7 Реалізації ReactiveX для популярних мов програмування
6.6.5 Загальний огляд реактивного програмування за допомогою фреймворку Spring WebFlux з набору фреймворків Spring
6.6.6 Паралельне програмування
6.7 Шаблони проектування програмного забезпечення
6.7.1 Використання шаблонів при проектуванні програмного забезпечення
6.7.2 GoF-шаблони
6.7.2.1 Твірні шаблони *(Creational pattern)*
6.7.2.2 Структурні шаблони *(Structural pattern)*
6.7.2.3 Поведінкові шаблони *(Behavioral pattern)*
6.7.3 GRASP-шаблони
6.7.4 Шаблони рівночасних обчислень *(Concurrency pattern)*
6.7.5 Шаблони архітектури програмного забезпечення *(Architectural pattern)*
6.7.6 Використані шаблони в ядрі Linux