«Алгоритми та моделі обчислень»

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

1.1. Неформальне тлумачення алгоритму.

1.1.1. Історія поняття алгоритму.
1.1.2. Визначення 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*(Probabilistic algorithms)*.

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 Чьомскі.
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)

1.4.1. Детермінізація недетермінованого скінченного автомату.

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 для найкращого, середнього та найгіршого випадку.
- 2.1.1.6. Асимптотичний аналіз.
  - 2.1.1.6.1. Загальні визначення.
  - 2.1.1.6.2. 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 Абстрактоного типу даних. Контейнери та колекції.
  - 3.2.6.2. Стек.
  - 3.2.6.3. Черга. Черга з пріоритетом. Двобічна черга та двобічна черга з пріоритетом.
  - 3.2.6.4. Список.
  - 3.2.6.5. Граф.
  - 3.2.6.6. Дерево.
  - 3.2.6.7. Множина. Мультимножина.
  - 3.2.6.8. Асоціативний масив(Словник). Мультисловник.
- 3.2.7. Класифікація структур даних.
- 3.2.8. Деякі важливі структури даних для реалізації АТД.
  - 3.2.8.1. Геш-таблиця.
  - 3.2.8.2. Одно- та двобічнозв'язні списки. Список з пропусками. Розгорнутий зв'язаний список.
  - 3.2.8.3. Бінарне дерево пошуку.
    - 3.2.8.3.1. Збалансоване дерево.
    - 3.2.8.3.2. АВЛ-дерево.
    - 3.2.8.3.3. Червоно-чорне дерево.
  - 3.2.8.4. Б-дерево*(англ. B-tree)* та R-дерево.
  - 3.2.8.5. Купа.
    - 3.2.8.5.1. Двійкова купа.
    - 3.2.8.5.2. Біноміальна купа.
    - 3.2.8.5.3. Фібоначчієва купа.
  - 3.2.8.6. 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.
  - 3.3.2.2. Метод зменшення розміру задачі.
  - 3.3.2.3. Метод декомпозиції*(«розділяй та володарюй»)*.
  - 3.3.2.4. Метод перетворень.
  - 3.3.2.5. Динамічне програмування.
  - 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*(просторово-часове балансування)* при проектуванні алгоритмів.
  - 3.3.2.10. Прогнозування складності алгоритму під час застосування відповідних стратегії.

Частина 1. Тема №4. Базові алгоритми обробки інформації.

4.1. Алгоритми пошуку.

4.1.1. Послідовний пошук.
4.1.2. Послідовний пошук з бар'єром.
4.1.3. Бінарний пошук.
4.1.4. Порозрядний пошук.
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.
4.2.2. Сортування вставками.
4.2.3. Сортування обміном.
4.2.4. Сортування злиттям.
4.2.5. Сортування Шелла.
4.2.6. Швидке сортування.
4.2.7. Пірамідальне сортування.
4.2.8. Порозрядне сортування.
4.2.9. Мережі сортування.
4.2.10. Зовнішнє сортування.

4.3. Алгоритми порівняння зі взірцем.

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

4.4. Чисельні алгоритми.

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. Графи та мережеві алгоритми.

4.5.1. Пошук у графі.
4.5.2. Породження всіх каркасів графа.
4.5.3. Каркас мінімальної ваги. Метод Дж. Крускала. Метод Р. Пріма.
4.5.4. Досяжність. Визначення зв'язності. Двозв'язність.
4.5.5. Ейлерові цикли.
4.5.6. Гамільтонові цикли.
4.5.7. Фундаментальна множина циклів.
4.5.8. Алгоритм Дейкстри.
4.5.9. Алгоритм Флойда.
4.5.10. Метод генерації всіх максимальних незалежних множин графа. Задача про найменше покриття.
4.5.11. Задача про найменше розбиття.
4.5.12. Розфарбування графа.
4.5.13. 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(метод локальної оптимізації, алгоритм Эйлера, алгоритм Крістофідеса).
4.5.17. Аналіз 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 основних 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. Приклад коду мовою 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. Приклад коду мовою C++.
- 5.6.5. Приклад коду мовою Python.

Частина 2. Тема 6. Моделі обчислень.

6.1. Елементи теорії моделей.

6.1.1. Визначення теорії моделей.
6.1.2. Моделі в теорії моделей.
6.1.3. Інтерпретації формальних мов.
6.1.4. Теорія повноти моделей*(для логіки першого порядку)*.
6.1.5. Принцип перенесення моделей*(для логіки першого порядку)*.

6.2. Елементи теорії обчислюваності. Алгоритмічно розв'язні та нерозв'язні проблеми.

6.2.1. Визначення теорії обчислюваності.
6.2.2. Поняття обчислюваної функції.
6.2.3. Примітивно-рекурсивні функції.
6.2.4. Теза Черча-Тюрінга.
6.2.5. Задача про прийняття рішень.
6.2.6. Необчислюваність.
6.2.6.1. Теореми Геделя про неповноту.
6.2.6.2. Поняття необчислюваної функції*(Uncomputable function)*.
6.2.6.3. Задача про зупинку машини Тюрінга.
6.2.6.4. Нерозв'язувана задача про прийняття рішень та степінь Тюрінга.
6.2.7. Обчислення з оракулом.
6.2.8. Зведення однієї задачі до іншої за поліноміальний час.
6.2.8.1. Зведення Карпа*(Karp reductions, Many-one reductions)*.
6.2.8.2. Зведення Кука*(Cook reduction)*.
6.2.8.3. Зведення Левіна*(Levin reduction)*.
6.2.8.4. Зведення через таблицю істинності*(Truth-table reduction)*.
6.2.8.5. Зведення Тюрінга*(Turing reduction)*.

6.3. Елементи теорії категорій.

6.3.1. Визначення теорії категорій.
6.3.2. Приклади категорій.
6.3.3. Дуальна категорія.
6.3.4. Морфізми в теорії категорій.
6.3.5. Початковий та термінальний об'єкти в теорії категорій.
6.3.6. Функтори в теорії категорій.
6.3.7. Натуральне перетворення в теорії категорій.
6.3.8. Моноїдальна категорія*(тензорна категорія)*.

6.4. Імперативний та декларативний підходи до 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.