

Чт 1

Алгоритми та моделі обчислень
KI-201, KI-202, KI-203, KI-204, 512 V н.к.,
лекція

Алгоритми та моделі обчислень
KI-205, KI-206, KI-207, KI-208, 512 V н.к.,
лекція

Пт 2

Алгоритми та моделі обчислень
KI-205, KI-206, KI-207, KI-208, 906 V н.к.,
лекція

3

Алгоритми та моделі обчислень
KI-201, KI-202, KI-203, KI-204, 512 V н.к.,
лекція

Алгоритми та моделі обчислень

Додаткове вступне лекційне заняття

Лабораторні та практичні роботи з курсу АМО

▼ Матеріали лабораторних занять.

Лабораторна робота №1



ЗАВДАННЯ

[Завантаження звітів до лабораторної роботи №1](#)



ГІПЕРПОСИЛАННЯ

[Методичні вказівки до виконання лабораторної роботи №1](#)



ГІПЕРПОСИЛАННЯ

[lab1 source code](#)



ГІПЕРПОСИЛАННЯ

[source code for sort implementation](#)

Лабораторна робота №2



ЗАВДАННЯ

[Завантаження звітів до лабораторної роботи №2](#)

Лабораторна робота №6



ЗАВДАННЯ

Завантаження звітів до лабораторної роботи №6



ГІПЕРПОСИЛАННЯ

Методичні вказівки до виконання лабораторної роботи №6



ГІПЕРПОСИЛАННЯ

lab6 source code


unpack lab6 source code



ГІПЕРПОСИЛАННЯ

unpack lab6 source code

Курс: Алгоритми та моделі обчислень

Завдання: Завантаження звітів до лабораторної роботи №2 

[Переглянути всі роботи](#)

Оцінка

Оцінка (макс. 5)

Поточна оцінка в журналі

▼ Матеріали практичних занять.

Практичне заняття №1

(завдання до практичного заняття №1 відсутнє, оскільки це вступне заняття)



ГІПЕРПОСИЛАННЯ

Інформація до практичного заняття №1

Практичне заняття №2



ЗАВДАННЯ

Завантаження звітів до практичного заняття №2



ГІПЕРПОСИЛАННЯ

Інформація до практичного заняття №2



ГІПЕРПОСИЛАННЯ

Завдання до практичного заняття №2

Практичне заняття №3

Курс: Алгоритми та моделі обчислень

Завдання: Завантаження звітів до практичного заняття №2 ⚙

Переглянути всі роботи

Оцінка

Оцінка

Оцінка:

Без оцінки ⚡

Без оцінки

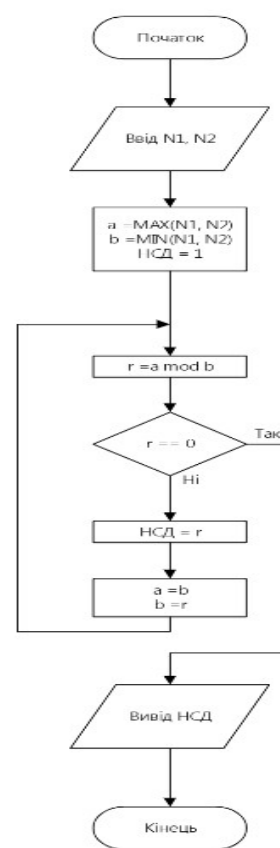
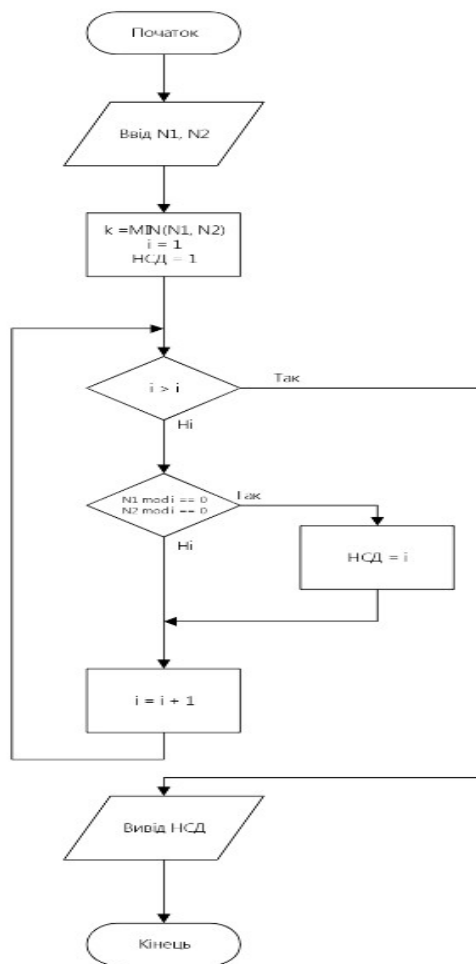
Yes

No

в журналі

Назва роботи: алгоритм; властивості, параметри та характеристики складності алгоритму.

Мета роботи: проаналізувати складність заданих алгоритмів.



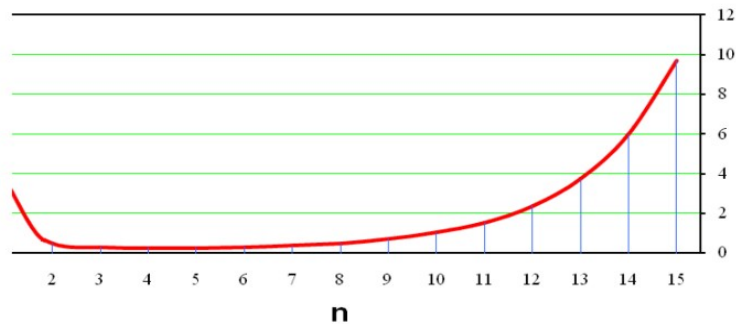
Назва роботи: асимптотичні характеристики складності алгоритму;
алгоритми з поліноміальною та експоненціальною складністю.

Мета роботи: ознайомитись з асимптотичними характеристиками складності та класами складності алгоритмів.

$$Y(n) = \frac{2^n}{n^3} :$$

K=3

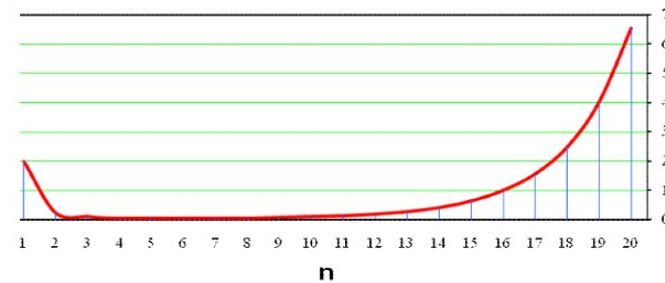
Y(n)



$$Y(n) = \frac{2^n}{n^4} :$$

K=4

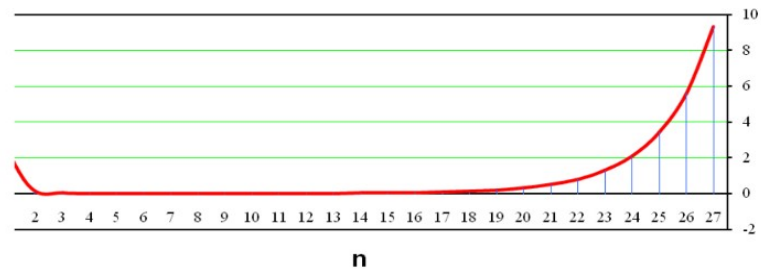
Y(n)



$$Y(n) = \frac{2^n}{n^5} :$$

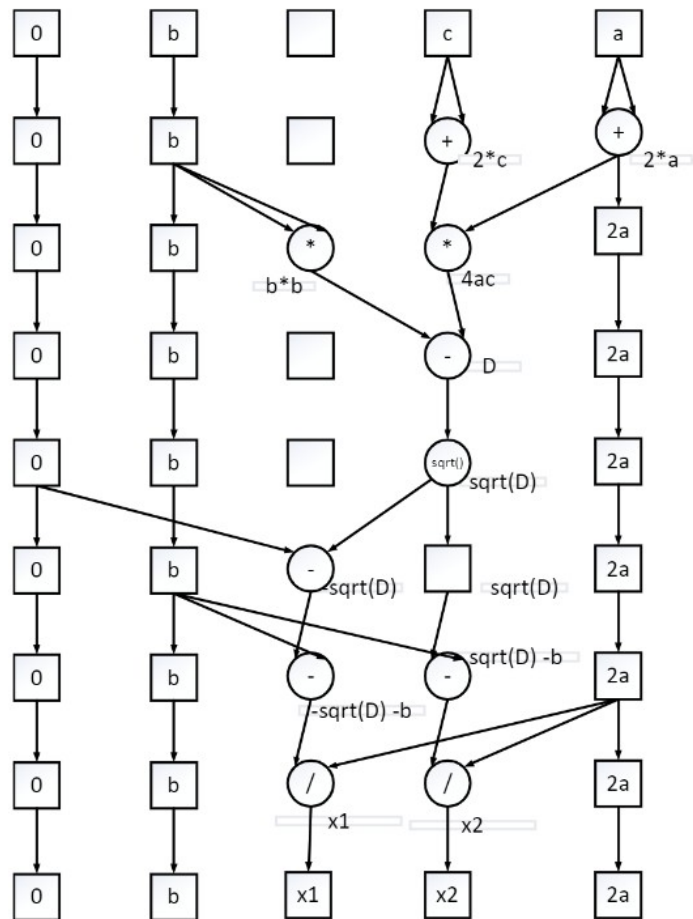
K=5

Y(n)



Назва роботи: використання потокового графу алгоритму при проектуванні паралельних обчислень.

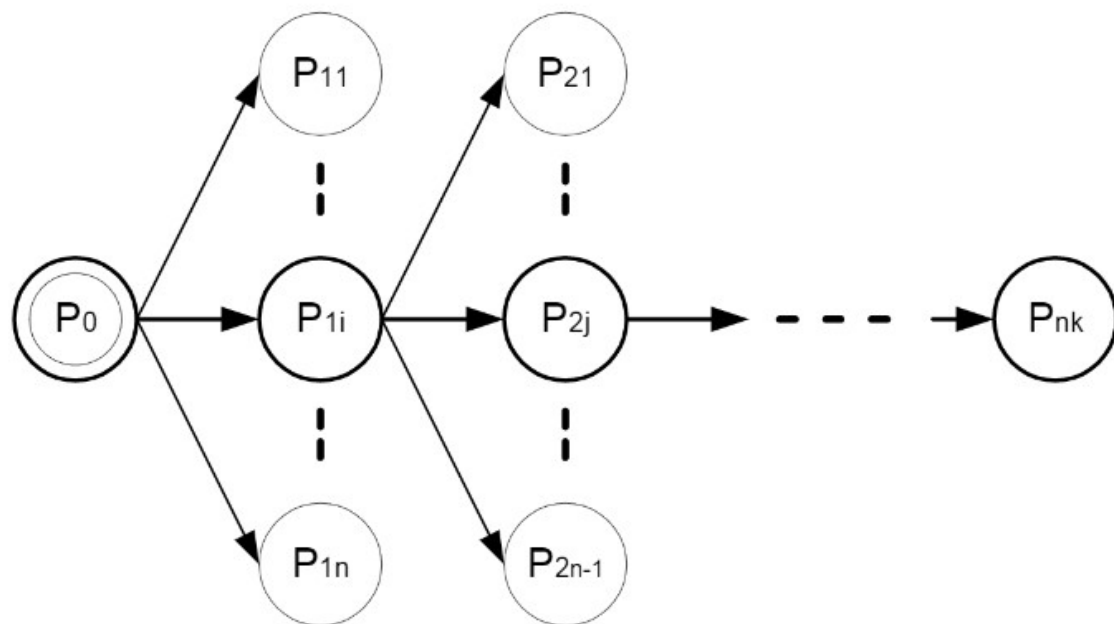
Мета роботи: ознайомитися з використанням потокового графу алгоритму при паралельному програмуванні.



ЛАБОРАТОРНА РОБОТА №4

Назва роботи: побудова алгоритмів ефективних за часовою складністю;
задача квадратичного призначення.

Мета роботи: виконати зменшення часової складності методом «гілок і
границь».



Задане дискретне робоче поле (ДРП) і матриця зв'язності R. Розташувати елементи X1, X2, X3, X4, X5 на ДРП за критерієм мінімальної сумарної довжини з'єднань.

ДРП

R

x	1	2	3	4	5
1	0	1	2	0	3
2	1	0	3	0	2
3	2	3	0	1	2
4	0	0	1	0	1
5	3	2	2	1	0

Позначимо на ДРП номери позицій в правому верхньому куті вільних позицій та визначимо матрицю відстаней D.

ДРП

1			
	2		
		4	5
	3		

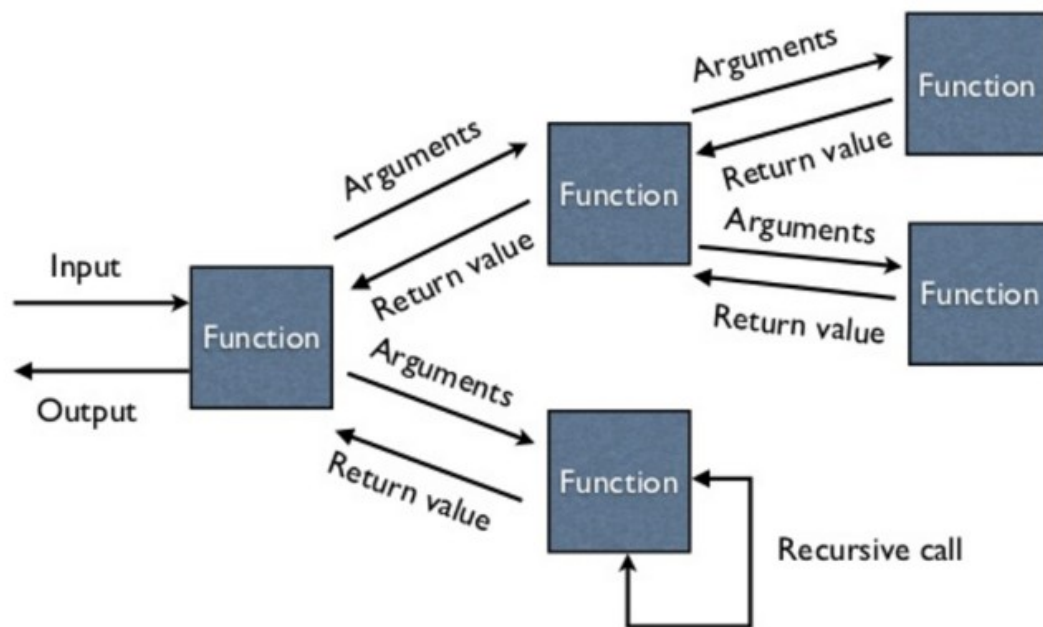
D

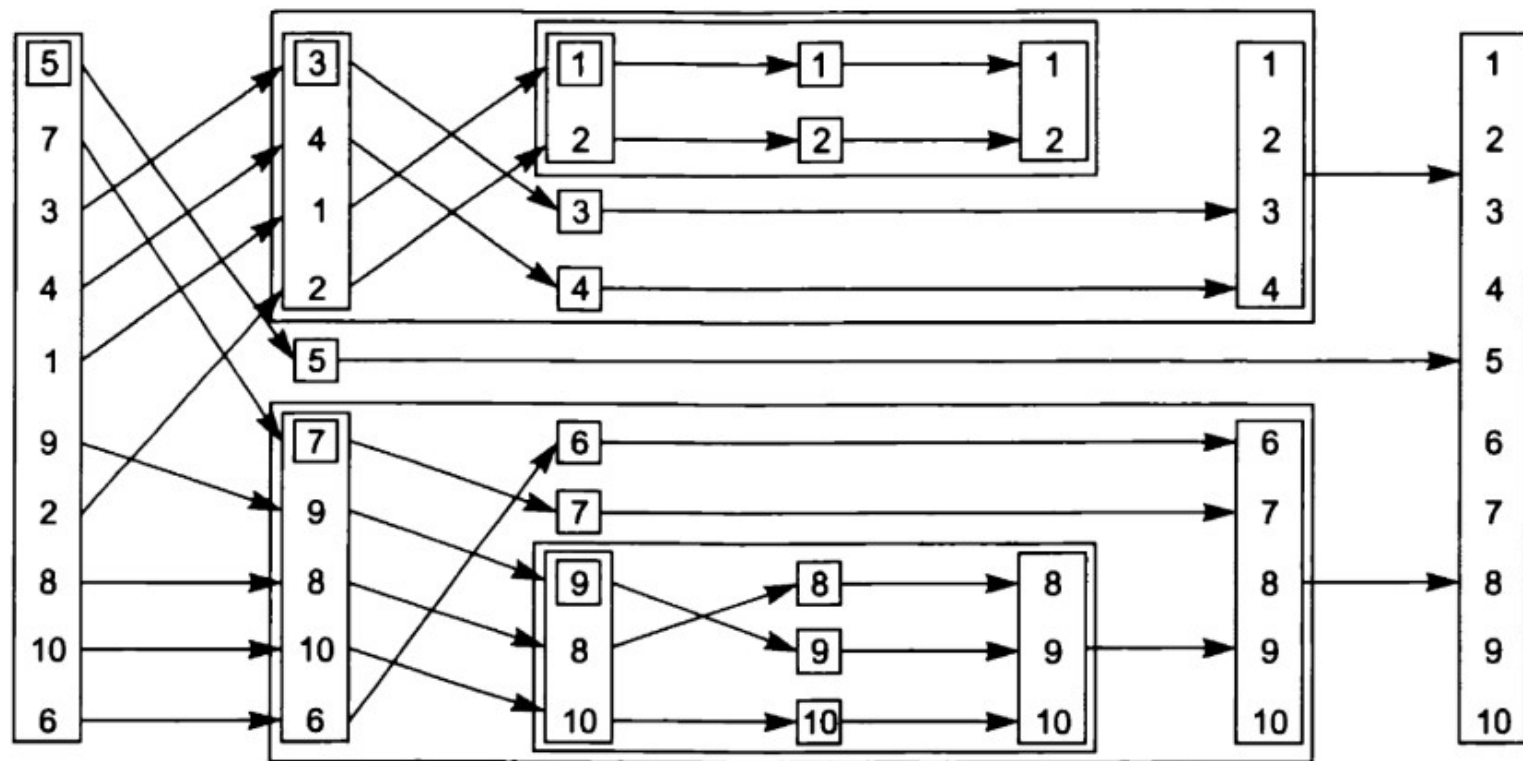
p	1	2	3	4	5
1	0	2	4	4	5
2	2	0	2	2	3
3	4	2	0	2	3
4	4	2	2	0	1
5	5	3	3	1	0

ЛАБОРАТОРНА РОБОТА №5

Назва роботи: функційне програмування.

Мета роботи: ознайомитися з парадигмою функційного програмування.

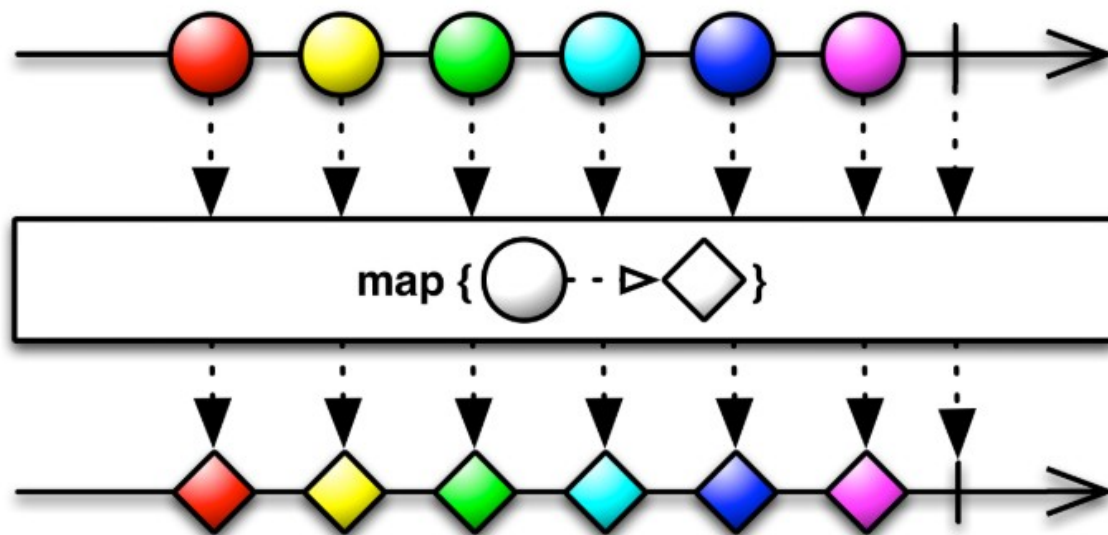




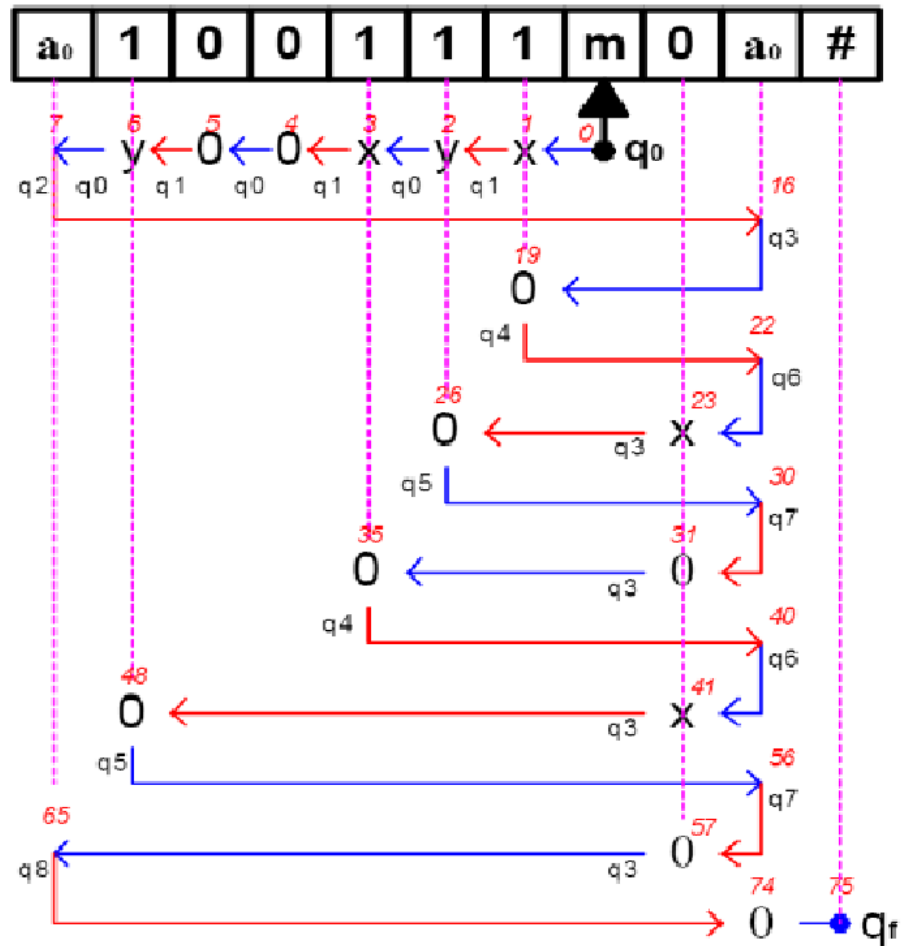
ЛАБОРАТОРНА РОБОТА №6

Назва роботи: реактивне програмування.

Мета роботи: ознайомитися з парадигмою реактивного програмування.



Формальні алгоритмічні системи (ФАС). Машина Тюрінга (МТ).



Лабораторні та практичні заняття

Графік виконання практичних та лабораторних робіт з курсу «Алгоритми та моделі обчислень» у 2022/2023 навчальному році

Тиждень	Практичні заняття	Відмітка про виконання	Лабораторні заняття	Бали						
1	Вступне заняття.	--	Вступне заняття.							
2										
3	Виконання завдання до практичного заняття №2	«зараховано»/«не зараховано»	Виконання лабораторної роботи №1.	максимум балів	5					
4				можливі оцінки	1	2	3	4	5	
5	Виконання завдання до практичного заняття №3	«зараховано»/«не зараховано»	Виконання лабораторної роботи №2.	максимум балів	5					
6				можливі оцінки	1	2	3	4	5	
7	Виконання завдання до практичного заняття №4	«зараховано»/«не зараховано»	Виконання лабораторної роботи №3.	максимум балів	5					
8				можливі оцінки	1	2	3	4	5	
9	Виконання завдання до практичного заняття №5	«зараховано»/«не зараховано»	Виконання лабораторної роботи №4.	максимум балів	5					
10				можливі оцінки	1	2	3	4	5	
11	Виконання завдання до практичного заняття №6	«зараховано»/«не зараховано»	Виконання лабораторної роботи №5.	максимум балів	5					
12				можливі оцінки	1	2	3	4	5	
13	Виконання завдання до практичного заняття №7	«зараховано»/«не зараховано»	Виконання лабораторної роботи №6.	максимум балів	5					
14				можливі оцінки	1	2	3	4	5	
15	Заключні заняття. Отримання балів за поточний контроль та допуску до іспиту.									
П.К.(Σ30)		--		30						

* у 2022/2023 навчальному році за виконання завдань на практичних заняттях ставиться відмітка «зараховано» або «не зараховано», отримання всіх відміток «зараховано» є обов'язковою умовою допуску до іспиту

<https://github.com/KozakNazar1/acm2023/blob/master/grafikACM2023LabPract.pdf>

Лінки на матеріали до лабораторних занять

Лабораторна робота №1: https://github.com/KozakNazar1/acm2023/blob/master/ACM2021_labs/acmlab1/acm2021Lab1.pdf .

Лабораторна робота №2: https://github.com/KozakNazar1/acm2023/blob/master/ACM2021_labs/acmlab2/acm2021Lab2.pdf .

Лабораторна робота №3: https://github.com/KozakNazar1/acm2023/blob/master/ACM2021_labs/acmlab3/acm2021Lab3.pdf .

Лабораторна робота №4: https://github.com/KozakNazar1/acm2023/blob/master/ACM2021_labs/acmlab4/acm2021Lab4.pdf .

Лабораторна робота №5: https://github.com/KozakNazar1/acm2023/blob/master/ACM2021_labs/acmlab5/acm2021Lab5.pdf .

Лабораторна робота №6: https://github.com/KozakNazar1/acm2023/blob/master/ACM2021_labs/acmlab6/acm2021Lab6.pdf .

Зразок коду до лабораторної роботи №1: <https://github.com/KozakNazar/srcacmlab1> .

Зразок коду до лабораторної роботи №2: <https://github.com/KozakNazar/srcacmlab2> .

Зразок коду до лабораторної роботи №3: <https://github.com/KozakNazar/srcacmlab3> .

Зразок коду до лабораторної роботи №4: <https://github.com/KozakNazar/srcacmlab4> .

Зразок коду до лабораторної роботи №5: <https://github.com/KozakNazar/srcacmlab5> .

Зразок коду до лабораторної роботи №6: <https://github.com/KozakNazar/srcacmlab6> .

Лінки на матеріали до практичних занять

Інформація до практичного заняття №1: https://github.com/KozakNazar1/acm2023/blob/master/ACM2021_practs/info/acm2021InfoForPract1.pdf .

Інформація до практичного заняття №2: https://github.com/KozakNazar1/acm2023/blob/master/ACM2021_practs/info/acm2021InfoForPract2.pdf .

Інформація до практичного заняття №3: https://github.com/KozakNazar1/acm2023/blob/master/ACM2021_practs/info/acm2021InfoForPract3.pdf .

Інформація до практичного заняття №4: https://github.com/KozakNazar1/acm2023/blob/master/ACM2021_practs/info/acm2021InfoForPract4.pdf .

Інформація до практичного заняття №5: https://github.com/KozakNazar1/acm2023/blob/master/ACM2021_practs/info/acm2021InfoForPract5.pdf .

Інформація до практичного заняття №6: https://github.com/KozakNazar1/acm2023/blob/master/ACM2021_practs/info/acm2021InfoForPract6.pdf .

Інформація до практичного заняття №7: https://github.com/KozakNazar1/acm2023/blob/master/ACM2021_practs/info/acm2021InfoForPract7.pdf .

Завдання до практичного заняття №1: завдання до практично заняття №1 відсутнє, оскільки це вступне заняття .

Завдання до практичного заняття №2: https://github.com/KozakNazar1/acm2023/blob/master/ACM2021_practs/tasks/acm2021TaskForPract2.pdf .

Завдання до практичного заняття №3: https://github.com/KozakNazar1/acm2023/blob/master/ACM2021_practs/tasks/acm2021TaskForPract3.pdf .

Завдання до практичного заняття №4: https://github.com/KozakNazar1/acm2023/blob/master/ACM2021_practs/tasks/acm2021TaskForPract4.pdf .

Завдання до практичного заняття №5: https://github.com/KozakNazar1/acm2023/blob/master/ACM2021_practs/tasks/acm2021TaskForPract5.pdf .

Завдання до практичного заняття №6: https://github.com/KozakNazar1/acm2023/blob/master/ACM2021_practs/tasks/acm2021TaskForPract6.pdf .


Завдання до практичного заняття №7: https://github.com/KozakNazar1/acm2023/blob/master/ACM2021_practs/tasks/acm2021TaskForPract7.pdf .


Лекційні домашні та контрольні завдання з курсу АМО

Задача Дойча. Задача Дойча-Джозі. Задача Бернштайна-Вазірані. Задача Саймона.

7.4. Важливі алгоритми квантових обчислень.

Алгоритм Шора. Алгоритм Гровера.


 матеріали попередніх навчальних років

 Квантові обчислення

Контрольні роботи лекційних занять

 Лекція №1

 Пробне контрольне завдання №1


 Пробне контрольне завдання №2




Домашні завдання, що прив'язані до лекційних занять

Перейти до навігації по списку методичного забезпечення курсу

 HW1 (за темою лекції №2)

 Домашнє завдання №1

 hw1 source code

 Завантаження звітів до домашнього завдання №1

Домашні завдання №1 та №2

(1.2.6. Формальні граматика та формальні мови. (1.3.))

З використанням `std::regex` розробити програму, яка шукає в тексті лексеми заданого формату.

Розробити програму, яка за допомогою `boost::spirit` реалізовує заданий відповідно до варіанту синтаксис для запису виразу.

Домашні завдання №3 та №4

(1.2.2. Формалізація поняття алгоритму. Формальні алгоритмічні системи(ФАС).)

З використанням реалізації поведінки детермінованого кінцевого автомату написати на C/C++ програму(*англ. automata-based programming*) виявлення ділянки тексту у вхідній стрічці. Програма має лише відображати перехід моделі автомату у кінцевий стан без виявлення позиції входження шуканої ділянки тексту.

Виконати реалізацію моделі машини Тюрінга на C/C++ для виконання обчислень згідно власного варіанту.

Домашні завдання №5, №7 та №8

(2. 2. Синтез алгоритмів.)

Скласти програму (C/C++), яка дозволяє знайти невідоме значення x методом повного перебору для заданих констант a, b, c, d, e, f, g та h та порівняти його з обчисленням.

Скласти програму (C/C++), яка з застосуванням алгоритмічної стратегії «перебір з поверненням» (англ. *backtracking*) дозволяє виконати таке розміщення фігур ферзів на шахівниці, що жодна з них не ставить під удар іншу.

Скласти програму (C/C++), яка з застосуванням алгоритмічної стратегії «динамічне програмування» дозволяє виконати такі два завдання:

1. Менеджмент ІТ-компанії розглядає можливість старту нових напрямків. Усі нові напрямки за прибутковістю можна розділити на T типів. Кожна група певного типу має складатися з N_i розробників та орієнтовно може приносити компанії прибуток K_i . Сформуванати оптимальну кількість команд відповідно до напрямків, якщо компанії вдалося найняти M нових розробників.

2. Існує T номіналів монет. Видати здачу M мінімальною кількістю монет.

Для порівняння повторно виконати завдання за допомогою «прямого перебору».

Домашні завдання №12, №9, №6 та №10

(4.1. Алгоритми пошуку. 4.2. Алгоритми сортування даних. + Тема №3)

Скласти програму (C/C++), яка виконує імплементацію хеш-таблиці(*хеш-таблиці, англ. Hash table*) без застосування готових реалізацій.

Для алгоритму, що заданий за допомогою функції мовою C, розрахувати обчислювальну складність для «найгіршого випадку», сформулювати блок-схему алгоритму та перевірити коректність його роботи.

Скласти програму (C/C++), яка дозволяє відсортувати вхідні величини методом злиття. Безпосередньо сортування методом злиття виконується при об'єднанні частин вхідних даних, які в свою чергу сортуються за допомогою швидкого сортування з стандартної бібліотеки мови C. Кількість частин для злиття та розмір вхідного масиву даних обрати відповідно до варіанту.

Скласти програму (C/C++), яка за допомогою швидкого сортування(з стандартної бібліотеки мови C) дозволяє з вхідного тексту вивести слова, що починаються з заданої літери(решта слів вивести в алфавітному порядку). Сортування потрібно виконати без використання додаткової пам'яті, дозволяється використовувати тільки масив, що зберігає вказівники на початок слів.

Домашнє завдання №11

(4.3. Алгоритми порівняння зі взірцем.)

Написати на C/C++ програму, яка реалізовує алгоритм Бояра-Мура для знаходження у вхідній стрічці всіх входжень заданого слова.

Домашні завдання №13 та №14

(4.4. Чисельні алгоритми.)

Скласти програму (C/C++), що виконує множення матриць. Для виконання завдання замість простого двовимірного масиву використати власну реалізацію.

Написати на C/C++ реалізацію довгих чисел на основі списку(елементи списку – десяткові цифри) з використанням вказівників. Реалізувати дію згідно варіанту.

Домашні завдання №15 та №16

(4.5. Графи та мережеві алгоритми.)

Скласти програму (C/C++), яка дозволяє знаходити мінімальне кістякове дерево(*англ. minimum spanning tree*) для заданого графу за допомогою алгоритму Крускала.

Скласти програму (C/C++), яка дозволяє знаходити у графі мінімальний шлях від заданої вершини до інших вершин за допомогою алгоритму Дейкстри.

Домашні завдання
№17, №18, №19, №20 та №20б

Домашні завдання №17 та №18

Застосовуючи STL скласти програму (C/C++), яка за допомогою *std::sort* дозволяє з вхідного тексту(*std::string*) вивести слова, що починаються з заданої літери(решта слів вивести в алфавітному порядку). Сортування потрібно виконати без використання додаткової пам'яті, дозволяється використовувати тільки *std::vector*, що зберігає індекси на початок слів. Додатково потрібно зберегти результати роботи програми у *std::map*.

Індекси із *std::vector* вивести на екран за допомогою *std::copy*, а слова за допомогою звичайного циклу *for*. Для виводу даних з *std::map* застосувати *std::transform*.

Замінивши використання *std::vector* на *boost::container::vector*, а *std::map* на *boost::container::map*, виконати домашнє завдання №17 повторно.

Домашнє завдання №19

Застосовуючи JCL скласти програму (Java), яка за допомогою *Collections.sort* дозволяє з вхідного тексту(*String*) вивести слова, що починаються з заданої літери(решта слів вивести в алфавітному порядку). Сортування потрібно виконати без використання додаткової пам'яті, дозволяється використовувати тільки *ArrayList*, що зберігає індекси на початок слів. Додатково потрібно зберегти результати роботи програми у *HashMap*.

Застосувати різні способи для виводу результатів роботи програми.

Домашні завдання №20а та №20б

Застосовуючи FCL скласти програму (C#), яка за допомогою *List<T>.Sort* дозволяє з вхідного тексту(*String*) вивести слова, що починаються з заданої літери(решта слів вивести в алфавітному порядку). Сортування потрібно виконати без використання додаткової пам'яті, дозволяється використовувати тільки *List*, що зберігає індекси на початок слів. Додатково потрібно зберегти результати роботи програми у *Dictionary*.

Застосувати різні способи для виводу результатів роботи програми.

C++/CLI дозволяє писати програми, які водночас можуть містити і звичайний некерований код(*англ. unmanaged code*) написаний на C++ для компіляції безпосередньо в машинний код, і керований код (*англ. managed code*) написаний на C++ для .NET.

В якості самостійної роботи пропонується виконати домашнє завдання №20 повторно за допомогою C++/CLI.

Домашні завдання №21, №22а та №22б

Домашнє завдання №21

Написати на C++ програму для розв'язання системи лінійних алгебраїчних рівнянь (СЛАР) з використанням бібліотеки *uBLAS* (з набору бібліотек *Boost*).

СЛАР			Макровизначення
$\begin{cases} x_1 + 5x_2 + 3x_3 - 4x_4 = 20 \\ 3x_1 + x_2 - 2x_3 = 9 \\ 5x_1 - 7x_2 + 10x_4 = -9 \\ 3x_2 - 5x_3 = 1 \end{cases}$	=>	$\begin{pmatrix} 1 & 5 & 3 & -4 \\ 3 & 1 & -2 & 0 \\ 5 & -7 & 0 & 10 \\ 0 & 3 & -5 & 0 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 20 \\ 9 \\ -9 \\ 1 \end{pmatrix}$	
		$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$	<pre>#define EQUATIONS_COUNT 4</pre>
		$\begin{pmatrix} 1 & 5 & 3 & -4 \\ 3 & 1 & -2 & 0 \\ 5 & -7 & 0 & 10 \\ 0 & 3 & -5 & 0 \end{pmatrix}$	<pre>#define COEFFICIENTS \ 1.0, 5.0, 3.0, -4.0, \ 3.0, 1.0, -2.0, 0.0, \ 5.0, -7.0, 0.0, 10.0, \ 0.0, 3.0, -5.0, 0.0</pre>
		$\begin{pmatrix} 20 \\ 9 \\ -9 \\ 1 \end{pmatrix}$	<pre>#define CONSTANT_TERMS \ 20.0, \ 9.0, \ -9.0, \ 1.0</pre>

Домашні завдання №22а та №22б

Застосовуючи OpenCV скласти програму (Python), яка дозволяє виявляти на фото обличчя людей. При цьому потрібно виявляти не більше MAX_FACES_COUNT облич.

В якості самостійної роботи пропонується виконати домашнє завдання №22 повторно за допомогою C++.

Домашні завдання
№23а, №23б1, №23б2, №23в, №24
(та лабораторна робота №5)

Домашні завдання №23а, №23б1 та №23б2

23 а) Застосовуючи парадигму функційного програмування скласти програму мовою Haskell, яка виконує імплементацію швидкого сортування без використання готових бібліотечних реалізацій.

23 б) Одною з альтернатив Haskell при використанні парадигми функційного програмування є Erlang/Elixir.

Загалом, якщо Haskell можна вважати базовою стандартизованою мовою при застосуванні парадигми функційного програмування, то Erlang та Elixir володіють значною практичною цінністю. Це пов'язано з тим, що для них доступні потужні Web-фреймворки, які необхідні для швидкого написання сучасного масового програмного забезпечення. Також отриманий байт-код після трансляції коду мовою Elixir виконується на віртуальній машині Erlang (BEAM), тому Elixir має сумісність з фреймворком Erlang/OTP та іншими бібліотеками і фреймворками мови Erlang.

В якості самостійної роботи пропонується виконати домашнє завдання №23 повторно за допомогою Erlang або Elixir без використання готових бібліотечних реалізацій.

Домашні завдання №23в та №24

24) Виконати домашнє завдання №23 повторно за допомогою мови Java без використання готових бібліотечних реалізацій.

23 в) Мова Kotlin це намагання спростити програмування для платформи Java.

Байт-код після трансляції коду мовою Kotlin або мовою Scala виконується на віртуальній машині Java (JVM), тому Java/Scala/Kotlin можна віднести до спільної категорії засобів програмування.

Особливою популярністю мова Kotlin користується при створенні Android-додатків, де замість JVM використовується Dalvik/ART.

В якості самостійної роботи пропонується виконати домашнє завдання №23 повторно за допомогою Kotlin без використання готових бібліотечних реалізацій.

Домашні завдання

№25а, №25б, №25в, №26, №27а, №27б, №28
(та лабораторна робота №6)

Домашнє завдання №25а

Застосовуючи елементи реактивної парадигми програмування(код має містити в явній формі Observable та Observer) за допомогою RxJS(реалізація ReactiveX для JavaScript) скласти програму (мовою JavaScript для платформи NodeJS), яка дозволяє виконувати валідацію ключа(25 шістнадцяткових цифр) ліцензії для деякого програмного продукту. Для зручності вводу програма має групувати шістнадцяткові цифри по п'ять цифр та відображас 'A', 'B', 'C', 'D', 'E' та 'F' завжди у верхньому регістрі. Окрім шістнадцяткових цифр та символів пробілу і табуляції(що трактуються як одна цифра 0), програма валідації не дозволяє вводити жодних інших символів, які не належать шістнадцятковій системі числення. При цьому надається не більше ATTEMPTS_COUNT спроб вводу ключа ліцензії. Між ітераціями сусідніх спроб введені значення мають зберігатися для редагування у наступній спробі.

Ключ ліцензії «11111 22222 33333 44444 55555» має міститися в коді програми валідації у неявній формі.

** акцентується увага на тому, що завдання має бути виконано за допомогою RxJS відповідно до парадигми реактивного програмування, а не подійно-орієнтованої парадигми, на основі якої працює рушій платформи NodeJS*

Домашнє завдання №256

Виконати домашнє завдання №25 повторно за допомогою мови C++ використовуючи RxCpp(реалізація ReactiveX для C++).

Домашнє завдання №25в

(додаткова самотійна робота)

Хоча React і не застосовує парадигму реактивного програмування, але в якості самотійної роботи пропонується виконати завдання №25 повторно за допомогою React та з використанням Redux.

Домашнє завдання №26

Виконати домашнє завдання №25 повторно за допомогою мови Python використовуючи RxPY(реалізація ReactiveX для Python)

Домашнє завдання №27а

Виконати домашнє завдання №25 повторно(без використання RxJS) за допомогою мови JavaScript для платформи NodeJS(застосовуючи тільки руніч цієї платформи).

** код домашнього завдання №25 та домашнього завдання №27 має відрізнятися тільки кількома останніми стрічками, які в першому випадку показують використання реактивної парадигми програмування, а в другому випадку – подійно-орієнтованої*

Домашнє завдання №28

Виконати домашнє завдання №25 повторно як альтернативну низькорівневу реалізацію домашнього завдання №27 мовою C. Для цього використати poll(системний виклик ОС Linux) з відстеженням POLLIN у revents з структури pollfd при передачі даних за допомогою неіменованого каналу(pipe) до обробника вводу(в прикладі він називається inputHandler). (Це імітує `process.stdin.on('keypress', inputHandler)` з прикладу коду домашнього завдання №27). Саму передачу даних виконує додатковий скануючий обробник(в прикладі він називається stdinInputHandler), який виступає в ролі аналога поведінки вбудованої реалізації `process.stdin` з NodeJS.

Замість використання системного виклику poll також можна використати системний виклик select, тоді відстежуватися буде безпосередньо неіменований канал.

Домашнє завдання №276

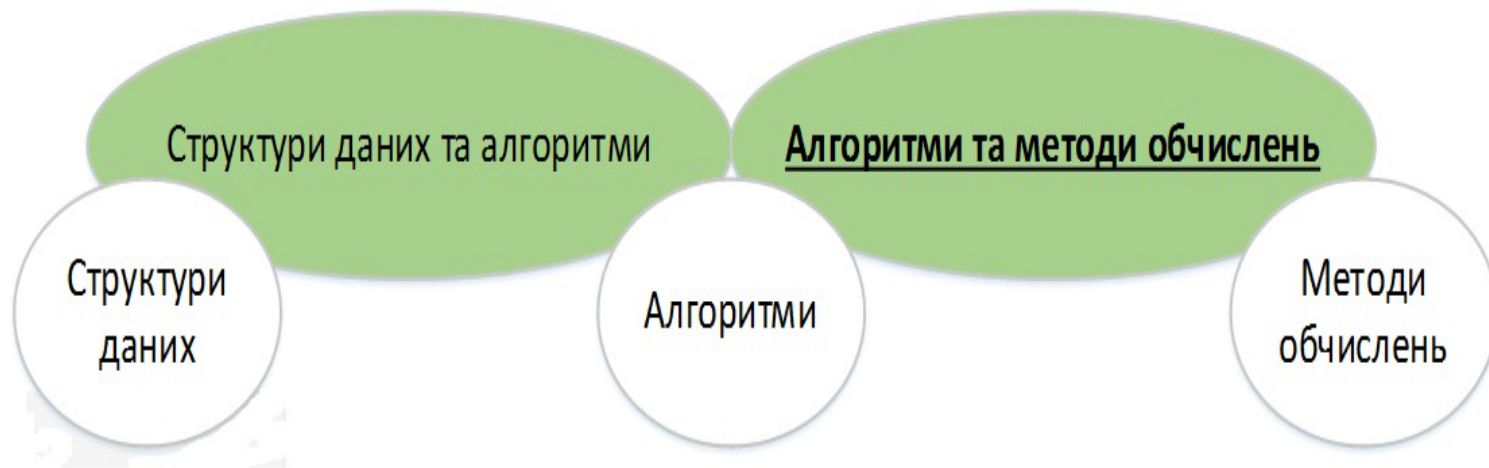
У 28 домашньому завданні потрібно було виконати домашнє завдання №25 повторно як альтернативну низькорівневу реалізацію домашнього завдання №27 мовою C. Сам рушій платформи NodeJS побудований на основі бібліотеки libuv, яка була створена для заміни libeio(імплементує Tread Pool) та libev(імплементує Event Loop). *(Рушій JavaScript для NodeJS це V8, але рушієм подійно-орієнтованої парадигми у NodeJS є саме libuv)*. В якості самостійної роботи пропонується знову виконати домашнє завдання №25 повторно як альтернативну низькорівневу реалізацію домашнього завдання №27 мовою C, але на цей раз за допомогою бібліотеки libuv.



<https://repl.it/languages/cpp>



<https://www.tutorialspoint.com/codingground.htm>



Алгоритми та моделі обчислень

Лекція 1

Тема №1. Вступ до теорії алгоритмів.

1.1. Неформальне тлумачення алгоритму.

