Тема 1: Основи аналізу алгоритмів

§1. Зміст дисципліни "Теорія алгоритмів", її зв’язок із іншими дисциплінами.
§2. Поняття алгоритму. Способи задання.
§3. Властивості алгоритмів
§4. Етапи проектування і аналізу алгоритмів
§5.Основи аналізу ефективності алгоритмів
§6. Ефективність алгоритму в різних випадках

§1. Зміст дисципліни "Теорія алгоритмів", її зв’язок із іншими дисциплінами.

Теорія алгоритмів – це наука, що вивчає загальні властивості і закономірності алгоритмів та різноманітні формальні моделі їх представлення. На основі формалізації поняття алгоритму можливе порівняння алгоритмів за їх ефективністю, перевірка їх на еквівалентність, визначення областей застосування. Теорія алгоритмів утворює теоретичний фундамент обчислювальних наук. Застосування теорії алгоритмів здійснюється як у використанні самих результатів (особливо це стосується використання розроблених алгоритмів), так і у виявленні нових понять і уточненні старих. З її допомогою проясняються такі поняття як довідність, ефективність, можливість розв'язання і інші.

***Мета дисципліни*:** вивчення змістовних основ математичного апарату теорії алгоритмів та представлення знань потрібних для постановки та розв'язування задач аналізу та побудови оптимальних, за вибраними критеріями певної предметної області, інформаційних та управляючих систем.

***Задачі дисципліни*:**

- ознайомлення слухачів з основами теорії алгоритмів, з методами оцінки складності алгоритмів і розробки ефективних алгоритмів.
- формалізація поняття «алгоритм» і дослідження формальних алгоритмічних систем;
- формалізація доказу алгоритмічної нерозв'язності ряду задач;
- проведення асимптотичного аналізу складності алгоритмів;
- одержання явних функцій трудомісткості з метою порівняльного аналізу алгоритмів;
- дослідження й аналіз рекурсивних алгоритмів;
- розробка критеріїв порівняльної оцінки якості алгоритмів;
- Ознайомлення з базовими алгоритмами.

Аль-Хорезмі (786-850 рр. н.е.) *- засновник алгебри, від його імені стався термін «алгоритм». У світовій науці він був відомий своїм трактатом з математики, заснованому на позиційному принципі.*

Згодом слово «алгоритм» стало позначати кожен регулярний процес за кінцеве число кроків дає рішення певного класу завдань.

§2. Поняття алгоритму. Способи задання алгоритмів

Алгоритм – це послідовність чітко визначених інструкцій призначених для розв’язку деякої задачі. Іншими словами, це послідовність команд, які з конкретних вхідних даних дозволяють отримати необхідні вихідні дані за обмежений проміжок часу.

Алгоритм вважається ***правильним***, якщо при будь-яких припустимих даних він закінчує роботу і видає результат, що задовольняє вимогам задачі.

Алгоритм ***однозначний***, якщо при застосуванні до тих самих вхідних даних він дає той самий результат.

Числовий алгоритм – це алгоритм, який зводить розв’язання задачі до арифметичних дій над числами.

Алгоритм, що містить розпорядження, яке стосується не цифр, а об’єктів будь-якої природи називається ***логічним алгоритмом***.

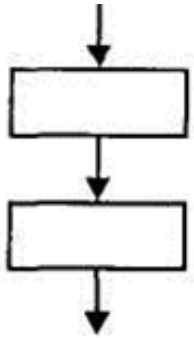
Виділяють три класи алгоритмів: обчислювальні, інформаційні і керуючі.

Обчислювальні алгоритми, як правило, працюють із простими видами даних (числа, матриці), але сам процес обчислення може бути довгим і складним.

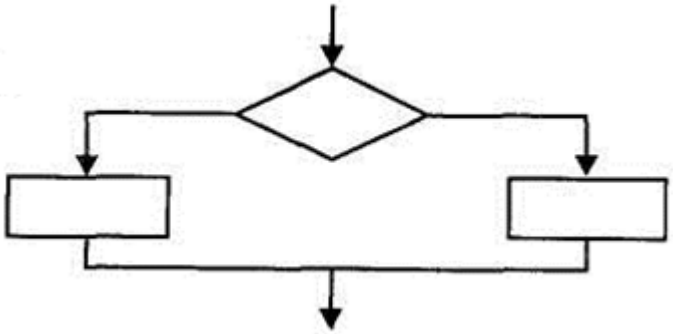
Інформаційні алгоритми являють собою набір порівняно невеликих процедур (наприклад, пошук числа, слова), але працюючих з великими обсягами інформації (бази даних). Для того щоб вони працювали ефективно, важливо мати гарну організацію даних.

Керуючі алгоритми характерні тим, що дані до них надходять від зовнішніх процесів, якими вони керують. Результатом роботи цих алгоритмів є різні керуючі впливи.

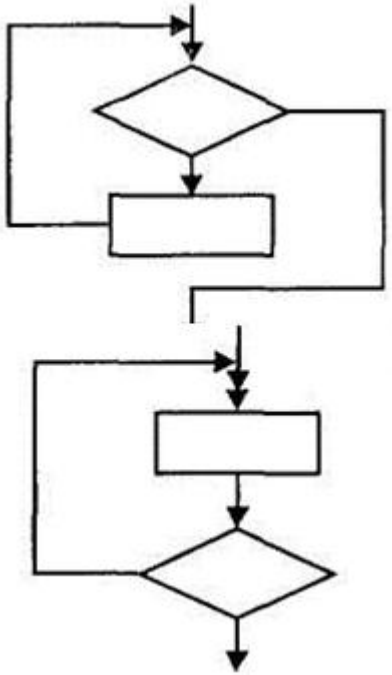
Типи процесів оброблення інформації



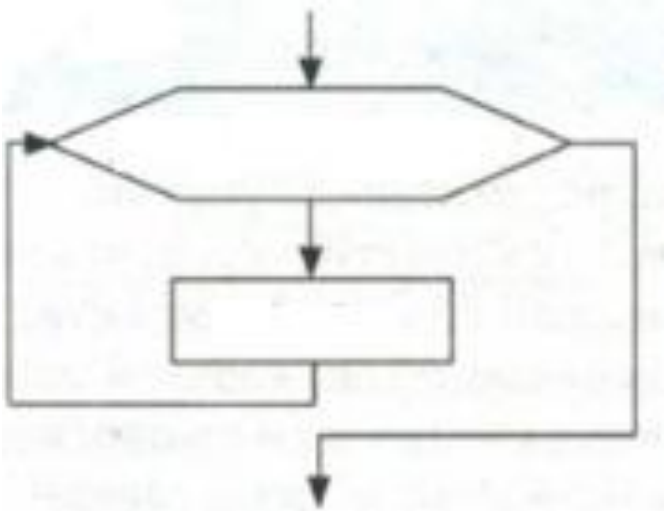
Лінійний процес (слідування) - оброблення інформації здійснюється послідовно, одна дія за іншою, і кожен етап алгоритму виконується тільки один раз.



Розгалужувальний процес (альтернатива) - інформація обробляється по одному із двох можливих шляхів, тобто ті чи інші дії виконуються залежно від деякої умови.



При *циклічному процесі* ті самі дії з оброблення інформації потрібно виконати багаторазово. Розрізняють цикли з передумовою та постумовою.



Реальний алгоритм будь-якого ступеня складності можна задати комбінацією зазначених базових процесів.

Способи задання алгоритмів

Існують різні способи опису алгоритмів (словесний, табличний, псевдокод, графічний тощо). Алгоритм для ЕОМ найзручніше зображати графічно у вигляді схем.

1. Словесний спосіб опису алгоритму

Рецепт "Піца нашвидкуруч":

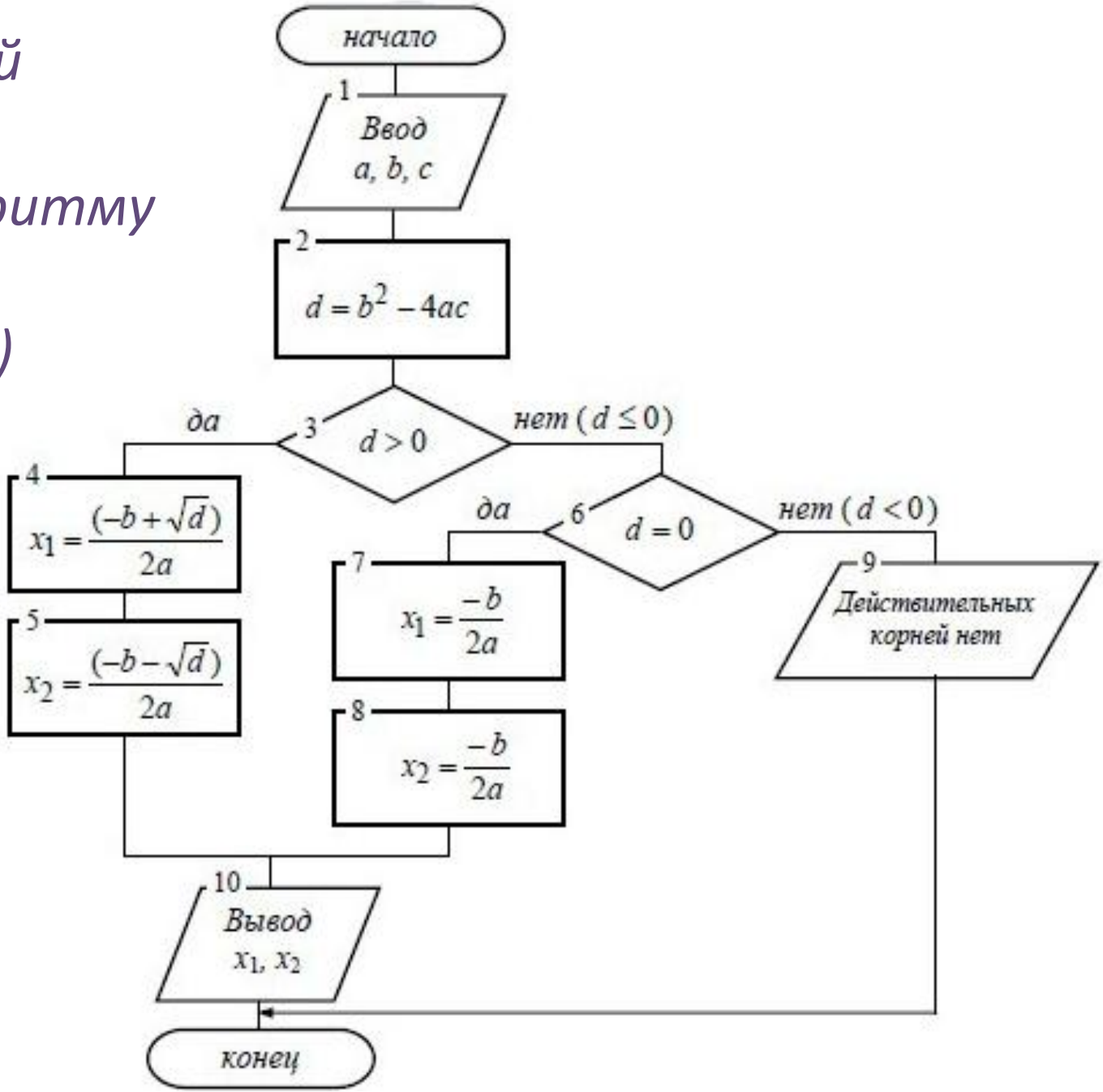
1. В рівних пропорціях змішайте сметану з томатною пастою, за бажанням додайте орегано чи інші спеції.
2. Отриманий соус намажте на хліб.
3. Зверху викладіть порізані на шматочки помідори та болгарський перець, половинки маслин. Або інші улюблені варіанти начинки.
4. Натертий чи нарізаний скибочками сир покласти зверху.
5. На кілька хвилин поставити отримане в духовку, аерогриль чи мікрохвильовку.

2. Табличний спосіб опису алгоритму

Таблиця розрахунку заробітної плати

№	Прізвище	Місячна ставка	Робочих днів в місяці	Денний заробіток	Відпрацьовані дні	Заробітна платня за місяць	Сума податку (23%)	До видачі
1	2	3	4	5	6	7	8	9
(1)	(2)	(3)	(4)	(3):(4)	(6)	(7)*(6)	(7)*0,23	(7-8)
1	Асєєв Б.	420	24	17,5	24	420	96,6	323,4
2	Волін О.	420	24	17,5	20	350	80,5	269,5
3	Котик І.	360	24	15,0	23	345	79,35	265,65

3. Графічний спосіб опису алгоритму (схема алгоритму)



4. Псевдокод

Алгоритм Дейкстры для нахождения кратчайших путей.

Вход: граф $G(V, E)$ (ориентированный или нет) с неотрицательными длинами рёбер $\{l_e : e \in E\}$; вершина $s \in V$.

Выход: Выход: для всех вершин u, достижимых из s, $dist[u]$ будет равно расстоянию от s до u (и ∞ для недостижимых).

```
procedure DISJKSTRA(G, l, s)
    for всех вершин u ∈ V do
        dist[u] ← ∞
        prev[u] ← nil
    end for
    dist[s] ← 0
    H ← MAKE-QUEUE(V)          ▷ в качестве ключей используются dist
    while H ≠ ∅ do
        u ← DELETE-MIN(H)
        for всех рёбер (u, v) ∈ E do
            if dist[v] > dist[u] + l(u, v) then
                dist[v] ← dist[u] + l(u, v)
                prev[v] ← u
                DECREASE-KEY(H, v, dist[v])
            end if
        end for
    end while
end procedure
```

§3. Властивості алгоритмів

Детермінованість — визначення кроків алгоритму, тобто після кожного кроку або зазначається, який крок слід робити далі, або дається команда на зупинку, після чого робота алгоритму вважається закінченою.

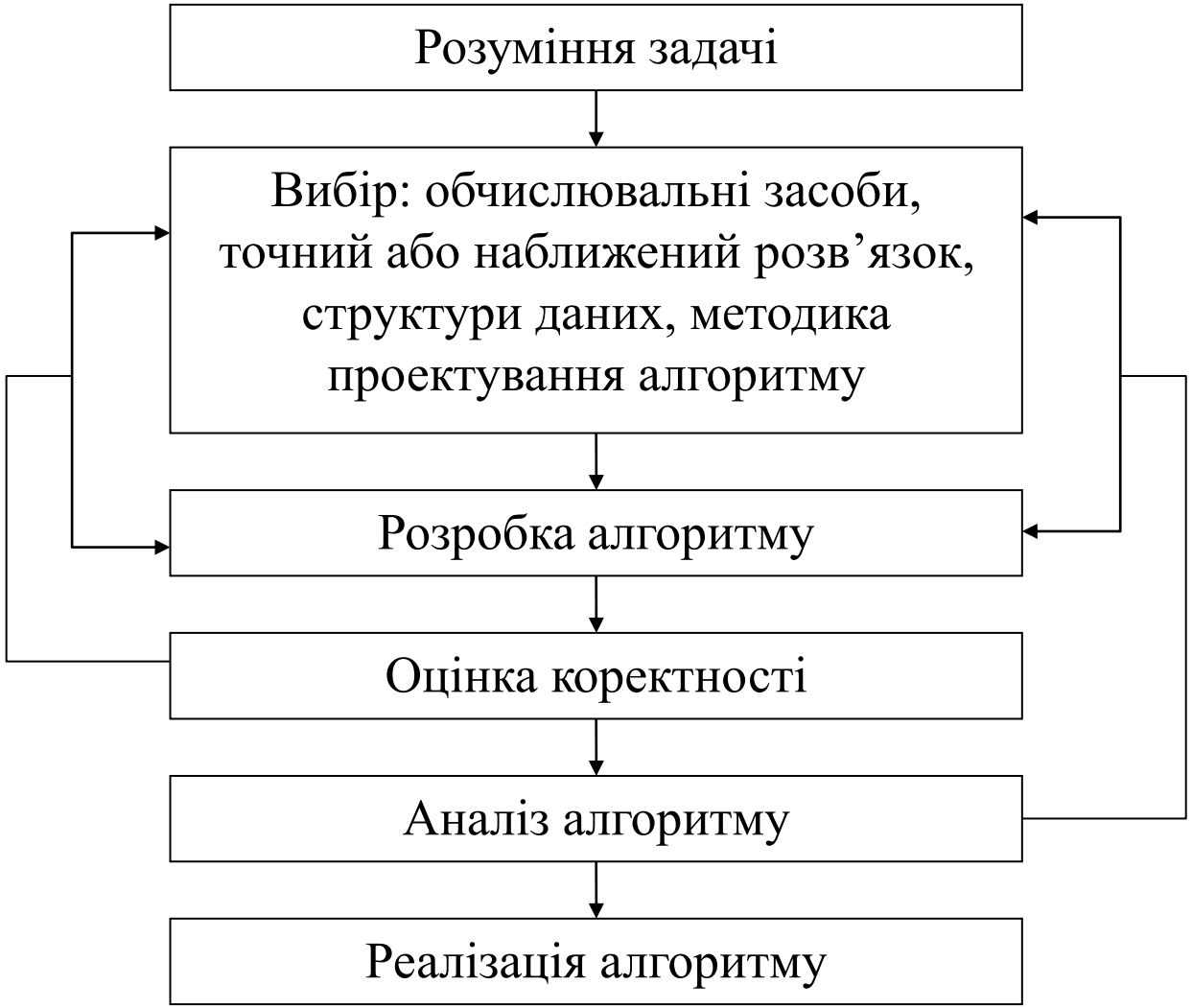
Результативність (скінченність, збіжність) — виконання алгоритму має або закінчуватися результатом, або інформацією про те, чому не може бути одержаний результат, причому множина різних кроків, з яких складено алгоритм, є скінченою.

Масовість — алгоритм може бути використаний для розв'язання цілого класу задач одного типу.

Зрозумілість — алгоритм має бути зрозумілим конкретному виконавцю, який повинен виконати кожну команду алгоритму у строгій відповідності з її призначенням.

Дискретність — можливість розбиття алгоритму на скінчену кількість етапів, причому результати попереднього етапу є вхідними для наступного.

§4. Етапи проектування і аналізу алгоритмів



§5.Основи аналізу ефективності алгоритмів

Часова ефективність алгоритму визначається як функція від розміру даних, шляхом підрахунку кількості виконаних базових операцій.

Базова операція – це операція, яка робить основний внесок в загальний час виконання алгоритму. Зазвичай це операція з найбільшим часом роботи в найбільш глибоко вкладеному циклі.

В залежності від впливу вхідних даних на функцію ефективності алгоритму алгоритми бувають:

1. Кількісно-залежні по ефективності алгоритми

Це алгоритми, функція трудомісткості яких залежить тільки від розмірності конкретного входу, і не залежить від конкретних значень.

2. Параметрично-залежні по ефективності алгоритми

Це алгоритми, ефективності яких визначаються не розмірністю входу, а конкретними значеннями оброблюваних слів пам'яті.

3. Кількісно-параметричні по ефективності алгоритми

Це алгоритми, ефективності яких залежить як від кількості даних на вході, так і від значень вхідних даних, у цьому випадку.

4. Порядково-залежні по ефективності алгоритми

Це алгоритми, в яких кількість операцій залежить від порядку розташування вихідних об'єктів.

§6. Ефективність алгоритму в різних випадках

Під ефективністю алгоритму в *найгіршому випадку* розуміють ефективність для найгіршої сукупності вхідних даних розміром *n* (час роботи алгоритму найбільший).

Під ефективністю алгоритму в *найкращому випадку* розуміють ефективність для найкращої сукупності вхідних даних розміром *n* (час роботи алгоритму найменший).

Приклад: Пошук заданого елемента (ключа пошуку *k*) в списку з *n* елементів, шляхом послідовного порівняння ключа *k* з кожним із елементів списку. Робота алгоритму завершується коли заданий ключ або знайдено, або ні і список при цьому закінчився.

Program Search

Вхідні дані: массив чисел $A[0...n-1]$, *k*.

Вихідні дані: індекс першого елементу масиву *A*, який дорівнює *k* або –1, якщо такого елемента не знайдено.

$i \leftarrow 0$

while $i<n$ **and** $A[i] \neq k$ **do**

$i \leftarrow i+1$

if $i<n$

return *i*

else

return –1

Для даного прикладу найгірший випадок, коли в списку немає шуканого елемента або він на останньому місці:

$$C_{worst}(n) = n$$

$C(n)$ – кількість разів, які дана операція повинна виконатись при роботі алгоритму.

Найкращий випадок буде тоді, коли елемент рівний ключу *k* буде першим:

$$C_{best}(n) = 1$$

Виконаємо аналіз алгоритму для середнього випадку.

Для цього зробимо декілька припущень:

– ймовірність успішного пошуку *p*, $0 \le p \le 1$

– ймовірність першого співпадання ключа з *i*-м елементом списку однакова для будь-якого *i*.

Обчислимо середню кількість операцій $C_{avg}(n)$ (averege-case) порівняння з ключем k.

Якщо співпадіння з ключем знайдене, ймовірність того, що це вийшло саме на i-му елементі списку, дорівнює p/n для будь-якого i. При цьому кількість операцій порівняння, які виконує алгоритм в подібному випадку, дорівнює i.

Якщо співпадіння з ключем не знайдено, кількість операцій порівняння дорівнює n, а ймовірність цієї події дорівнює $(1-p)$.

$$C_{avg}(n) = \left[1 \cdot \frac{p}{n} + 2 \cdot \frac{p}{n} + \ldots + i \cdot \frac{p}{n} + \ldots + n \cdot \frac{p}{n}\right] + n(1-p) =$$

$$= \frac{p}{n}\left[1 + 2 + \ldots + i + \ldots + n\right] + n(1-p) = \frac{p}{n} \cdot \frac{n(n+1)}{2} + n(1-p) = \frac{p(n+1)}{2} + n(1-p)$$

Якщо p = 1 (шуканий елемент точно присутній в списку), то

$$C_{avg}(n) = \frac{n+1}{2}$$

Якщо p = 0 (шуканий елемент відсутній в списку), то

$$C_{avg}(n) = n$$