# KIẾN THỨC TỔNG HỢP - DÀI

## TOÁN HỌC

### Đại số

#### Phương trình và bất phương trình

**Phương trình bậc 2**: ax² + bx + c = 0 (a ≠ 0)

* Biệt thức: Δ = b² - 4ac
  + Δ > 0: hai nghiệm phân biệt x₁,₂ = (-b ± √Δ)/(2a)
  + Δ = 0: nghiệm kép x = -b/(2a)
  + Δ < 0: vô nghiệm trong R
* Định lý Viète: x₁ + x₂ = -b/a, x₁·x₂ = c/a
* Ứng dụng: phân tích thành nhân tử, tìm GTLN/GTNN

**Phương trình bậc cao**:

* Phương trình trùng phương: ax⁴ + bx² + c = 0
* Phương trình có chứa căn: đặt điều kiện, bình phương hai vế
* Phương trình mũ: a^x = b ⟺ x = log\_a(b)
* Phương trình logarit: log\_a(x) = b ⟺ x = a^b

**Hệ phương trình**:

* Hệ tuyến tính: phương pháp thế, cộng đại số, định thức Cramer
* Hệ phi tuyến: phương pháp thế, đặt ẩn phụ

#### Hàm số và đồ thị

**Hàm số bậc nhất**: y = ax + b (a ≠ 0)

* Đồ thị: đường thẳng có hệ số góc a, cắt Oy tại (0,b)
* Tính đồng biến (a > 0) hoặc nghịch biến (a < 0)

**Hàm số bậc 2**: y = ax² + bx + c (a ≠ 0)

* Đồ thị: parabol có đỉnh I(-b/(2a), -Δ/(4a))
* Trục đối xứng: x = -b/(2a)
* Hướng miệng: a > 0 (lên), a < 0 (xuống)
* GTLN/GTNN trên đoạn

**Hàm số mũ và logarit**:

* y = a^x: tập xác định R, tập giá trị (0,+∞)
* y = log\_a(x): tập xác định (0,+∞), tập giá trị R
* Tính chất: log\_a(xy) = log\_a(x) + log\_a(y), log\_a(x/y) = log\_a(x) - log\_a(y)

#### Dãy số và cấp số

**Cấp số cộng**: u\_n = u₁ + (n-1)d

* Số hạng tổng quát, tổng n số hạng đầu: S\_n = n(u₁ + u\_n)/2

**Cấp số nhân**: u\_n = u₁ · q^(n-1)

* Tổng n số hạng đầu: S\_n = u₁(q^n - 1)/(q - 1) (q ≠ 1)

### Hình học

#### Hình học phẳng

**Tam giác**:

* Định lý sin: a/sin(A) = b/sin(B) = c/sin(C) = 2R
* Định lý cosin: c² = a² + b² - 2ab·cos(C)
* Diện tích: S = (1/2)ab·sin(C) = √[p(p-a)(p-b)(p-c)] (Heron)
* Đường cao, trung tuyến, phân giác

**Đường tròn**:

* Phương trình: (x-a)² + (y-b)² = r²
* Tiếp tuyến: khoảng cách từ tâm đến đường thẳng = bán kính
* Góc nội tiếp, góc tạo bởi tia tiếp tuyến và dây cung

**Hình học tọa độ**:

* Khoảng cách hai điểm: d = √[(x₂-x₁)² + (y₂-y₁)²]
* Phương trình đường thẳng: ax + by + c = 0
* Góc giữa hai đường thẳng, khoảng cách từ điểm đến đường thẳng

#### Hình học không gian

**Khối đa diện**:

* Hình hộp chữ nhật: V = abc, S\_tp = 2(ab + bc + ca)
* Lăng trụ: V = S\_đáy · h
* Chóp: V = (1/3)S\_đáy · h

**Mặt cầu và khối cầu**:

* Diện tích mặt cầu: S = 4πr²
* Thể tích khối cầu: V = (4/3)πr³
* Mặt phẳng cắt mặt cầu

### Giải tích

#### Giới hạn

* Giới hạn hàm số: lim[x→a] f(x) = L
* Quy tắc tính: giới hạn của tổng, hiệu, tích, thương
* Các dạng vô định: 0/0, ∞/∞, ∞-∞

#### Đạo hàm

**Định nghĩa**: f'(x) = lim[h→0] [f(x+h) - f(x)]/h **Quy tắc tính**:

* (u ± v)' = u' ± v'
* (uv)' = u'v + uv'
* (u/v)' = (u'v - uv')/v²
* Đạo hàm hàm hợp: (f(g(x)))' = f'(g(x))·g'(x)

**Ứng dụng**:

* Khảo sát hàm số: tính đơn điệu, cực trị
* Tiếp tuyến: y - y₀ = f'(x₀)(x - x₀)
* Bài toán tối ưu

#### Tích phân

**Tích phân bất định**: ∫f(x)dx = F(x) + C **Tích phân xác định**: ∫[a→b] f(x)dx = F(b) - F(a) **Ứng dụng**: tính diện tích, thể tích

## VẬT LÝ

### Cơ học

#### Động học chất điểm

**Chuyển động thẳng**:

* Chuyển động đều: v = const, s = vt
* Chuyển động biến đổi đều: v = v₀ + at, s = v₀t + (1/2)at²
* Phương trình độc lập thời gian: v² = v₀² + 2as

**Chuyển động tròn**:

* Tọa độ góc: φ = φ₀ + ωt + (1/2)αt²
* Vận tốc góc: ω = dφ/dt
* Gia tốc góc: α = dω/dt
* Liên hệ: v = ωr, a\_t = αr, a\_n = ω²r = v²/r

**Chuyển động ném**:

* Ném ngang: x = v₀t, y = (1/2)gt²
* Ném xiên: x = v₀cos(α)t, y = v₀sin(α)t - (1/2)gt²

#### Động lực học

**Các định luật Newton**:

1. Định luật quán tính: Vật giữ nguyên trạng thái khi không có ngoại lực
2. F = ma: Gia tốc tỉ lệ thuận với lực, tỉ lệ nghịch với khối lượng
3. Tương tác: F\_AB = -F\_BA

**Các loại lực**:

* Trọng lực: P = mg
* Lực đàn hồi: F = -kx (định luật Hooke)
* Lực ma sát: f = μN
* Lực hấp dẫn: F = G(m₁m₂)/r²

#### Năng lượng và động lượng

**Động năng**: W\_đ = (1/2)mv² **Thế năng**:

* Trọng trường: W\_t = mgh
* Đàn hồi: W\_t = (1/2)kx²

**Định luật bảo toàn**:

* Năng lượng: W\_đ + W\_t = const
* Động lượng: p = mv, Σp\_trước = Σp\_sau

### Nhiệt học

#### Cơ sở phân tử động học

* Nội năng: tổng động năng và thế năng phân tử
* Nhiệt độ: đại lượng đặc trưng cho mức độ chuyển động nhiệt
* Phương trình trạng thái khí lý tưởng: PV = nRT

#### Nguyên lý nhiệt động lực học

**Nguyên lý I**: ΔU = Q - A

* Q > 0: hệ nhận nhiệt
* A > 0: hệ sinh công
* ΔU: độ biến thiên nội năng

**Nguyên lý II**: Nhiệt không thể tự truyền từ vật lạnh sang vật nóng

### Điện học

#### Tĩnh điện học

**Định luật Coulomb**: F = k(q₁q₂)/r² **Cường độ điện trường**: E = F/q = kQ/r² **Điện thế**: V = W/q, V = kQ/r **Tụ điện**: C = Q/U, năng lượng W = (1/2)CU²

#### Dòng điện không đổi

**Các đại lượng cơ bản**:

* Cường độ: I = Q/t
* Điện trở: R = ρl/S
* Định luật Ohm: U = IR

**Mạch điện**:

* Nối tiếp: I\_chung, R\_tổng = ΣR\_i, U\_tổng = ΣU\_i
* Song song: U\_chung, 1/R\_tổng = Σ(1/R\_i), I\_tổng = ΣI\_i

**Công và công suất**:

* Công: A = UIt = I²Rt = U²t/R
* Công suất: P = UI = I²R = U²/R

#### Từ học

**Từ trường**: vùng không gian có tác dụng lực từ **Lực từ**: F = BIl·sin(α) **Cảm ứng điện từ**:

* Định luật Faraday: ε = -dΦ/dt
* Định luật Lenz: chiều dòng điện cảm ứng chống lại sự biến đổi

### Dao động và sóng

#### Dao động cơ

**Dao động điều hòa**: x = A·cos(ωt + φ)

* Biên độ A, tần số góc ω, pha ban đầu φ
* Vận tốc: v = -Aω·sin(ωt + φ)
* Gia tốc: a = -Aω²·cos(ωt + φ) = -ω²x

**Con lắc**:

* Lò xo: T = 2π√(m/k)
* Đơn: T = 2π√(l/g) (với góc nhỏ)

#### Sóng cơ

**Phương trình sóng**: u = A·cos(2π(t/T - x/λ)) **Đặc trưng**: biên độ A, chu kỳ T, tần số f, bước sóng λ **Liên hệ**: v = λf = λ/T

### Quang học

#### Quang hình học

**Định luật phản xạ**: góc tới = góc phản xạ **Định luật khúc xạ**: n₁sin(i) = n₂sin(r) **Thấu kính mỏng**: 1/f = 1/d + 1/d'

#### Quang vật lý

**Tính chất sóng**: giao thoa, nhiễu xạ **Tính chất hạt**: hiệu ứng quang điện, E = hf

## HÓA HỌC

### Cấu tạo nguyên tử và phân tử

#### Cấu tạo nguyên tử

**Thành phần**: proton (+), neutron (0), electron (-) **Đặc trưng**:

* Số hiệu nguyên tử Z = số proton
* Số khối A = số proton + số neutron
* Nguyên tử trung hòa: số proton = số electron

**Đồng vị**: cùng Z, khác số neutron

* Khối lượng nguyên tử trung bình
* Phóng xạ của một số đồng vị

#### Cấu hình electron

**Nguyên lý sắp xếp**:

* Nguyên lý Pauli: mỗi orbital tối đa 2 electron
* Quy tắc Hund: electron phân bổ đơn lẻ trước khi ghép đôi
* Nguyên lý năng lượng tối thiểu

**Phân lớp electron**: s, p, d, f

* s: tối đa 2e, p: tối đa 6e, d: tối đa 10e, f: tối đa 14e

#### Bảng tuần hoàn

**Cấu trúc**:

* 7 chu kỳ: tương ứng số lớp electron
* 18 nhóm: A (8 nhóm chính), B (10 nhóm phụ)

**Quy luật biến đổi tính chất**:

* Bán kính nguyên tử: giảm trong chu kỳ, tăng trong nhóm
* Năng lượng ion hóa: tăng trong chu kỳ, giảm trong nhóm
* Độ âm điện: tăng trong chu kỳ, giảm trong nhóm

### Liên kết hóa học

#### Liên kết cộng hóa trị

**Đặc điểm**: chia sẻ cặp electron **Phân loại**:

* Cộng hóa trị không cực: chia sẻ đều electron
* Cộng hóa trị có cực: chia sẻ không đều electron
* Liên kết σ và π

#### Liên kết ion

**Đặc điểm**: cho nhận electron hoàn toàn **Điều kiện**: hiệu độ âm điện lớn (> 1.7) **Tính chất**: nhiệt độ nóng chảy/sôi cao, dẫn điện khi nóng chảy

#### Liên kết kim loại

**Đặc điểm**: "biển electron" di động **Tính chất**: dẫn điện, dẫn nhiệt, dẻo, ánh kim

### Phản ứng hóa học

#### Phân loại phản ứng

**Theo cơ chế**:

* Tổng hợp: A + B → AB
* Phân hủy: AB → A + B
* Thế: A + BC → AC + B
* Trao đổi: AB + CD → AD + CB

**Theo trao đổi electron**:

* Phản ứng oxi hóa-khử: có sự thay đổi số oxi hóa
* Phản ứng axit-bazơ: trao đổi proton

#### Tốc độ phản ứng

**Yếu tố ảnh hưởng**:

* Nồng độ: tăng nồng độ → tăng tốc độ
* Nhiệt độ: tăng nhiệt độ → tăng tốc độ
* Diện tích tiếp xúc: tăng diện tích → tăng tốc độ
* Xúc tác: làm tăng tốc độ phản ứng

#### Cân bằng hóa học

**Đặc điểm**: v\_thuận = v\_nghịch **Hằng số cân bằng**: K\_c = [sản phẩm]/[chất phản ứng] **Nguyên lý Le Chatelier**: khi có tác động bên ngoài, cân bằng chuyển dịch theo chiều chống lại tác động đó

### Hóa học hữu cơ

#### Hidrôcacbon

**Ankan**: C\_nH\_(2n+2)

* Mạch thẳng: phản ứng thế, tách hydrogen
* Đồng phân cấu tạo

**Anken**: C\_nH\_(2n)

* Liên kết đôi C=C
* Phản ứng cộng: với H₂, Br₂, HX, H₂O
* Phản ứng trùng hợp

**Ankadien và ankin**:

* Ankadien: hai liên kết đôi
* Ankin: liên kết ba C≡C

**Aren**: vòng benzen C₆H₆

* Công thức cộng hưởng
* Phản ứng thế thơm

#### Dẫn xuất halogen

**Ankin halogenua**: R-X

* Phản ứng thế nucleophile
* Phản ứng tách để tạo anken

#### Hợp chất chứa oxy

**Ancol**: R-OH

* Ancol bậc 1, 2, 3
* Phản ứng với Na, axit carboxylic
* Phản ứng oxi hóa

**Phenol**: Ar-OH

* Tính axit yếu
* Phản ứng thế ở vòng benzen

**Anbehit và xeton**:

* Anbehit: R-CHO
* Xeton: R-CO-R'
* Phản ứng cộng, oxi hóa

**Axit carboxylic**: R-COOH

* Tính axit
* Phản ứng với bazơ, ancol (este hóa)

## LẬP TRÌNH

### Lập trình hướng đối tượng (OOP)

#### Các khái niệm cơ bản

**Lớp (Class)**: khuôn mẫu để tạo đối tượng **Đối tượng (Object)**: thực thể cụ thể của lớp **Thuộc tính (Attribute)**: đặc điểm của đối tượng **Phương thức (Method)**: hành động của đối tượng

class Vehicle:

def \_\_init\_\_(self, brand, model, year):

self.brand = brand # Thuộc tính

self.model = model

self.year = year

self.is\_running = False

def start\_engine(self): # Phương thức

self.is\_running = True

return f"{self.brand} {self.model} engine started"

def stop\_engine(self):

self.is\_running = False

return f"{self.brand} {self.model} engine stopped"

#### Tính kế thừa (Inheritance)

class Car(Vehicle): # Car kế thừa từ Vehicle

def \_\_init\_\_(self, brand, model, year, num\_doors):

super().\_\_init\_\_(brand, model, year)

self.num\_doors = num\_doors

def open\_trunk(self):

return f"Trunk of {self.brand} {self.model} opened"

class Motorcycle(Vehicle):

def \_\_init\_\_(self, brand, model, year, has\_sidecar=False):

super().\_\_init\_\_(brand, model, year)

self.has\_sidecar = has\_sidecar

def wheelie(self):

return f"{self.brand} {self.model} doing a wheelie!"

#### Tính đóng gói (Encapsulation)

class BankAccount:

def \_\_init\_\_(self, account\_number, initial\_balance=0):

self.account\_number = account\_number

self.\_\_balance = initial\_balance # Private attribute

def deposit(self, amount):

if amount > 0:

self.\_\_balance += amount

return f"Deposited ${amount}. New balance: ${self.\_\_balance}"

return "Invalid deposit amount"

def withdraw(self, amount):

if 0 < amount <= self.\_\_balance:

self.\_\_balance -= amount

return f"Withdrew ${amount}. New balance: ${self.\_\_balance}"

return "Insufficient funds or invalid amount"

def get\_balance(self): # Getter method

return self.\_\_balance

#### Tính đa hình (Polymorphism)

class Animal:

def make\_sound(self):

pass

class Dog(Animal):

def make\_sound(self):

return "Woof!"

class Cat(Animal):

def make\_sound(self):

return "Meow!"

class Cow(Animal):

def make\_sound(self):

return "Moo!"

# Polymorphism in action

animals = [Dog(), Cat(), Cow()]

for animal in animals:

print(animal.make\_sound()) # Calls appropriate method

### Cấu trúc dữ liệu nâng cao

#### Stack (Ngăn xếp)

class Stack:

def \_\_init\_\_(self):

self.items = []

def is\_empty(self):

return len(self.items) == 0

def push(self, item):

self.items.append(item)

def pop(self):

if not self.is\_empty():

return self.items.pop()

return None

def peek(self):

if not self.is\_empty():

return self.items[-1]

return None

def size(self):

return len(self.items)

#### Queue (Hàng đợi)

from collections import deque

class Queue:

def \_\_init\_\_(self):

self.items = deque()

def is\_empty(self):

return len(self.items) == 0

def enqueue(self, item):

self.items.append(item)

def dequeue(self):

if not self.is\_empty():

return self.items.popleft()

return None

def front(self):

if not self.is\_empty():

return self.items[0]

return None

def size(self):

return len(self.items)

#### Linked List (Danh sách liên kết)

class Node:

def \_\_init\_\_(self, data):

self.data = data

self.next = None

class LinkedList:

def \_\_init\_\_(self):

self.head = None

def append(self, data):

new\_node = Node(data)

if not self.head:

self.head = new\_node

return

current = self.head

while current.next:

current = current.next

current.next = new\_node

def prepend(self, data):

new\_node = Node(data)

new\_node.next = self.head

self.head = new\_node

def delete(self, data):

if not self.head:

return

if self.head.data == data:

self.head = self.head.next

return

current = self.head

while current.next and current.next.data != data:

current = current.next

if current.next:

current.next = current.next.next

def display(self):

elements = []

current = self.head

while current:

elements.append(current.data)

current = current.next

return elements

### Thuật toán nâng cao

#### Thuật toán sắp xếp

**Merge Sort (O(n log n))**:

def merge\_sort(arr):

if len(arr) <= 1:

return arr

mid = len(arr) // 2

left = merge\_sort(arr[:mid])

right = merge\_sort(arr[mid:])

return merge(left, right)

def merge(left, right):

result = []

i, j = 0, 0

while i < len(left) and j < len(right):

if left[i] <= right[j]:

result.append(left[i])

i += 1

else:

result.append(right[j])

j += 1

result.extend(left[i:])

result.extend(right[j:])

return result

**Quick Sort (O(n log n) average)**:

def quick\_sort(arr):

if len(arr) <= 1:

return arr

pivot = arr[len(arr) // 2]

left = [x for x in arr if x < pivot]

middle = [x for x in arr if x == pivot]

right = [x for x in arr if x > pivot]

return quick\_sort(left) + middle + quick\_sort(right)

#### Thuật toán tìm kiếm

**Binary Search (O(log n))**:

def binary\_search(arr, target):

left, right = 0, len(arr) - 1

while left <= right:

mid = (left + right) // 2

if arr[mid] == target:

return mid

elif arr[mid] < target:

left = mid + 1

else:

right = mid - 1

return -1 # Not found

#### Thuật toán đồ thị

**Depth-First Search (DFS)**:

def dfs(graph, start, visited=None):

if visited is None:

visited = set()

visited.add(start)

print(start, end=' ')

for neighbor in graph.get(start, []):

if neighbor not in visited:

dfs(graph, neighbor, visited)

return visited

**Breadth-First Search (BFS)**:

from collections import deque

def bfs(graph, start):

visited = set()

queue = deque([start])

visited.add(start)

while queue:

vertex = queue.popleft()

print(vertex, end=' ')

for neighbor in graph.get(vertex, []):

if neighbor not in visited:

visited.add(neighbor)

queue.append(neighbor)

return visited

### Web Development

#### Backend với Flask

from flask import Flask, request, jsonify, render\_template

app = Flask(\_\_name\_\_)

# Dữ liệu mẫu

users = [

{"id": 1, "name": "Alice", "email": "alice@example.com"},

{"id": 2, "name": "Bob", "email": "bob@example.com"}

]

@app.route('/')

def home():

return render\_template('index.html')

@app.route('/api/users', methods=['GET'])

def get\_users():

return jsonify(users)

@app.route('/api/users', methods=['POST'])

def create\_user():

data = request.json

new\_user = {

"id": len(users) + 1,

"name": data.get('name'),

"email": data.get('email')

}

users.append(new\_user)

return jsonify(new\_user), 201

@app.route('/api/users/<int:user\_id>', methods=['GET'])

def get\_user(user\_id):

user = next((u for u in users if u['id'] == user\_id), None)

if user:

return jsonify(user)

return jsonify({"error": "User not found"}), 404

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

#### Frontend với JavaScript ES6+

// Modern JavaScript features

class UserManager {

constructor() {

this.users = [];

this.apiUrl = '/api/users';

}

// Async/await for API calls

async fetchUsers() {

try {

const response = await fetch(this.apiUrl);

if (!response.ok) {

throw new Error(`HTTP error! status: ${response.status}`);

}

this.users = await response.json();

return this.users;

} catch (error) {

console.error('Error fetching users:', error);

throw error;

}

}

async createUser(userData) {

try {

const response = await fetch(this.apiUrl, {

method: 'POST',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify(userData)

});

if (!response.ok) {

throw new Error(`HTTP error! status: ${response.status}`);

}

const newUser = await response.json();

this.users.push(newUser);

return newUser;

} catch (error) {

console.error('Error creating user:', error);

throw error;

}

}

// Arrow functions and array methods

getUsersByDomain = (domain) => {

return this.users.filter(user =>

user.email.endsWith(`@${domain}`)

);

}

getUserNames = () => {

return this.users.map(user => user.name);

}

// Destructuring and template literals

displayUser({id, name, email}) {

return `

<div class="user-card" data-id="${id}">

<h3>${name}</h3>

<p>Email: ${email}</p>

<button onclick="editUser(${id})">Edit</button>

<button onclick="deleteUser(${id})">Delete</button>

</div>

`;

}

}

// DOM manipulation and event handling

document.addEventListener('DOMContentLoaded', async () => {

const userManager = new UserManager();

const userContainer = document.getElementById('users-container');

const userForm = document.getElementById('user-form');

// Load initial users

try {

await userManager.fetchUsers();

renderUsers();

} catch (error) {

showError('Failed to load users');

}

// Form submission

userForm.addEventListener('submit', async (e) => {

e.preventDefault();

const formData = new FormData(e.target);

const userData = {

name: formData.get('name'),

email: formData.get('email')

};

try {

await userManager.createUser(userData);

renderUsers();

e.target.reset();

showSuccess('User created successfully');

} catch (error) {

showError('Failed to create user');

}

});

function renderUsers() {

const userHTML = userManager.users

.map(user => userManager.displayUser(user))

.join('');

userContainer.innerHTML = userHTML;

}

function showError(message) {

showNotification(message, 'error');

}

function showSuccess(message) {

showNotification(message, 'success');

}

function showNotification(message, type) {

const notification = document.createElement('div');

notification.className = `notification ${type}`;

notification.textContent = message;

document.body.appendChild(notification);

setTimeout(() => {

notification.remove();

}, 3000);

}

});

### Database và SQL

-- Tạo bảng

CREATE TABLE users (

id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100) NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE TABLE posts (

id INT PRIMARY KEY AUTO\_INCREMENT,

title VARCHAR(200) NOT NULL,

content TEXT,

user\_id INT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users(id)

);

-- Truy vấn cơ bản

SELECT \* FROM users WHERE email LIKE '%@gmail.com';

SELECT u.name, COUNT(p.id) as post\_count

FROM users u

LEFT JOIN posts p ON u.id = p.user\_id

GROUP BY u.id, u.name

HAVING post\_count > 0

ORDER BY post\_count DESC;

-- Indexes for performance

CREATE INDEX idx\_user\_email ON users(email);

CREATE INDEX idx\_post\_user\_id ON posts(user\_id);