HANDOUT BUỔI 1: GIỚI THIỆU MÔN HỌC VÀ CÁC THUẬT TOÁN CƠ BẢN

✍ Mục tiêu của buổi học

- **Hiểu về Toán rời rạc**: Nắm rõ khái niệm, sự khác biệt với Toán liên tục và ứng dụng trong Công nghệ Thông tin.
- **Nắm vững thuật toán**: Hiểu định nghĩa, đặc trưng và các phương pháp biểu diễn thuật toán (ngôn ngữ tự nhiên, sơ đồ khối, mã giả).
- **Thực hành và phân tích thuật toán**: Viết mã giả, sơ đồ khối, cài đặt và đánh giá thuật toán tìm kiếm.
- **Đánh giá thuật toán**: So sánh độ phức tạp thời gian và tối ưu thuật toán cho các bài toán thực tế.

PHẦN 1: GIỚI THIỆU MÔN HỌC

1.1. Giới thiệu chung

1.1.1. Toán rời rạc là gì?

Toán rời rạc (Discrete Mathematics) là một lĩnh vực của toán học chuyên nghiên cứu về các đối tượng rời rạc, tức là các cấu trúc không liên tục, không có giá trị trung gian giữa các phần tử. Khác với toán liên tục (liên quan đến giải tích, vi phân và tích phân), toán rời rạc tập trung vào các tập hợp hữu hạn hoặc đếm được như đồ thị, cây, tổ hợp, số nguyên, quan hệ và thuật toán.

Tính chất đặc trưng của Toán rời rạc:

- **Không liên tục**: Làm việc với tập hợp số nguyên, đồ thị, tập hợp rời rạc thay vì các đại lượng thay đổi liên tục như trong toán giải tích.
- **Tính logic cao**: Yêu cầu tư duy chặt chẽ để phân tích và chứng minh các mệnh đề toán học.
- **Ứng dụng rộng rãi trong CNTT**: Đóng vai trò nền tảng trong lập trình, thuật toán, trí tuệ nhân tạo và an toàn thông tin.

1.1.2. Các chủ đề chính của môn học

Môn học bao gồm 6 chủ đề chính, cung cấp nền tảng cho nhiều lĩnh vực trong Công nghệ Thông tin:

Chủ đề	Nội dung chính	Ứng dụng thực tế
Thuật toán	Định nghĩa, biểu diễn, độ phức tạp thuật toán	Lập trình, tối ưu hóa thuật toán
Bài toán đếm	Quy tắc đếm, tổ hợp, hoán vị, nguyên lý Dirichlet	Xác suất, mật mã, phân tích dữ liệu
Lý thuyết đồ thị	Cấu trúc đồ thị, đồ thị Euler, Hamilton, đường đi ngắn nhất	Google Maps, phân tích mạng xã hội
Cây	Cây nhị phân, cây tìm kiếm, cây khung nhỏ nhất	Tìm kiếm AI, hệ thống phân cấp dữ liệu

1.1.3. Ứng dụng thực tế của Toán rời rạc

1. Google Search: Tìm kiếm thông tin tối ưu

Google sử dụng thuật toán **PageRank** để xác định độ quan trọng của một trang web dựa trên lý thuyết đồ thị. Các trang web được mô hình hóa như một đồ thị có các nút (nodes) và các liên kết giữa trang web là các cạnh (edges).

Ví dụ:

- Khi một trang web có nhiều liên kết từ các trang uy tín, giá trị của nó được tăng lên theo thuật toán PageRank.
- Các thuật toán tối ưu hóa đồ thị giúp giảm thời gian tìm kiếm thông tin và cải thiện tốc độ truy vấn.

2. Trí tuệ nhân tạo (AI) & Học máy (Machine Learning)

Trong AI, toán rời rạc đóng vai trò quan trọng trong việc xây dựng các mô hình học máy. Một số ứng dụng bao gồm:

- **Cây quyết định (Decision Tree)**: Dùng trong phân loại dữ liệu, nhận diện khuôn mặt, phát hiện gian lận tài chính.
- **Mạng nơ-ron nhân tạo (Artificial Neural Networks - ANN)**: Dựa trên lý thuyết đồ thị để huấn luyện mô hình nhận dạng mẫu.
- **Xác suất và tổ hợp** giúp tối ưu hóa các thuật toán dự đoán trong AI.

3. Bảo mật thông tin và mã hóa RSA

Hệ thống mã hóa RSA, được sử dụng trong giao dịch ngân hàng và bảo mật dữ liệu, dựa trên **lý thuyết số và số nguyên tố**.

- RSA hoạt động dựa trên phép toán **lũy thừa modulo** và tính chất khó phân tích của số nguyên tố lớn.
- Các thuật toán băm (hashing) như SHA-256 được ứng dụng rộng rãi trong bảo mật mật khẩu và Blockchain.

4. Hệ thống giao thông và thuật toán tìm đường đi ngắn nhất

Thuật toán **Dijkstra** được sử dụng để tìm tuyến đường ngắn nhất trong hệ thống giao thông, ứng dụng trong Google Maps, Uber, Grab và các hệ thống logistics.

Ví dụ:

- Khi đặt xe Grab, hệ thống sử dụng thuật toán tìm đường ngắn nhất để xác định tuyến đường tối ưu.
- Các công ty vận tải sử dụng thuật toán này để tối ưu hóa chi phí vận chuyển.

1.1.4. Hoạt động thảo luận lớp học

Sau khi trình bày các nội dung trên, sinh viên tham gia thảo luận nhóm với các câu hỏi:

- Bạn mong đợi điều gì từ môn học Toán rời rạc?
- Theo bạn, chủ đề nào quan trọng nhất trong Toán rời rạc?
- Bạn đã từng sử dụng một thuật toán toán rời rạc nào trong thực tế chưa?

Kết thúc phần thảo luận, sinh viên liệt kê ít nhất 3 ứng dụng thực tế của toán rời rạc mà họ quan sát được trong cuộc sống hoặc ngành CNTT.

1.2. Phương pháp học & đánh giá

1.2.1. Hình thức đánh giá học phần

Môn học Toán rời rạc được đánh giá dựa trên các tiêu chí sau:

1. **Chuyên cần (10%)**

- Sinh viên cần tham gia đầy đủ các buổi học và làm bài tập được giao.
- Điểm chuyên cần được tính dựa trên mức độ tham gia vào bài giảng và các hoạt động trên lớp.
- Vắng mặt quá 25% số buổi học sẽ không đủ điều kiện tham gia kiểm tra cuối kỳ.

2. **Bài tập nhóm & lập trình (25%)**
   - Sinh viên làm việc theo nhóm để giải quyết các bài toán thực tế bằng thuật toán.
   - Yêu cầu lập trình thuật toán, kiểm thử, tối ưu hóa và trình bày giải pháp.
   - Điểm số dựa trên độ chính xác, hiệu suất thuật toán và sự đóng góp của từng thành viên.
3. **Kiểm tra giữa kỳ (15%)**
   - Bài kiểm tra gồm phần lý thuyết và bài tập thực hành về thuật toán, bài toán đếm và lý thuyết đồ thị.
   - Hình thức: Tự luận, trắc nghiệm hoặc lập trình thuật toán đơn giản.
4. **Kiểm tra cuối kỳ (50%)**
   - Bao gồm phần lý thuyết và lập trình thuật toán giải quyết bài toán thực tế.
   - Tiêu chí đánh giá:
     - Tính đúng đắn và hiệu quả của thuật toán.
     - Khả năng tối ưu hóa về thời gian và bộ nhớ.
     - Cách trình bày và giải thích thuật toán.

1.2.2. Phương pháp giảng dạy

Môn học kết hợp giữa lý thuyết và thực hành, bao gồm:

- **Giảng dạy lý thuyết**: Trình bày khái niệm nền tảng, công thức và chứng minh toán học.
- **Bài tập thực hành & lập trình**: Giúp sinh viên áp dụng lý thuyết vào các bài toán thực tế.
- **Thảo luận nhóm & làm việc nhóm**: Sinh viên được khuyến khích trao đổi ý tưởng, phát triển tư duy thuật toán.

PHẦN 2: CHƯƠNG 1 - THUẬT TOÁN (PHẦN 1)

2.1. Khái niệm thuật toán

2.1.1. Mở đầu

Khái niệm thuật toán

Thuật toán là một tập hợp hữu hạn các bước xác định nhằm giải quyết một bài toán cụ thể. Trong khoa học máy tính, thuật toán đóng vai trò nền tảng cho mọi chương trình và hệ thống, đảm bảo tính chính xác, hiệu suất và tối ưu hóa tài nguyên tính toán. Một thuật toán hiệu quả cần đáp ứng các tiêu chí như tính đúng đắn, hiệu quả thời gian và không gian, cũng như khả năng mở rộng.

Ví dụ minh họa

Một bài toán cơ bản nhưng quan trọng trong lập trình là **hoán đổi giá trị hai số nguyên**, có thể thực hiện theo nhiều phương pháp khác nhau:

- **Sử dụng biến tạm:**

```
int a = 5, b = 10;
int temp = a;
a = b;
b = temp;
```

  - Phương pháp này đảm bảo tính đơn giản và dễ hiểu.
- **Không sử dụng biến tạm (dùng phép toán số học):**

```
a = a + b;
b = a - b;
a = a - b;
```

  - Giảm yêu cầu về bộ nhớ nhưng có thể dẫn đến lỗi tràn số.
- **Sử dụng phép toán XOR:**

```
a = a ^ b;
b = a ^ b;
a = a ^ b;
```

  - Phương pháp này thường được sử dụng trong các hệ thống nhúng để tối ưu tài nguyên.

Bài toán hoán đổi hai số minh họa một vấn đề có thể được giải quyết bằng nhiều cách tiếp cận khác nhau, mỗi cách có ưu và nhược điểm riêng. Việc lựa chọn thuật toán phù hợp phụ thuộc vào các yêu cầu cụ thể của hệ thống.

Ứng dụng thuật toán trong công nghệ thông tin

Thuật toán là thành phần cốt lõi trong nhiều lĩnh vực của công nghệ thông tin, bao gồm:

Lĩnh vực	**Ứng dụng thuật toán**
Công cụ tìm kiếm	Thuật toán PageRank trong Google giúp tối ưu kết quả tìm kiếm.
Thương mại điện tử	Hệ thống gợi ý sản phẩm sử dụng thuật toán Recommendation System.

Lĩnh vực	Ứng dụng thuật toán
Bảo mật & mã hóa	Thuật toán RSA được sử dụng để mã hóa dữ liệu trong giao dịch trực tuyến.
Học máy & AI	Thuật toán Gradient Descent tối ưu mô hình học sâu.
Hệ thống giao thông	Các thuật toán tìm đường như Dijkstra và A* tối ưu lộ trình.

Thuật toán không chỉ giúp cải thiện hiệu suất hệ thống mà còn là cơ sở cho các ứng dụng thông minh, tự động hóa quy trình và tối ưu hóa tài nguyên.

Phần này đã giới thiệu khái niệm thuật toán, minh họa bằng một ví dụ đơn giản và trình bày các ứng dụng thực tế. Việc hiểu và lựa chọn thuật toán phù hợp là yếu tố quan trọng trong phát triển phần mềm và tối ưu hóa hệ thống.

Câu hỏi thảo luận:

1. Hãy đưa ra một ví dụ thực tế trong đó thuật toán đóng vai trò quan trọng.
2. Trong ba phương pháp hoán đổi hai số trên, phương pháp nào tối ưu hơn trong điều kiện bộ nhớ hạn chế?

Bài tập thực hành:

1. Viết mã giả thuật toán tìm số lớn nhất trong một danh sách số nguyên.
2. Cài đặt thuật toán tính tổng từ 1 đến n bằng hai phương pháp: sử dụng vòng lặp và công thức Gauss.

2.1.2. Định nghĩa thuật toán

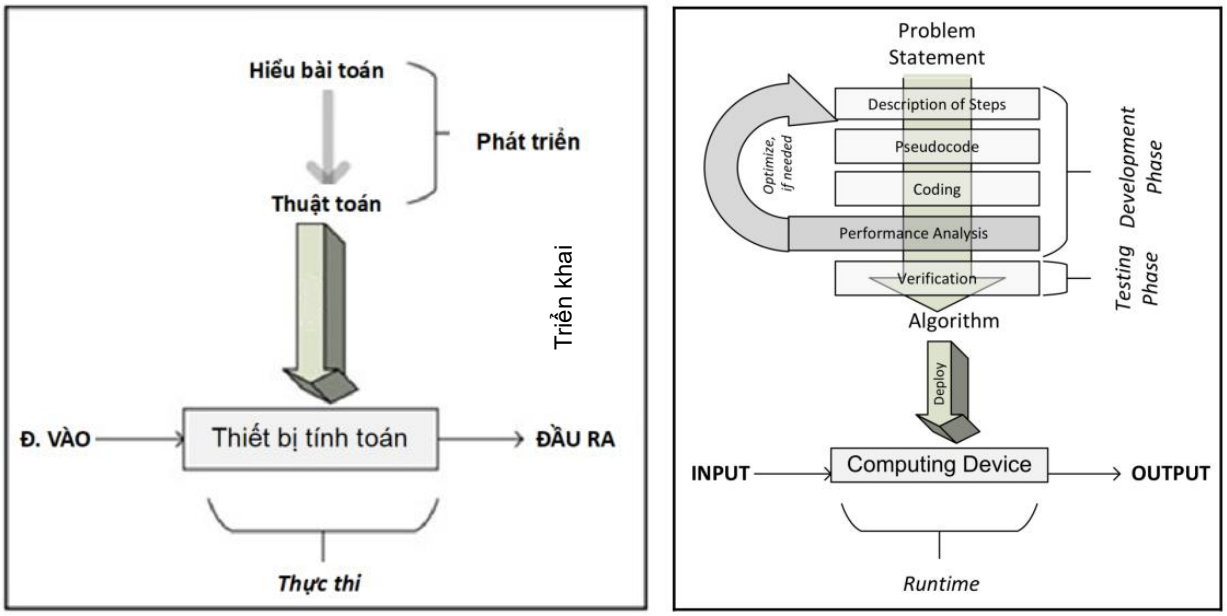
Khái niệm thuật toán

Thuật toán là một tập hợp hữu hạn các bước được xác định rõ ràng để giải quyết một bài toán cụ thể. Mỗi bước trong thuật toán phải có tính xác định, nghĩa là với cùng một đầu vào, thuật toán phải luôn cho cùng một kết quả.

Trong khoa học máy tính, thuật toán không chỉ được sử dụng để xử lý dữ liệu mà còn là nền tảng cho mọi hệ thống thông minh, từ trí tuệ nhân tạo đến các hệ thống điều khiển nhúng. Một thuật toán tốt cần đảm bảo các tiêu chí:

- **Tính đúng đắn**: Đưa ra kết quả chính xác cho mọi đầu vào hợp lệ.
- **Tính hiệu quả**: Sử dụng tài nguyên tối ưu, bao gồm thời gian và bộ nhớ.
- **Tính tổng quát**: Áp dụng được cho nhiều bài toán có đặc điểm tương tự.
- **Tính xác định**: Mỗi bước đều rõ ràng và không gây ra sự mơ hồ.

Các pha của một thuật toán



Hình 1. Các giai đoạn phát triển thuật toán

Quá trình phát triển, triển khai và sử dụng một thuật toán bao gồm các giai đoạn sau:

1. **Hiểu bài toán**
    - Bước đầu tiên là phân tích yêu cầu từ bài toán để xác định những gì cần thực hiện.
2. **Pha phát triển (Development Phase)**
    - **Pha thiết kế (Design Phase)**:
        - Ở giai đoạn này, chúng ta xây dựng kiến trúc, logic và các chi tiết triển khai của thuật toán.
        - Cần cân nhắc cả độ chính xác và hiệu suất của thuật toán.
        - Trong nhiều trường hợp, có thể có nhiều thuật toán thay thế cho cùng một bài toán. Khi đó, cần so sánh các thuật toán để chọn giải pháp phù hợp nhất.
        - Một số thuật toán có thể đơn giản và nhanh nhưng kém chính xác, trong khi một số khác rất chính xác nhưng lại mất nhiều thời gian tính toán.
        - Thiết kế thuật toán là một quá trình lặp, yêu cầu đánh giá và điều chỉnh liên tục.
    - **Pha mã hóa (Coding Phase)**:

        - Sau khi hoàn thành thiết kế, thuật toán được chuyển thành chương trình máy tính.
        - Việc lập trình cần đảm bảo đúng theo thiết kế đã đề xuất.
        - Cả pha thiết kế và mã hóa đều có tính lặp lại để tối ưu hóa thuật toán.
3. **Pha triển khai và thực thi (Deployment & Runtime Phase)**
    - Thuật toán sau khi lập trình sẽ được triển khai trên thiết bị tính toán (Computing Device).
    - Khi thực thi, thuật toán nhận đầu vào (Input), xử lý và tạo ra đầu ra (Output).
    - Hiệu suất của thuật toán được đánh giá trong quá trình chạy thực tế.

So sánh thuật toán xác định và thuật toán gần đúng

Trong thực tế, không phải bài toán nào cũng có thể giải quyết bằng một thuật toán chính xác. Do đó, có thể phân loại thuật toán thành hai nhóm chính:

1. **Thuật toán xác định (Exact Algorithm)**
    - Luôn tìm được lời giải chính xác.
    - Thường có độ phức tạp cao đối với bài toán lớn.
    - Ví dụ:
        - Thuật toán Dijkstra tìm đường đi ngắn nhất.
        - Thuật toán Kruskal xây dựng cây khung nhỏ nhất.
2. **Thuật toán gần đúng (Approximate Algorithm)**
    - Đưa ra lời giải xấp xỉ với độ chính xác có thể chấp nhận được.
    - Hiệu quả hơn trong các bài toán có không gian tìm kiếm lớn.
    - **Ví dụ:**
        - Thuật toán **A*** trong tìm đường đi ngắn nhất sử dụng heuristic để tăng tốc tìm kiếm.
        - Thuật toán **Simulated Annealing** trong tối ưu hóa tổ hợp.
        - Thuật toán **Heuristic** trong các bài toán tối ưu hóa như lập lịch hoặc quy hoạch tuyến đường.
        - Thuật toán **Di truyền (Genetic Algorithm)** trong tối ưu hóa và học máy.

Phần này đã giới thiệu định nghĩa thuật toán, các pha phát triển thuật toán từ thiết kế đến triển khai, cũng như phân loại thuật toán theo mức độ chính xác. Việc hiểu rõ các khía cạnh này giúp lập trình viên lựa chọn phương pháp phù hợp cho từng bài toán cụ thể.

Câu hỏi thảo luận:

1. Trong các chiến lược thiết kế thuật toán, phương pháp nào phù hợp nhất cho bài toán tối ưu hóa tuyến đường trong hệ thống giao thông? Giải thích lý do.
2. Hãy đưa ra một ví dụ thực tế trong đó thuật toán gần đúng được sử dụng thay vì thuật toán xác định.

Bài tập thực hành:

1. Viết mã giả thuật toán tìm số lớn nhất trong một danh sách số nguyên theo phương pháp Brute Force.
2. Phân tích độ phức tạp thuật toán tìm kiếm nhị phân và so sánh với tìm kiếm tuần tự.

2.1.3. Các đặc trưng của thuật toán

Một thuật toán được coi là hợp lệ khi nó đáp ứng các tiêu chí quan trọng đảm bảo tính khả thi và hiệu quả trong quá trình thực thi. Dưới đây là các đặc trưng chính của một thuật toán:

1. Đầu vào (Input)

Thuật toán cần có **tối thiểu một đầu vào** để xác định dữ liệu cần xử lý. Đầu vào có thể là số nguyên, số thực, chuỗi ký tự hoặc các cấu trúc dữ liệu phức tạp như danh sách, cây, đồ thị.

Ví dụ:

- Trong thuật toán tính tổng từ 1 đến n, đầu vào là số nguyên **n**.
- Trong thuật toán tìm kiếm nhị phân, đầu vào bao gồm **một danh sách đã sắp xếp** và **giá trị cần tìm**.

2. Đầu ra (Output)

Thuật toán phải cung cấp **ít nhất một đầu ra** tương ứng với yêu cầu của bài toán. Đầu ra phải có ý nghĩa và có thể kiểm chứng được.

Ví dụ:

- Thuật toán tìm số lớn nhất trong danh sách sẽ trả về một số nguyên.

- Thuật toán tìm đường đi ngắn nhất trên đồ thị sẽ trả về một tập hợp các đỉnh theo thứ tự tối ưu.

3. Tính dừng (Finiteness)

Một thuật toán hợp lệ phải luôn kết thúc sau một số hữu hạn bước thực hiện. Nếu một thuật toán không có điểm dừng hoặc lặp vô hạn, nó không được coi là một thuật toán hợp lệ.

Ví dụ:

- Một vòng lặp **không có điều kiện dừng** sẽ khiến chương trình chạy mãi mãi.
- Thuật toán đệ quy cần có điều kiện cơ sở để đảm bảo quá trình gọi hàm kết thúc.

4. Tính xác định (Definiteness)

Mỗi bước trong thuật toán phải được xác định rõ ràng, không gây ra sự nhập nhằng. Với cùng một đầu vào, thuật toán phải luôn tạo ra **kết quả giống nhau**.

Ví dụ:

- Thuật toán tính tổng từ 1 đến n có các bước xác định rõ ràng, không phụ thuộc vào yếu tố ngẫu nhiên.
- Thuật toán chọn số ngẫu nhiên từ danh sách **không mang tính xác định** do đầu ra có thể thay đổi giữa các lần chạy.

5. Tính hiệu quả (Effectiveness)

Thuật toán cần đảm bảo khả năng thực thi với số bước hợp lý, tránh lãng phí tài nguyên tính toán. Tính hiệu quả của thuật toán thường được đánh giá thông qua **độ phức tạp thời gian và không gian**.

Ví dụ:

- **Thuật toán tìm kiếm tuần tự có độ phức tạp O(n),** nghĩa là số lần kiểm tra tỷ lệ thuận với kích thước dữ liệu.
- **Thuật toán tìm kiếm nhị phân có độ phức tạp O(log n),** cho phép tìm kiếm nhanh hơn nhiều trên tập dữ liệu lớn.

6. Tính phổ dụng (Generality)

Một thuật toán tốt có thể áp dụng cho nhiều bài toán có tính chất tương tự, không bị giới hạn bởi một trường hợp cụ thể.

Ví dụ:

- Thuật toán sắp xếp nhanh (Quicksort) có thể áp dụng cho cả **mảng số nguyên** và **chuỗi ký tự**.
- Thuật toán Dijkstra có thể áp dụng cho các bài toán **định tuyến mạng**, **hệ thống giao thông** và **trí tuệ nhân tạo**.

Bài tập nhóm

Yêu cầu: Phân tích thuật toán tính tổng từ 1 đến n và xác định các đặc trưng của nó theo từng tiêu chí trên.

Gợi ý thảo luận:

1. Xác định đầu vào và đầu ra của thuật toán.
2. Thuật toán có đảm bảo tính dừng không? Nếu có, tại bước nào?
3. Nếu áp dụng thuật toán này cho số nguyên âm, nó có còn mang tính tổng quát không? Nếu không, làm thế nào để mở rộng thuật toán?

Các đặc trưng của thuật toán giúp xác định tính hợp lệ, hiệu quả và khả năng ứng dụng trong thực tế. Việc thiết kế một thuật toán không chỉ đòi hỏi đảm bảo tính đúng đắn mà còn phải cân nhắc đến hiệu suất và khả năng tổng quát hóa cho nhiều bài toán khác nhau.

2.1.4. Các cách biểu diễn thuật toán

Một thuật toán có thể được biểu diễn theo nhiều cách khác nhau để giúp con người và máy tính dễ dàng hiểu và thực thi. Dưới đây là ba phương pháp phổ biến:

1. Biểu diễn bằng ngôn ngữ tự nhiên (Giả lệnh - Natural Language/Pseudocode in Plain Language)

Ngôn ngữ tự nhiên sử dụng câu văn để mô tả các bước của thuật toán mà không cần tuân theo quy tắc chặt chẽ của ngôn ngữ lập trình. Cách biểu diễn này giúp mô tả ý tưởng thuật toán một cách đơn giản nhưng có thể gây mơ hồ do cách diễn đạt không nhất quán.

Ví dụ: Thuật toán tính giai thừa của một số nguyên dương n

1. Nhập số nguyên **n**.
2. Nếu **n = 0** hoặc **n = 1**, trả về kết quả **1**.
3. Khởi tạo biến **giaiThua = 1**.

4. Duyệt từ **i = 2** đến **n**, nhân **giaiThua** với **i**.
5. Xuất kết quả **giaiThua**.

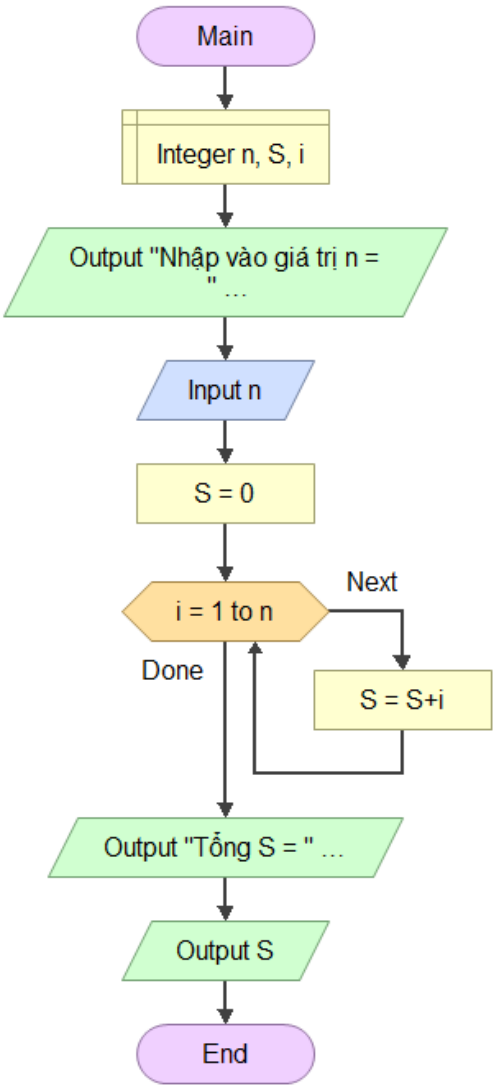
Nhược điểm của phương pháp này là dễ gây hiểu nhầm nếu không được viết rõ ràng, đặc biệt khi mô tả các thuật toán phức tạp.

2. Biểu diễn bằng sơ đồ khối (Flowchart)

Sơ đồ khối là cách biểu diễn trực quan sử dụng các hình học tiêu chuẩn để mô tả luồng thực thi của thuật toán. Một số ký hiệu quan trọng trong sơ đồ khối bao gồm:

- **Hình oval (◯)**: Điểm bắt đầu hoặc kết thúc.
- **Hình chữ nhật (▭)**: Biểu diễn một thao tác xử lý.
- **Hình thoi (◆)**: Biểu diễn một điều kiện kiểm tra.
- **Mũi tên (→)**: Chỉ hướng thực thi của thuật toán.

Ví dụ: Sơ đồ khối thuật toán tính S=1+2+3+…+n



Lưu ý: Sơ đồ khối giúp trực quan hóa thuật toán nhưng có thể trở nên phức tạp khi biểu diễn thuật toán có nhiều vòng lặp hoặc điều kiện rẽ nhánh phức tạp.

3. Biểu diễn bằng mã giả (Pseudocode)

Mã giả là một phương pháp biểu diễn thuật toán có cú pháp gần giống ngôn ngữ lập trình nhưng không phụ thuộc vào một ngôn ngữ cụ thể. Cách viết này giúp thuật toán dễ dàng được chuyển đổi sang mã nguồn thực tế.

Ví dụ: Thuật toán tính giai thừa bằng mã giả

```
Algorithm GiaiThua(n)
    Input: Một số nguyên dương n
    Output: Giá trị n!

    If n = 0 or n = 1 then
        Return 1
    End If

    giaiThua ← 1
    For i ← 2 to n do
        giaiThua ← giaiThua * i
    End For
    Return giaiThua
End Algorithm
```

Ưu điểm của mã giả là giúp mô tả thuật toán một cách chính xác, dễ hiểu hơn so với ngôn ngữ tự nhiên nhưng vẫn duy trì tính khái quát hơn so với một ngôn ngữ lập trình cụ thể.

Thực hành

1. Viết thuật toán **tính giai thừa** bằng **ba cách biểu diễn** trên:
    - Ngôn ngữ tự nhiên
    - Sơ đồ khối
    - Mã giả
2. Vẽ sơ đồ khối thuật toán **tìm số lớn nhất trong danh sách** số nguyên.

Mỗi phương pháp biểu diễn thuật toán có ưu điểm và nhược điểm riêng. Ngôn ngữ tự nhiên dễ hiểu nhưng có thể gây nhập nhằng, sơ đồ khối trực quan nhưng phức tạp với thuật toán lớn, còn mã giả giúp mô tả thuật toán chặt chẽ và dễ chuyển đổi sang lập

trình. Lựa chọn phương pháp biểu diễn phù hợp tùy thuộc vào mục đích sử dụng và đối tượng đọc thuật toán.

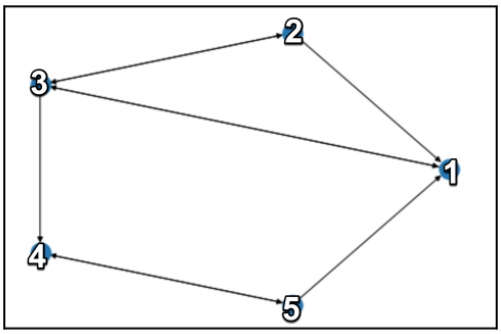
2.1.5. Ví dụ ứng dụng thực tế của thuật toán

Thuật toán đóng vai trò quan trọng trong nhiều hệ thống và ứng dụng hiện đại. Dưới đây là một số ví dụ điển hình về cách thuật toán được áp dụng trong thực tế để xử lý dữ liệu, tối ưu hóa và tự động hóa quy trình.

1. Google Search: Thuật toán PageRank

Google sử dụng thuật toán **PageRank** để xếp hạng các trang web dựa trên mức độ quan trọng của chúng trong hệ thống tìm kiếm.

- **Nguyên lý hoạt động:**
    - Mô hình hóa internet dưới dạng **đồ thị có hướng**, trong đó mỗi trang web là một **đỉnh (node)** và liên kết giữa các trang là **cạnh (edge)**.
    - Một trang web càng có nhiều liên kết từ các trang khác (đặc biệt là các trang có độ uy tín cao), thì điểm PageRank của nó càng lớn.
    - Thuật toán sử dụng phương pháp **lặp lại (iterative approach)** để cập nhật điểm PageRank cho đến khi hội tụ.



Hình 2. Đồ thị liên kết trong thuật toán PageRank

```
[6] G, p = createPageRank(myWeb)
    print (G)
      1        2          3          4          5
[[0.        0.5        0.33333333 0.         0.5       ]
 [0.        0.         0.33333333 0.         0.        ]
 [1.        0.5        0.         0.         0.        ]
 [0.        0.         0.33333333 0.         0.5       ]
 [0.        0.         0.         1.         0.        ]]
```

Hình 3. Ma trận chuyển đổi trong thuật toán PageRank

- **Ứng dụng:**

- Cải thiện độ chính xác của công cụ tìm kiếm.
- Đánh giá mức độ ảnh hưởng của các trang web trong hệ sinh thái internet.

- **Độ phức tạp:**
  - **O(N log N)** trong trường hợp tối ưu với hệ thống phân tán.
  - Với dữ liệu web ngày càng mở rộng, Google kết hợp **Machine Learning** và **công cụ tìm kiếm ngữ nghĩa** để cải thiện thuật toán.

2. Amazon: Hệ thống gợi ý sản phẩm (Recommendation System)

Amazon và các nền tảng thương mại điện tử sử dụng thuật toán **Recommendation System** để gợi ý sản phẩm phù hợp cho người dùng.

- **Nguyên lý hoạt động:**
  - Hệ thống phân tích **hành vi mua sắm** và **lịch sử tìm kiếm** của khách hàng.
  - Áp dụng các thuật toán **Machine Learning**, chẳng hạn như **Collaborative Filtering** (lọc cộng tác) và **Content-Based Filtering** (lọc dựa trên nội dung).
  - Tính toán mức độ tương đồng giữa người dùng hoặc giữa các sản phẩm để đưa ra dự đoán.
- **Ứng dụng:**
  - Cá nhân hóa trải nghiệm mua sắm.
  - Tăng doanh thu bằng cách tối ưu hóa chiến lược bán hàng.
- **Độ phức tạp:**
  - Hệ thống có thể xử lý hàng triệu giao dịch mỗi giây bằng cách sử dụng **Big Data** và **Deep Learning** để cải thiện tốc độ đề xuất.

3. Thuật toán hoán đổi giá trị hai biến

Trong lập trình, hoán đổi giá trị giữa hai biến là một thao tác cơ bản và có nhiều cách tiếp cận khác nhau.

- **Sử dụng biến tạm:**

```
int a = 5, b = 10;
int temp = a;
a = b;
b = temp;
```

  - **Ưu điểm:** Đơn giản, dễ hiểu.
  - **Nhược điểm:** Cần thêm một biến tạm, tiêu tốn bộ nhớ.

- **Sử dụng phép toán XOR để tối ưu:**

```
a = a ^ b;
b = a ^ b;
a = a ^ b;
```

  - **Ưu điểm:** Không cần sử dụng biến tạm, tiết kiệm bộ nhớ.
  - **Nhược điểm:** Khó hiểu hơn đối với người mới học lập trình.

- **Ứng dụng:**
  - Hoán đổi giá trị trong các chương trình xử lý dữ liệu tối ưu.
  - Được sử dụng trong các thuật toán mã hóa nhị phân.

4. Thuật toán tìm phần tử nhỏ nhất trong danh sách

Tìm phần tử nhỏ nhất trong một danh sách là một bài toán quan trọng trong xử lý dữ liệu, thường được sử dụng trong sắp xếp và tìm kiếm.

- **Nguyên lý hoạt động:**
  - Duyệt qua từng phần tử trong danh sách.
  - So sánh và cập nhật giá trị nhỏ nhất.
- **Độ phức tạp:**
  - **O(n)** do cần duyệt qua toàn bộ danh sách một lần.

- **Mã giả:**

```
Algorithm TimMin(A, n)
    Input: Danh sách A gồm n phần tử
    Output: Giá trị nhỏ nhất trong danh sách

    min ← A[0]
    For i ← 1 to n-1 do
        If A[i] < min then
            min ← A[i]
        End If
    End For

    Return min
End Algorithm
```

- **Sơ đồ khối:**

1. Bắt đầu.
2. Nhập danh sách **A** có **n** phần tử.
3. Gán **min = A[0]**.
4. Lặp qua các phần tử còn lại trong danh sách, cập nhật giá trị nhỏ nhất.
5. Xuất kết quả **min**.
6. Kết thúc.

- **Ứng dụng:**
    - Tìm giá trị nhỏ nhất trong tập dữ liệu lớn (ví dụ: giá cổ phiếu thấp nhất trong ngày).
    - Là bước cơ bản trong các thuật toán sắp xếp như **Selection Sort**.

Các ví dụ trên minh họa cách thuật toán được ứng dụng trong các hệ thống thực tế, từ công cụ tìm kiếm, thương mại điện tử đến các bài toán lập trình cơ bản. Hiểu rõ nguyên lý hoạt động, độ phức tạp và ứng dụng của thuật toán giúp lập trình viên thiết kế các giải pháp hiệu quả hơn cho các bài toán thực tế.

2.2: THUẬT TOÁN TÌM KIẾM

2.2.1. Bài toán tìm kiếm

Khái niệm tìm kiếm

Tìm kiếm là một trong những thao tác quan trọng trong khoa học máy tính, cho phép xác định vị trí của một phần tử trong tập hợp dữ liệu có cấu trúc hoặc không có cấu trúc. Các thuật toán tìm kiếm đóng vai trò quan trọng trong nhiều lĩnh vực, từ cơ sở dữ liệu, hệ điều hành đến trí tuệ nhân tạo.

Ứng dụng thực tế

1. **Công cụ tìm kiếm (Google Search)**:
    - Thuật toán tìm kiếm giúp xác định và xếp hạng các trang web chứa từ khóa mà người dùng nhập vào.
    - Các hệ thống tìm kiếm sử dụng nhiều thuật toán khác nhau, bao gồm tìm kiếm tuyến tính, nhị phân và tìm kiếm trên đồ thị.
2. **Hệ điều hành (Tìm kiếm tệp tin trong máy tính)**:
    - Các hệ điều hành cung cấp tính năng tìm kiếm tệp tin dựa trên tên tệp, nội dung hoặc thuộc tính của tệp.
    - Hệ thống tệp tin có thể sử dụng thuật toán tìm kiếm tuần tự hoặc tìm kiếm nhị phân, tùy thuộc vào cách dữ liệu được lưu trữ.
3. **Hệ quản trị cơ sở dữ liệu (CSDL)**:
    - Trong các hệ thống quản lý cơ sở dữ liệu quan hệ (RDBMS), tìm kiếm dữ liệu là một thao tác cơ bản.
    - Các chỉ mục (index) trong CSDL giúp tối ưu hóa tìm kiếm, tương tự như thuật toán tìm kiếm nhị phân.

2.2.2. Thuật toán tìm kiếm tuần tự

Nguyên lý hoạt động

Thuật toán tìm kiếm tuần tự (Linear Search) là phương pháp tìm kiếm đơn giản nhất, trong đó từng phần tử trong danh sách được kiểm tra theo thứ tự cho đến khi tìm thấy giá trị cần tìm hoặc đã duyệt qua toàn bộ danh sách.

- **Nếu tìm thấy phần tử mong muốn**, thuật toán trả về vị trí của phần tử đó.
- **Nếu không tìm thấy**, thuật toán trả về một giá trị báo hiệu (thường là -1).

Độ phức tạp

Thuật toán này có độ phức tạp thời gian là **O(n)**, nghĩa là trong trường hợp xấu nhất, thuật toán phải duyệt qua toàn bộ danh sách gồm nnn phần tử để tìm kiếm giá trị cần tìm.

Ví dụ minh họa

Giả sử cần tìm số 5 trong danh sách **[3, 1, 4, 5, 9]**, thuật toán sẽ thực hiện các bước sau:

1. Kiểm tra phần tử đầu tiên (3) → Không khớp.
2. Kiểm tra phần tử thứ hai (1) → Không khớp.
3. Kiểm tra phần tử thứ ba (4) → Không khớp.
4. Kiểm tra phần tử thứ tư (5) → Khớp → Kết thúc tìm kiếm.

Cài đặt thuật toán tìm kiếm tuần tự bằng Java

```
public class LinearSearch {
    public static int search(int[] arr, int x) {
        for (int i = 0; i < arr.length; i++) {
            if (arr[i] == x) {
                return i; // Trả về vị trí tìm thấy
            }
        }
        return -1; // Không tìm thấy
    }

    public static void main(String[] args) {
        int[] arr = {3, 1, 4, 5, 9};
        int x = 5;
        int result = search(arr, x);
        System.out.println(result != -1 ? "Phần tử được tìm thấy tại vị trí " + result
: "Không tìm thấy phần tử");
    }
}
```

Thực hành

Sinh viên triển khai thuật toán tìm kiếm tuần tự bằng ngôn ngữ lập trình Java hoặc Python, sau đó thử nghiệm trên danh sách số nguyên có độ dài khác nhau để so sánh thời gian thực thi.

2.2.3. Thuật toán tìm kiếm nhị phân

Nguyên lý hoạt động

Thuật toán tìm kiếm nhị phân (Binary Search) áp dụng cho danh sách **đã được sắp xếp**. Phương pháp này hoạt động bằng cách chia đôi danh sách tại mỗi bước tìm kiếm:

- So sánh phần tử cần tìm với phần tử ở giữa danh sách.
- Nếu phần tử cần tìm nhỏ hơn phần tử giữa, thuật toán tiếp tục tìm kiếm trong nửa đầu danh sách.
- Nếu phần tử cần tìm lớn hơn phần tử giữa, thuật toán tìm kiếm trong nửa cuối danh sách.
- Quá trình này lặp lại cho đến khi tìm thấy phần tử hoặc danh sách con trở nên rỗng.

Độ phức tạp

Thuật toán tìm kiếm nhị phân có độ phức tạp thời gian là **O(log n)**, nghĩa là số lần so sánh cần thực hiện giảm theo cấp số nhân khi kích thước danh sách tăng lên. Điều này làm cho thuật toán hiệu quả hơn nhiều so với tìm kiếm tuần tự khi làm việc với tập dữ liệu lớn.

Ví dụ minh họa

Giả sử cần tìm số 5 trong danh sách đã sắp xếp **[1, 3, 4, 5, 9, 12, 15]**, thuật toán sẽ thực hiện các bước sau:

1. Xác định phần tử giữa danh sách: 4 (tại vị trí index 2).
2. So sánh với giá trị cần tìm: 4<5 → Tìm kiếm tiếp trong nửa sau danh sách.

3. Xác định phần tử giữa của nửa sau: 9 (tại vị trí index 4).
4. So sánh với giá trị cần tìm: 9>5 → Tìm kiếm tiếp trong nửa trước của phân đoạn này.
5. Phần tử ở vị trí index 3 là 5 → Kết thúc tìm kiếm.

Cài đặt thuật toán tìm kiếm nhị phân bằng Java

```
public class BinarySearch {
    public static int search(int[] arr, int x) {
        int left = 0, right = arr.length - 1;
        while (left <= right) {
            int mid = left + (right - left) / 2;
            if (arr[mid] == x) {
                return mid; // Trả về vị trí tìm thấy
            }
            if (arr[mid] < x) {
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        }
        return -1; // Không tìm thấy
    }

    public static void main(String[] args) {
        int[] arr = {1, 3, 4, 5, 9, 12, 15};
        int x = 5;
        int result = search(arr, x);
        System.out.println(result != -1 ? "Phần tử được tìm thấy tại vị trí " + result :
"Không tìm thấy phần tử");
    }
}
```

Thực hành nhóm

Sinh viên viết thuật toán tìm kiếm nhị phân và thử nghiệm trên danh sách gồm 1 triệu phần tử. So sánh thời gian thực thi giữa tìm kiếm tuần tự và tìm kiếm nhị phân để đánh giá hiệu quả của từng phương pháp.

Tổng kết

- Phân biệt thuật toán **tìm kiếm tuần tự** và **tìm kiếm nhị phân**, nắm vững ưu nhược điểm của từng phương pháp.
- Thực hành lập trình thuật toán tìm kiếm và đánh giá hiệu suất của từng phương pháp.
- Chuẩn bị cho nội dung tiếp theo về **thuật toán xử lý số** và **sắp xếp dãy số** trong buổi học kế tiếp.

BÀI TẬP

I. Câu hỏi trắc nghiệm

Mục tiêu: Kiểm tra kiến thức về tổng quan môn học, khái niệm thuật toán, đặc trưng của thuật toán, và cách biểu diễn thuật toán.

Phần 1: Khái niệm thuật toán và toán rời rạc

Câu 1: Toán rời rạc tập trung vào nghiên cứu đối tượng nào sau đây?
A. Các tập hợp liên tục như số thực
B. Các cấu trúc dữ liệu rời rạc như đồ thị, cây, tổ hợp
C. Giải tích vi phân và tích phân
D. Mô hình hóa hệ thống động lực học

Câu 2: Thuật toán được định nghĩa là gì?
A. Một tập hợp hữu hạn các bước xác định để giải quyết một bài toán cụ thể
B. Một đoạn mã nguồn trong chương trình máy tính
C. Một phương pháp lập trình nâng cao
D. Một mô hình thiết kế phần mềm

Câu 3: Đâu là một trong những tiêu chí quan trọng của một thuật toán?
A. Phải có số bước vô hạn
B. Có thể chạy mãi mãi để tạo ra nhiều kết quả khác nhau
C. Đưa ra kết quả đúng đắn và dừng sau một số hữu hạn bước
D. Không cần đầu vào hoặc đầu ra

Phần 2: Đặc trưng của thuật toán

Câu 4: Tính chất nào đảm bảo rằng thuật toán luôn có một kết quả xác định khi chạy với cùng một đầu vào?
A. Tính hiệu quả
B. Tính xác định
C. Tính tổng quát
D. Tính phổ dụng

Câu 5: Tính chất nào của thuật toán yêu cầu thuật toán phải kết thúc sau một số bước hữu hạn?
A. Tính xác định
B. Tính phổ dụng
C. Tính dừng
D. Tính hiệu quả

Câu 6: Một thuật toán có đầu vào nhưng không có đầu ra có phải là thuật toán hợp lệ không?
A. Có, miễn là nó thực thi được
B. Không, vì thuật toán phải tạo ra ít nhất một đầu ra có ý nghĩa
C. Có, nếu nó chỉ thực hiện xử lý dữ liệu nội bộ
D. Không, vì thuật toán không thể có đầu vào

Phần 3: Các pha của thuật toán

Câu 7: Giai đoạn nào của thuật toán liên quan đến việc lựa chọn chiến lược giải quyết bài toán, chẳng hạn như **chia để trị, quy hoạch động**?
A. Giai đoạn thiết kế
B. Giai đoạn cài đặt
C. Giai đoạn đánh giá
D. Giai đoạn triển khai

Câu 8: Đánh giá độ phức tạp thời gian và không gian của một thuật toán được thực hiện trong giai đoạn nào?
A. Giai đoạn thiết kế
B. Giai đoạn cài đặt
C. Giai đoạn đánh giá
D. Giai đoạn triển khai

Câu 9: Thuật toán nào dưới đây không phải là thuật toán xác định (Exact Algorithm)?
A. Thuật toán Dijkstra tìm đường đi ngắn nhất
B. Thuật toán Kruskal xây dựng cây khung nhỏ nhất
C. Thuật toán tham lam giải bài toán cái túi (Knapsack Problem)
D. Thuật toán tìm kiếm nhị phân

Phần 4: Các cách biểu diễn thuật toán

Câu 10: Trong sơ đồ khối, hình chữ nhật biểu diễn điều gì?
A. Điểm bắt đầu hoặc kết thúc thuật toán
B. Một thao tác xử lý hoặc tính toán
C. Một điều kiện kiểm tra
D. Một bước nhập/xuất dữ liệu

Câu 11: Trong mã giả (pseudocode), điều nào sau đây không đúng?
A. Mã giả có thể được thực thi trực tiếp trên máy tính
B. Mã giả giúp mô tả thuật toán mà không phụ thuộc vào ngôn ngữ lập trình cụ thể
C. Mã giả cần rõ ràng và có cấu trúc logic
D. Mã giả có thể dễ dàng chuyển đổi sang mã nguồn của ngôn ngữ lập trình

Câu 12: Khi sử dụng **ngôn ngữ tự nhiên** để biểu diễn thuật toán, nhược điểm chính là gì?
A. Thiếu tính trực quan
B. Dễ gây hiểu nhầm do cách diễn đạt không rõ ràng
C. Không thể chuyển đổi thành mã lập trình
D. Cả A, B và C đều đúng

Phần 5: Ứng dụng thực tế của thuật toán

Câu 13: Google sử dụng thuật toán nào để đánh giá mức độ quan trọng của một trang web?
A. Thuật toán tìm kiếm nhị phân
B. Thuật toán PageRank
C. Thuật toán sắp xếp chèn
D. Thuật toán Floyd-Warshall

Câu 14: Hệ thống gợi ý sản phẩm của Amazon dựa trên thuật toán nào?
A. Thuật toán tìm kiếm tuần tự
B. Thuật toán Recommendation System
C. Thuật toán hoán đổi hai số nguyên
D. Thuật toán tìm phần tử nhỏ nhất

Câu 15: Khi muốn tìm phần tử nhỏ nhất trong một danh sách không sắp xếp, thuật toán nào được sử dụng?
A. Tìm kiếm nhị phân
B. Sắp xếp nổi bọt trước rồi lấy phần tử đầu tiên
C. Duyệt toàn bộ danh sách và ghi nhận giá trị nhỏ nhất
D. Sử dụng thuật toán PageRank

Phần 6: Thuật toán hoán đổi giá trị hai biến

Câu 16: Phương pháp nào giúp hoán đổi giá trị của hai biến mà không cần dùng biến tạm?
A. Sử dụng phép toán cộng và trừ
B. Sử dụng phép XOR
C. Cả A và B đều đúng
D. Không thể hoán đổi mà không cần biến tạm

Câu 17: Khi sử dụng phép toán XOR để hoán đổi hai số, ưu điểm chính là gì?
A. Dễ hiểu hơn so với phương pháp dùng biến tạm
B. Tránh lỗi tràn số khi xử lý số nguyên lớn
C. Giảm thiểu việc sử dụng bộ nhớ bổ sung
D. Không có ưu điểm đáng kể so với phương pháp thông thường

Phần 7: Độ phức tạp của thuật toán

Câu 18: Độ phức tạp thời gian của thuật toán tìm phần tử nhỏ nhất trong danh sách có n phần tử là bao nhiêu?
A. O(1)
B. O(log n)
C. O(n)
D. $O(n^2)$

Câu 19: Thuật toán tìm kiếm nhị phân có độ phức tạp thời gian trong trường hợp xấu nhất là:
A. O(1)
B. O(n)
C. O(log n)
D. $O(n^2)$

Câu 20: Trong một danh sách đã sắp xếp gồm 1 triệu phần tử, tìm kiếm nhị phân cần tối đa bao nhiêu lần kiểm tra để tìm một phần tử?
A. 1000000
B. 500000
C. 20
D. 100

BÀI TẬP TỰ GIẢI

Bài tập tự giải giúp sinh viên áp dụng các kiến thức đã học về thuật toán, phân tích thuật toán, biểu diễn thuật toán và tìm kiếm tuần tự. Sinh viên cần thực hiện từng bài tập theo yêu cầu và tự xây dựng lời giải dựa trên hướng dẫn.

Bài 1: Viết thuật toán hoán đổi giá trị hai biến

Yêu cầu
Viết thuật toán bằng ngôn ngữ tự nhiên để hoán đổi giá trị của hai biến theo hai cách:

1. **Sử dụng biến tạm**
2. **Không sử dụng biến tạm**

Hướng dẫn

- Xác định hai biến cần hoán đổi giá trị.
- Sử dụng phép gán để thay đổi giá trị của hai biến theo từng phương pháp.
- Viết thuật toán dưới dạng các bước cụ thể theo ngôn ngữ tự nhiên.

Bài 2: Viết thuật toán tính tổng S=1+2+...+n

Yêu cầu
Viết thuật toán tính tổng của dãy số từ 1 đến nnn theo hai cách:

1. **Sử dụng vòng lặp**
2. **Sử dụng công thức Gauss**

Hướng dẫn

- Xác định phương pháp tính tổng theo cách cộng dồn từng số một (vòng lặp).
- Xác định công thức toán học tính tổng dãy số (công thức Gauss).
- Viết thuật toán bằng ngôn ngữ tự nhiên cho từng phương pháp.

Bài 3: Phân tích đặc trưng của thuật toán tính giai thừa n!

Yêu cầu

Phân tích thuật toán tính giai thừa n! dựa trên các đặc trưng cơ bản của thuật toán:

- **Đầu vào**
- **Đầu ra**
- **Tính xác định**
- **Tính dừng**
- **Tính hiệu quả**

Hướng dẫn

- Xác định dữ liệu đầu vào và đầu ra của thuật toán.
- Kiểm tra xem thuật toán có đảm bảo các đặc trưng của một thuật toán hợp lệ không.
- Đánh giá hiệu suất của thuật toán khi nnn lớn.

Bài 4: Vẽ sơ đồ khối thuật toán tìm số lớn nhất trong danh sách số nguyên

Yêu cầu

Xây dựng sơ đồ khối biểu diễn thuật toán tìm phần tử có giá trị lớn nhất trong một danh sách số nguyên.

Hướng dẫn

- Xác định các bước thực hiện thuật toán tìm số lớn nhất.
- Sử dụng các ký hiệu chuẩn của sơ đồ khối:
    - Hình oval cho điểm bắt đầu và kết thúc.
    - Hình chữ nhật cho bước xử lý.
    - Hình thoi cho điều kiện so sánh.
- Thiết kế sơ đồ khối sao cho dễ hiểu và logic.

Bài 5: Cài đặt thuật toán tìm kiếm tuần tự trên danh sách số nguyên ngẫu nhiên

Yêu cầu

Cài đặt thuật toán tìm kiếm tuần tự để xác định xem một phần tử có tồn tại trong danh sách số nguyên hay không.

Hướng dẫn

- Xác định phương pháp duyệt từng phần tử trong danh sách và so sánh với giá trị cần tìm.
- Xác định điều kiện dừng khi tìm thấy phần tử hoặc khi duyệt hết danh sách.
- Viết thuật toán bằng ngôn ngữ tự nhiên hoặc mã giả trước khi cài đặt bằng ngôn ngữ lập trình.

BÀI TẬP LUYỆN TẬP

Bài tập luyện tập giúp sinh viên áp dụng các kiến thức đã học về thuật toán, tìm kiếm, tối ưu hóa thuật toán và xử lý dữ liệu danh sách. Các bài tập được sắp xếp từ dễ đến khó, yêu cầu sinh viên tư duy thuật toán và triển khai cài đặt.

Bài 1: Viết mã giả cho thuật toán tìm kiếm tuần tự

Yêu cầu

Viết mã giả để triển khai thuật toán tìm kiếm tuần tự trong một danh sách số nguyên.

Hướng dẫn

- Duyệt qua từng phần tử của danh sách theo thứ tự từ đầu đến cuối.
- Kiểm tra xem phần tử hiện tại có trùng với giá trị cần tìm không.
- Nếu tìm thấy, trả về vị trí của phần tử trong danh sách.
- Nếu duyệt hết danh sách mà không tìm thấy, trả về giá trị báo hiệu (ví dụ: -1).

Bài 2: Viết mã giả cho thuật toán tìm kiếm nhị phân

Yêu cầu

Viết mã giả để triển khai thuật toán tìm kiếm nhị phân trên một danh sách số nguyên đã được sắp xếp.

Hướng dẫn

- Xác định phạm vi tìm kiếm bằng hai biến chỉ mục left và right.
- Tính phần tử giữa danh sách và so sánh với giá trị cần tìm.
- Nếu phần tử giữa bằng giá trị cần tìm, trả về vị trí của phần tử.
- Nếu nhỏ hơn, tìm kiếm trong nửa sau danh sách.
- Nếu lớn hơn, tìm kiếm trong nửa trước danh sách.
- Tiếp tục chia đôi danh sách cho đến khi tìm thấy phần tử hoặc phạm vi tìm kiếm trở nên rỗng.

Bài 3: Cài đặt thuật toán tìm kiếm tuần tự bằng Java/Python

Yêu cầu

Cài đặt thuật toán tìm kiếm tuần tự trong danh sách số nguyên bằng ngôn ngữ lập trình Java hoặc Python.

Hướng dẫn

- Viết một hàm nhận vào danh sách và giá trị cần tìm.
- Lặp qua từng phần tử trong danh sách, kiểm tra điều kiện khớp.
- In ra vị trí của phần tử nếu tìm thấy hoặc thông báo nếu không tìm thấy.

Bài 4: Cài đặt thuật toán tìm kiếm nhị phân bằng Java/Python

Yêu cầu

Cài đặt thuật toán tìm kiếm nhị phân trên danh sách đã được sắp xếp bằng Java hoặc Python.

Hướng dẫn

- Viết một hàm nhận vào danh sách đã sắp xếp và giá trị cần tìm.

- Sử dụng phương pháp chia đôi để tìm kiếm.
- Kiểm tra và trả về vị trí của phần tử nếu tìm thấy, nếu không tìm thấy trả về -1.

Bài 5: So sánh hiệu suất giữa tìm kiếm tuần tự và tìm kiếm nhị phân trên danh sách 1000 phần tử

Yêu cầu
Chạy thử nghiệm để so sánh thời gian thực thi của hai thuật toán tìm kiếm tuần tự và tìm kiếm nhị phân trên danh sách có 1000 phần tử.
Hướng dẫn

- Sinh viên tạo danh sách số nguyên ngẫu nhiên có 1000 phần tử.
- Cài đặt thuật toán tìm kiếm tuần tự và tìm kiếm nhị phân.
- Chạy thử nghiệm với nhiều giá trị khác nhau và đo thời gian thực thi.
- So sánh và phân tích kết quả.

Bài 6: Viết thuật toán tìm số nhỏ nhất trong danh sách số nguyên

Yêu cầu
Xây dựng thuật toán tìm phần tử nhỏ nhất trong danh sách số nguyên.
Hướng dẫn

- Khởi tạo biến min bằng phần tử đầu tiên của danh sách.
- Duyệt qua danh sách, nếu gặp phần tử nhỏ hơn min, cập nhật min.
- Sau khi duyệt hết danh sách, min sẽ chứa giá trị nhỏ nhất.

Bài 7: Xây dựng thuật toán đếm số lần xuất hiện của một phần tử trong danh sách

Yêu cầu
Viết thuật toán đếm số lần xuất hiện của một giá trị bất kỳ trong danh sách số nguyên.
Hướng dẫn

- Khởi tạo biến đếm count bằng 0.
- Duyệt qua danh sách, mỗi lần gặp giá trị cần tìm thì tăng count lên 1.
- Kết thúc vòng lặp, in ra số lần xuất hiện của phần tử trong danh sách.

Bài 8: Viết thuật toán kiểm tra một danh sách có chứa số chẵn hay không

Yêu cầu
Viết thuật toán kiểm tra xem trong danh sách số nguyên có tồn tại ít nhất một số chẵn hay không.
Hướng dẫn

- Duyệt qua từng phần tử trong danh sách.
- Kiểm tra nếu có phần tử chia hết cho 2 thì kết luận danh sách có chứa số chẵn.
- Nếu duyệt hết danh sách mà không tìm thấy số chẵn, kết luận danh sách toàn số lẻ.

Bài 9: Viết thuật toán tìm phần tử lớn thứ hai trong danh sách

Yêu cầu
Viết thuật toán tìm phần tử có giá trị lớn thứ hai trong danh sách số nguyên.
Hướng dẫn

- Duyệt qua danh sách để tìm phần tử lớn nhất.
- Duyệt qua danh sách lần thứ hai để tìm phần tử lớn nhất thứ hai (nhỏ hơn phần tử lớn nhất nhưng lớn hơn tất cả phần còn lại).
- Xử lý trường hợp danh sách có các phần tử trùng nhau hoặc có ít hơn hai phần tử.

Bài 10: Viết thuật toán kiểm tra xem một số nguyên có phải là số nguyên tố hay không

Yêu cầu
Viết thuật toán kiểm tra tính nguyên tố của một số nguyên dương nnn.
Hướng dẫn

- Một số nguyên tố là số tự nhiên lớn hơn 1 và chỉ chia hết cho 1 và chính nó.
- Nếu nnn nhỏ hơn hoặc bằng 1, kết luận không phải số nguyên tố.
- Duyệt từ 2 đến n\sqrt{n}n, nếu nnn chia hết cho bất kỳ số nào trong khoảng này thì không phải số nguyên tố.
- Nếu không tìm thấy ước số nào khác ngoài 1 và nnn, kết luận nnn là số nguyên tố.

Yêu cầu nộp bài

- Sinh viên trình bày rõ ràng thuật toán, có giải thích và mô tả từng bước thực hiện.
- Bài nộp có thể ở dạng viết tay (với bài mã giả) hoặc soạn thảo trên Word/PDF.
- Với các bài lập trình, sinh viên cần nộp mã nguồn kèm theo kết quả chạy thử nghiệm.
- Hạn nộp: **Trước buổi học tiếp theo**.

BÀI TẬP DỰ ÁN NHÓM

Bài tập dự án nhóm yêu cầu sinh viên làm việc theo nhóm để phân tích, thiết kế và triển khai các thuật toán tìm kiếm trong các ứng dụng thực tế. Mỗi nhóm cần nghiên cứu, thử nghiệm và đánh giá hiệu quả của thuật toán.

Bài 1: Phân tích và triển khai hệ thống tìm kiếm tệp tin trên máy tính

Mô tả bài toán
Xây dựng một hệ thống mô phỏng tìm kiếm tệp tin trên máy tính dựa trên tên tệp. Hệ thống cần có khả năng tìm kiếm trong danh sách tệp tin lớn và so sánh hiệu suất giữa tìm kiếm tuần tự và tìm kiếm nhị phân.
Yêu cầu

- Thiết kế thuật toán tìm kiếm tệp tin dựa trên tên tệp.
- Triển khai thuật toán theo hai phương pháp:

1. **Tìm kiếm tuần tự** – Duyệt qua từng tệp tin và so sánh với từ khóa tìm kiếm.
2. **Tìm kiếm nhị phân** – Áp dụng khi danh sách tệp tin đã được sắp xếp theo tên.

- So sánh thời gian thực thi của hai phương pháp với tập dữ liệu có kích thước lớn.
- Đánh giá ưu điểm và hạn chế của mỗi phương pháp.

Hướng dẫn thực hiện

- **Phân tích yêu cầu:** Xác định cấu trúc lưu trữ danh sách tệp tin và cách nhập dữ liệu.
- **Thiết kế thuật toán:** Xác định phương thức tìm kiếm phù hợp và các bước thực hiện.
- **Viết mã giả (Pseudocode):** Mô tả logic thuật toán trước khi lập trình.
- **Cài đặt thuật toán:** Lập trình hệ thống tìm kiếm bằng Java hoặc Python.
- **Thử nghiệm:** Kiểm tra hiệu suất với các bộ dữ liệu có kích thước khác nhau.
- **Đánh giá:** So sánh kết quả giữa hai phương pháp tìm kiếm về thời gian thực thi và độ chính xác.

Kết quả mong đợi

- Một hệ thống tìm kiếm đơn giản hoạt động chính xác.
- Báo cáo phân tích thuật toán, kết quả thử nghiệm và so sánh hiệu suất giữa hai phương pháp.
- Mã nguồn chương trình và hướng dẫn sử dụng.

Bài 2: Xây dựng chương trình tìm kiếm tài liệu trong một thư viện số

Mô tả bài toán

Thiết kế và triển khai một chương trình tìm kiếm tài liệu trong một thư viện số. Chương trình cần có khả năng tìm kiếm nhanh chóng trong danh sách sách được sắp xếp theo tiêu đề.

Yêu cầu

- Phân tích yêu cầu của hệ thống tìm kiếm tài liệu.
- Thiết kế thuật toán tìm kiếm tối ưu dựa trên tiêu đề sách.
- Viết mã giả mô tả thuật toán trước khi lập trình.
- Cài đặt thuật toán tìm kiếm và thử nghiệm với tập dữ liệu lớn.
- Đánh giá hiệu quả của thuật toán dựa trên thời gian thực thi và độ chính xác.

Hướng dẫn thực hiện

- **Phân tích yêu cầu:** Xác định các tiêu chí tìm kiếm (theo tiêu đề, tác giả, từ khóa,...).
- **Thiết kế thuật toán:** Chọn phương pháp tìm kiếm phù hợp (ví dụ: tìm kiếm nhị phân nếu danh sách đã sắp xếp).
- **Viết mã giả (Pseudocode):** Biểu diễn thuật toán dưới dạng logic dễ hiểu trước khi lập trình.
- **Cài đặt chương trình:** Lập trình bằng Java hoặc Python, tối ưu hóa hiệu suất.
- **Thử nghiệm:** Chạy chương trình với dữ liệu mô phỏng gồm hàng nghìn đầu sách.

- **Đánh giá:** So sánh tốc độ tìm kiếm khi danh sách lớn dần, kiểm tra độ chính xác khi tìm kiếm với các từ khóa khác nhau.

Kết quả mong đợi

- Một hệ thống tìm kiếm tài liệu hiệu quả, hỗ trợ nhập từ khóa và trả về kết quả chính xác.
- Báo cáo phân tích thuật toán, kết quả thử nghiệm và tối ưu hóa.
- Mã nguồn chương trình và tài liệu hướng dẫn sử dụng.

Yêu cầu nộp bài

- **Báo cáo nhóm:** Mô tả bài toán, thuật toán sử dụng, kết quả thử nghiệm, đánh giá hiệu suất.
- **Mã nguồn chương trình:** Được tổ chức rõ ràng, có chú thích và hướng dẫn chạy thử.
- **Bảng so sánh hiệu suất:** So sánh thời gian thực thi của các thuật toán trên các tập dữ liệu khác nhau.
- **Thời hạn nộp bài: Trước buổi học tiếp theo**.

Dự án này giúp sinh viên nâng cao **kỹ năng làm việc nhóm, tư duy thuật toán, tối ưu hóa tìm kiếm và triển khai ứng dụng thực tế**.