**Họ và tên: Nguyễn Phạm Thành Hưng**

**Mã SV: 6151071056**

**1.2 Cơ bản về giải quyết vấn đề thuật toán**

**Hiểu vấn đề**

Điều đầu tiên cần làm trước khi thiết kế một thuật toán là để hiểu hoàn toàn vấn đề. Đọc mô tả của vấn đề một cách cẩn thận và đặt câu hỏi nếu bạn có bất kỳ nghi ngờ nào về vấn đề này, hãy làm một vài ví dụ nhỏ bằng tay, hãy nghĩ về những trường hợp đặc biệt và hỏi lại nếu cần.

Nếu vấn đề trong câu hỏi là một trong số đó, bạn có thể sử dụng thuật toán đã biết để giải quyết nó. Tất nhiên, nó giúp hiểu được một thuật toán như thế nào và để biết điểm mạnh và điểm yếu của nó, đặc biệt nếu bạn phải chọn giữa các thuật toán có sẵn.

Nhưng thường thì bạn sẽ không tìm thấy thuật toán sẵn có và sẽ phải thiết kế riêng. Hãy nhớ rằng một thuật toán chính xác không phải là một thứ hoạt động hầu hết thời gian, nhưng nó hoạt động đúng với tất cả các đầu vào hợp pháp.

Đừng hà tiện vào bước đầu tiên của quá trình giải quyết vấn đề thuật toán ; nếu không, bạn sẽ có nguy cơ làm điều chỉnh không cần thiết.

**Xác định khả năng của thiết bị tính toán**

Khi bạn hoàn toàn hiểu một vấn đề, bạn cần xác định khả năng của thiết bị tính toán mà thuật toán dành cho. Phần lớn các thuật toán được sử dụng ngày nay vẫn được định sẵn là được lập trình cho một máy tính gần giống với máy von Neumann, một kiến ​​trúc máy tính được phác thảo bởi nhà toán học người Mỹ gốc Hungary nổi tiếng John von Neumann (1903–1957), cộng tác với A. Burks và H. Goldstine, vào năm 1946. Bản chất của kiến ​​trúc này được nắm bắt bởi cái gọi là máy truy cập ngẫu nhiên (RAM). Giả định trung tâm của nó là các lệnh được thực hiện lần lượt, từng thao tác tại một thời điểm. các thuật toán toán học được thiết kế để thực thi trên các máy như vậy được gọi là thuật toán tuần tự.

**Lựa chọn giữa giải quyết vấn đề chính xác và gần đúng**

Quyết định chính tiếp theo là lựa chọn giữa giải quyết vấn đề một cách chính xác hoặc giải quyết nó một cách gần đúng. Tại sao một người lại chọn một thuật toán gần đúng? Đầu tiên, có những vấn đề quan trọng không thể được giải quyết chính xác đối với hầu hết các trường hợp của chúng. Thứ hai, các thuật toán có sẵn để giải quyết chính xác một vấn đề có thể chậm đến mức không thể chấp nhận được do độ phức tạp nội tại của vấn đề. Thứ ba, thuật toán xấp xỉ có thể là một phần của thuật toán phức tạp hơn giúp giải quyết chính xác một vấn đề.

**Kỹ thuật thiết kế thuật toán**

Kỹ thuật thiết kế thuật toán là gì?

Kỹ thuật thiết kế thuật toán (hay “chiến lược” hoặc “mô hình”) là một cách tiếp cận chung để giải quyết các vấn đề theo thuật toán có thể áp dụng cho nhiều vấn đề từ các lĩnh vực máy tính khác nhau.

Kiểm tra mục lục của cuốn sách này và bạn sẽ thấy rằng phần lớn các chương của sách được dành cho các kỹ thuật thiết kế riêng lẻ. Họ chắt lọc một vài ý tưởng chính đã được chứng minh là hữu ích trong việc thiết kế các thuật toán. Học những kỹ thuật này là vô cùng quan trọng vì những lý do sau đây.

Đầu tiên, họ cung cấp hướng dẫn thiết kế các thuật toán cho các bài toán mới, tức là các bài toán chưa có thuật toán thỏa đáng được biết đến. Nó không phải là sự thật, của tất nhiên, mỗi kỹ thuật chung này sẽ nhất thiết phải áp dụng cho mọi vấn đề bạn có thể gặp phải. Nhưng kết hợp lại với nhau, chúng tạo thành một bộ sưu tập công cụ mạnh mẽ mà bạn sẽ thấy khá tiện dụng trong học tập và công việc của mình.

Thứ hai, thuật toán là nền tảng của khoa học máy tính. Mọi ngành khoa học đều quan tâm đến việc phân loại đối tượng chính của nó, và khoa học máy tính cũng không phải là ngoại lệ. Các kỹ thuật thiết kế thuật toán giúp chúng ta có thể phân loại các thuật toán theo một ý tưởng thiết kế cơ bản; do đó, chúng có thể dùng như một cách tự nhiên để vừa phân loại vừa nghiên cứu các thuật toán.

**Thiết kế thuật toán và cấu trúc dữ liệu**

Mặc dù các kỹ thuật thiết kế thuật toán cung cấp một tập hợp các phương pháp tiếp cận chung mạnh mẽ để giải quyết vấn đề theo thuật toán, việc thiết kế một thuật toán cho một vấn đề cụ thể vẫn có thể là một nhiệm vụ đầy thách thức. Với thực tế, cả việc lựa chọn trong số các kỹ thuật chung và áp dụng chúng đều trở nên dễ dàng hơn, nhưng chúng hiếm khi dễ dàng.

Tất nhiên, người ta nên chú ý đến việc lựa chọn cấu trúc dữ liệu thích hợp cho các hoạt động được thực hiện bởi thuật toán. Nhiều năm trước, một cuốn sách giáo khoa có ảnh hưởng đã tuyên bố tầm quan trọng cơ bản của cả thuật toán và cấu trúc dữ liệu đối với lập trình máy tính bằng chính tiêu đề của nó: Thuật toán + Cấu trúc dữ liệu = Chương trình [Wir76]. Trong thế giới mới của lập trình hướng đối tượng, cấu trúc dữ liệu vẫn cực kỳ quan trọng đối với cả thiết kế và phân tích thuật toán.

**Phương pháp chỉ định thuật toán**

Khi bạn đã thiết kế một thuật toán, bạn cần chỉ định nó theo một cách nào đó. Trong Phần 1.1, để cung cấp cho bạn một ví dụ, thuật toán của Euclid được mô tả bằng lời (ở dạng miễn phí và cũng là dạng từng bước) và ở dạng mã giả. Đây là hai tùy chọn được sử dụng rộng rãi nhất hiện nay để xác định các thuật toán.

Sử dụng một ngôn ngữ tự nhiên có một sức hấp dẫn rõ ràng; tuy nhiên, sự mơ hồ cố hữu của bất kỳ ngôn ngữ tự nhiên nào khiến cho việc mô tả các thuật toán ngắn gọn và rõ ràng trở nên khó khăn một cách đáng ngạc nhiên. Tuy nhiên, có thể làm được điều này là một kỹ năng quan trọng mà bạn nên cố gắng phát triển trong quá trình học thuật toán.

Mã giả là một hỗn hợp của ngôn ngữ tự nhiên và ngôn ngữ lập trình giống như các cấu trúc. Mã giả thường chính xác hơn ngôn ngữ tự nhiên và việc sử dụng nó thường mang lại các mô tả thuật toán ngắn gọn hơn.

Trong những ngày đầu của máy tính, phương tiện chủ đạo để chỉ định các thuật toán là một lưu đồ, một phương pháp thể hiện thuật toán bằng một tập hợp các hình dạng hình học được kết nối có chứa các mô tả về các bước của thuật toán. Kỹ thuật biểu diễn này đã được chứng minh là không thuận tiện đối với tất cả các thuật toán, trừ các thuật toán rất đơn giản; ngày nay, nó chỉ có thể được tìm thấy trong các sách thuật toán cũ.

Kỹ thuật của máy tính vẫn chưa đạt đến mức mô tả thuật toán bằng ngôn ngữ tự nhiên hoặc mã giả có thể được đưa trực tiếp vào máy tính điện tử. Thay vào đó, nó cần được chuyển đổi thành một chương trình máy tính được viết bằng một ngôn ngữ máy tính cụ thể. Chúng ta có thể coi một chương trình như vậy là một cách khác để chỉ định thuật toán, mặc dù tốt hơn nên coi đó là cách triển khai của thuật toán.

**Chứng minh tính đúng đắn của thuật toán**

Khi một thuật toán đã được chỉ định, bạn phải chứng minh tính đúng đắn của nó. Nghĩa là, bạn phải chứng minh rằng thuật toán mang lại kết quả bắt buộc cho mọi đầu vào hợp pháp trong một khoảng thời gian hữu hạn.

Đối với một số thuật toán, một bằng chứng về tính đúng đắn là khá dễ dàng; đối với những người khác, nó có thể khá phức tạp. Một kỹ thuật phổ biến để chứng minh tính đúng đắn là sử dụng quy nạp toán học vì các lần lặp lại của thuật toán cung cấp một chuỗi các bước tự nhiên cần thiết cho các chứng minh đó.

Khái niệm về tính đúng cho các thuật toán xấp xỉ ít đơn giản hơn so với các thuật toán chính xác. Đối với thuật toán gần đúng, chúng tôi thường muốn có thể chỉ ra rằng lỗi do thuật toán tạo ra không vượt quá giới hạn được xác định trước.

**Phân tích thuật toán**

Chúng tôi thường muốn các thuật toán của mình sở hữu một số phẩm chất. Sau sự đúng đắn, cho đến nay, điều quan trọng nhất là hiệu quả. Trên thực tế, có hai loại hiệu quả của thuật toán: hiệu quả về thời gian, cho biết tốc độ chạy của thuật toán và hiệu quả về không gian, cho biết nó sử dụng thêm bao nhiêu bộ nhớ.

Một đặc tính mong muốn khác của một thuật toán là tính đơn giản. Không giống như hiệu quả, có thể được xác định chính xác và nghiên cứu bằng sự nghiêm ngặt của toán học, sự đơn giản, giống như vẻ đẹp, ở một mức độ đáng kể trong mắt người xem. Đôi khi các thuật toán đơn giản hơn cũng hiệu quả hơn các giải pháp thay thế phức tạp hơn. Thật không may, nó không phải lúc nào cũng đúng, trong đó trong trường hợp cần phải thực hiện một thỏa hiệp sáng suốt.

Tuy nhiên, một đặc tính mong muốn khác của một thuật toán là tính tổng quát. Trên thực tế, có hai vấn đề ở đây: tính tổng quát của vấn đề mà thuật toán giải quyết và tập hợp các đầu vào mà nó chấp nhận. Về vấn đề đầu tiên, hãy lưu ý rằng đôi khi việc thiết kế một thuật toán cho một vấn đề được đặt ra theo các thuật ngữ chung hơn lại dễ dàng hơn.

Đối với tập hợp các đầu vào, mối quan tâm chính của bạn nên là thiết kế một thuật toán có thể xử lý một tập hợp các đầu vào tự nhiên cho vấn đề hiện tại.

Nếu bạn không hài lòng với tính hiệu quả, tính đơn giản hoặc tính tổng quát của thuật toán, bạn phải quay lại bảng vẽ và thiết kế lại thuật toán. Trên thực tế, ngay cả khi đánh giá của bạn là tích cực, nó vẫn đáng để tìm kiếm các giải pháp thuật toán khác. Bạn sẽ làm tốt nếu ghi nhớ nhận xét sau đây của Antoine de Saint-Exupery, nhà văn, phi công và nhà thiết kế máy bay người Pháp: “Một nhà thiết kế biết rằng anh ta đã đạt đến sự hoàn hảo không phải khi không có không còn là bất cứ thứ gì để thêm vào, nhưng khi không còn bất cứ điều gì để lấy đi.”

**Mã hóa thuật toán**

Hầu hết các thuật toán cuối cùng được thực hiện dưới dạng các chương trình máy tính. Lập trình một thuật toán thể hiện cả nguy cơ và cơ hội. Nguy hiểm nằm ở khả năng thực hiện chuyển đổi từ một thuật toán sang một chương trình không chính xác hoặc rất kém hiệu quả.

Như một vấn đề thực tế, tính hợp lệ của các chương trình vẫn được thiết lập bằng cách thử nghiệm. Kiểm tra các chương trình máy tính là một nghệ thuật hơn là một khoa học, nhưng điều đó không có nghĩa là không có gì trong đó để học.

Tất nhiên, việc triển khai một thuật toán một cách chính xác là cần thiết nhưng chưa đủ: bạn không muốn làm giảm sức mạnh thuật toán của mình bằng cách triển khai không hiệu quả. Các trình biên dịch hiện đại cung cấp một mạng lưới an toàn nhất định về mặt này, đặc biệt là khi chúng được sử dụng trong chế độ tối ưu hóa mã của chúng. Thông thường, những cải tiến như vậy chỉ có thể tăng tốc chương trình bằng một hệ số không đổi, trong khi một thuật toán tốt hơn có thể tạo ra sự khác biệt trong quá trình chạy thời gian theo thứ tự của độ lớn. Nhưng một khi một thuật toán được chọn, việc tăng tốc 10–50% có thể đáng để bạn nỗ lực.

Trên thực tế, đây là một tin tốt vì nó làm cho kết quả cuối cùng thú vị hơn rất nhiều. (Vâng, tôi đã nghĩ đến việc đặt tên cho cuốn sách này là Niềm vui của các thuật toán.) Mặt khác, làm thế nào để người ta biết khi nào nên dừng lại? Trong thế giới thực, thường không phải lịch trình của dự án hoặc sự thiếu kiên nhẫn của sếp sẽ ngăn cản bạn. Và vì vậy nó phải là: sự hoàn hảo là đắt tiền và trên thực tế không phải lúc nào cũng được yêu cầu. Thiết kế một thuật toán là một hoạt động giống như kỹ thuật đòi hỏi sự thỏa hiệp giữa các mục tiêu cạnh tranh dưới sự ràng buộc của các nguồn lực sẵn có, với thời gian của nhà thiết kế là một trong những nguồn lực.