2.0 Các khái niệm cơ bản

Created by TUNG DUC NGUYEN tung2.nguyen, last modified on 2021/05/11

1. Thuật toán

Thuật toán là một hệ thống chặt chẽ và rõ ràng các quy tắc nhằm xác định một dãy thao tác trên cấu trúc dữ liệu sao cho: với một dữ liệu đầu vào, sau một số hữu hạn bước thực hiện các tao tác đã chỉ ra, ta đạt được mục tiêu nhất định.

Các đặc trưng của thuật toán

1.1 Tính đơn định

Ở mỗi bước của thuật toán, các thao tác phải hết sức rõ ràng, không nên gây sự nhập nhằng, lộn xộn, tùy tiện, đa nghĩa. Thực hiện đúng các bước của thuật toán với một dữ liệu đầu vào thì chỉ ra duy nhất một kết quả.

1.2 Tính dừng

Thuật toán không được rơi vào vô hạn, phải dừng lại và cho kết quả sau một số hữu hạn bước.

1.3 Tính đúng

Sau khi thực hiện tất cả các bước của thuật toán theo đúng trình tự đã định, ta phải được kết quả mong muốn với mọi bộ dữ liệu đầu vào. Kết quả đó được kiểm chứng bằng yêu cầu bài toán.

1.4 Tính phổ dụng

Thuật toán phải dễ sửa đổi để thích ứng với bất kỳ bài toàn nào trong một lớp các bài toán và có thể làm việc trên các dữ liệu khác nhau.

1.5 Tính khả thi

- Kích thước phải đủ nhỏ: Một thuật toán sẽ có tính hiệu quả bằng 0 nếu số lượng bộ nhớ mà nó yêu cầu vượt quá khả năng lưu trữ của hệ thống máy tính.
- Thuật toán phải được máy tính thực hiện trong thời gian cho phép.
- Phải dễ hiểu và dễ cài đặt.

2. Lập trình

Sau khi có thuật toán, ta phải tiến hành lập trình để thể hiện thuật toán đó. Muốn lập trình đạt hiệu quả cao, cần phải có kỹ thuật lập trình tốt. Kỹ thuật lập trình tốt thể hiện ở kỹ năng viết chương trình, khả năng gỡ rối và thao tác nhanh. Lập trình tốt không phải chỉ cần nắm vững ngôn ngữ lập trình là đủ, phải biết cách viết chương trình uyển chuyển, khôn khéo và phát triển dần các ý tưởng thành chương trình hoàn chỉnh. Kinh nghiệm cho thấy một thuật toán hay nhưng do cài đặt vụng về nên khi chạy lại cho kết quả sai hoặc tốc độ chậm.

Thông thường, ta không nên cụ thể hóa ngay toàn bộ chương trình mà nên tiến hành theo phương pháp tinh chế từng bước một:

1. Ban đầu, chương trình được thể hiện bằng ngôn ngữ tự nhiên, thể hiện thuật toán với các bước tổng thể, mỗi bước nêu lên một công việc phải được thực hiện.
2. Nếu một công việc đơn giản hoặc là một đoạn chương trình đã được học thuộc thi ta tiến hành viết mã ngay.
3. Nếu một công việc phức tạp thì ta lại chia công việc ra thành những công việc nhỏ hơn để tiếp tục với những công việc nhỏ hơn đó. Trong quá trình tinh chế từng bước, ta phải đưa ra những biểu diễn kiểu dữ liệu.

Như vậy cùng với sự tinh chế các công việc, dữ liệu cũng dần được tinh chế, có cấu trúc hơn, thể hiện rõ hơn mối liên hệ giữa những dữ liệu.

Phương pháp tinh chế từng bước là một thể hiện của tư duy giải quyết vấn đề từ trên xuống, giúp cho người lập trình có một định hướng thể hện trong phong cách viết chương trình. Tránh việc mò mẫm, xóa đi viết lại nhiều lần, biến chương trình thành tờ giấy nháp.

3. Kiểm thử

3.1 Chạy thử và tìm lỗi

Chương trình là do con người viết ra, mà đã là con người thì ai cũng có thể nhầm lẫn. Một khi chương trình viết xong thì chưa chắc đã chạy trên máy tính cho kết quả như kỳ vọng. Kỹ năng tìm lỗi, sửa lỗi, điều chỉnh lại chương trình cũng là một kỹ năng quan trọng của người lập trình. Kỹ năng này chỉ có được bằng kinh nghiệm tìm và sửa lỗi của chính mình. Có ba loại lỗi:

- Lỗi cú pháp: Lỗi này hay gặp và dễ sửa nhất. Chỉ cần nắm vững ngôn ngữ lập trình là đủ. Một người được coi là không biết lập trình nếu không biết sửa lỗi cú pháp.
- Lỗi cài đặt: Việc cài đặt không đúng thuật toán đã định, đối với lỗi này phải xem lại tổng thể chương trình, kết hợp với các chức năng gỡ rối để sửa lại cho đúng.
- Lỗi thuật toán: Lỗi này ít gặp nhưng nguy hiểm nhất, nếu nhẹ thì phải điều chỉnh lại thuật toán. Nếu nặng thì có khi phải loại bỏ hoàn toàn thuật toán và làm lại từ đầu.

3.2 Xây dựng các bộ test

Có nhiều chương trình rất khó kiểm tra tính đúng đắn. Nhất là khi ta không biết kết quả đúng là như thế nào. Vì vậy nếu chương trình vẫn chạy ra kế quả mà không biết đúng hay sai thì việc tìm lỗi rất khó khăn. Khi đó chúng ta nên tạo ra các bộ test cho chương trình của mình.

Các bộ test nên đặt trong file văn bản, vì việc tạo một file văn bản rất nhanh và mỗi lần chạy thử chỉ cần thay đổi tên file dữ liệu là xong. Kinh nghiệm với các bộ test là:

- Bắt đầu với các bộ test nhỏ, đơn giản, làm bằng tay cũng có thể có được đáp án để so sánh với kết quả chương trình.
- Tiếp theo vẫn là các bộ test nhỏ nhưng chưa các giá trị đặc biệt hoặc bất thường. Kinh nghiệm cho thấy đây là những bộ test dễ sai nhất.
- Các bộ test phải da dạng, tránh sự lặp đi lặp lại.
- Có một vài test lớn để kiểm tra tính chịu đựng của chương trình, kết quả đúng hay không chúng ta rất khó kiểm chứng được.

Lưu ý rằng việc chạy qua được hết các bộ test không có nghĩa là chương trình đã đúng. Vì vậy nếu có thể, chúng ta nên tìm cách chứng minh tính đúng đắn của thuật toán và chương trình khi thực hành. Tuyệt đối không nên làm điều này lúc thi vì điều này thường rất khó (tương đương với việc mất rất nhiều thời gian).

4. Tối ưu chương trình

Một chương trình chạy đúng không có nghĩa là việc lập trình đã xong, chúng ta phải sửa đổi lại vài chi tiết để chương trình có thể chạy nhanh hơn, hiệu quả hơn. Thông thường, trước khi kiểm thử thì ta nên đặt mục tiêu viết chương trình sao cho đơn giản, **miễn chạy đúng** là được, sau đó tối ưu chương trình. Xem lại chỗ nào viết chưa tốt thì tối ưu lại. Không nên viết tới đâu, tối ưu tới đó. Vì chương trình có mã tối ưu thường phức tạp và khó kiểm soát.

Việc tối ưu chương trình dựa vào các tiêu chuẩn sau:

4.1 Tính tin cậy

Chương trình phải chạy đúng như dự định, mô tả đúng một giải thuật đúng. Thông thương, tại mỗi bước ta nên kiểm tra lại tính đúng đắn của bước đó.

4.2 Tính uyển chuyển

Chương trình phải dễ sửa đổi. Bởi ít chương trình nào viết ra đã hoàn hảo ngay được mà vẫn cần sửa đổi. Chương trình viết dễ sửa đổi sẽ làm giảm bớt công sức của lập trình viên khi phát triển chương trình.

4.3 Tính trong sáng

Chương trình viết ra phải dễ đọc, để sau một thời gian dài, khi đọc lại chúng ta hiểu mình đã làm cái gì. Để nếu có điều kiện thì còn có thể sửa sai (nếu phát hiện ra lỗi), cải tiến hay biến đổi chương trình (để giải quyết bài toán khác). Tính trong sáng phụ thuộc rất nhiều vào công cụng lập trình và phong cách lập trình.

4.4 Tính hữu hiệu

Chương trình phải chạy nhanh và tốn ít bộ nhớ, tức là tiết kiệm được về cả không gian và thời gian. Để có một chương trình hữu hiệu, cần phải có giải thuật tốt và những kỹ thuật cài đặt (tiểu xảo). Tuy nhiên nếu áp dụng quá nhiều tiểu xảo có thể khiến chương trình trở nên rối rắm, khó hiểu. Tiêu chuẩn hữu hiệu nên dừng ở mức chấp nhận được, không quan trọng bằng ba tiêu chuẩn trên.

Từ những phân tích ở trên, chúng ta nhận thấy rằng việc làm ra một chương trình đòi hỏi rất nhiều công đoạn và tiêu tốn khá nhiều công sức. Chỉ một công đoạn không hợp lý sẽ làm tăng chi phí viết chương trình. Việc hiểu điều này cho chúng ta bài học kinh nghiệm:

Đừng bao giờ viết chương trình mà chưa xuy xét về giải thuật và những dữ liệu cần thao tác.

5. Phân tích thời gian thực hiện giải thuật

Refer CBL \\10.218.140.56\Video Training\01. SD\03.Studying\02.Algorithm\Analyzing running time of algorithms - Nguyen Duc Hung 20210407

**: Với bản thân mình việc áp dụng thời gian giải thuật 1s = 1 tỷ phép tính.

No labels