LỜI NÓI ĐẦU

"Alorithms + Data Structures = Programs"
N. Wirth

"Computing is an art form. Some programs are elegant,
some are exquisite, some are sparkling.
My claim is it is possible to write grand programs,
noble programs, truly magnifient programs"
D.E.Knuth

Cuốn sách này trình bày các vấn đề cơ bản, quan trọng nhất của Cấu trúc dữ liệu (CTDL) và thuật toán đã được đề xuất trong IEEE/ACM computing curricula, theo quan điểm hiện đại.

Khi thiết kế thuật toán để giải quyết một vấn đề, chúng ta cần phải sử dụng các đối tượng dữ liệu và các phép toán trên các đối tượng dữ liệu ở mức độ trừu tượng. Một trong các nội dung chính của sách này là nghiên cứu các kiểu dữ liệu trừu tượng (KDLTT) và các CTDL để cài đặt các KDLTT. KDLTT quan trọng nhất là tập động (một tập đối tượng dữ liệu với các phép toán tìm kiếm, xen, loại, …), KDLTT này được sử dụng rộng rãi nhất trong các chương trình ứng dụng. Các KDLTT cơ bản khác sẽ được nghiên cứu là : danh sách, ngăn xếp, hàng đợi, hàng ưu tiên, từ điển, …

Chúng ta sẽ cài đặt các KDLTT bởi các lớp C + +. Sự cài đặt các KDLTT bởi các lớp C + + cho phép ta có thể biểu diễn các đối tượng dữ liệu và các phép toán trên các đối tượng dữ liệu trong các chương trình ứng dụng một cách toán học, ngắn gọn và dễ hiểu, tương tự như khi ta sử dụng các số nguyên, số thực trong chương trình. Một ưu điểm quan trọng khác là, nó cho phép khi thiết kế và cài đặt phần mềm, chúng ta có thể làm việc ở mức độ quan niệm cao, có thể thực hành được các nguyên lý lập trình.

Với mỗi KDLTT, chúng ta sẽ nghiên cứu các cách cài đặt bởi các CTDL khác nhau. Hiệu quả của các phép toán trong mỗi cách cài đặt sẽ được đánh giá. Sự đánh giá so sánh các cách cài đặt sẽ giúp cho người sử dụng có sự lựa chọn thích hợp cho từng chương trình ứng dụng. Thông qua sự cài đặt các lớp C + + cho mỗi KDLTT và các chương trình ứng dụng chúng, độc giả sẽ được cung cấp thêm nhiều kỹ thuật lập trình hữu ích.

Sự nghiên cứu mỗi KDLTT sẽ được tiến hành qua các bước sau đây.

- Đặc tả KDLTT. Chúng ta sẽ mô tả các đối tượng dữ liệu bằng cách sử dụng các ký hiệu, các khái niệm toán học và logic. Các phép toán trên các đối tượng dữ liệu sẽ được mô tả bởi các hàm toán học.
- Lựa chọn CTDL thích hợp để cài đặt đối tượng dữ liệu
- Thiết kế và cài đặt lớp C + +.
- Phân tích hiệu quả của các phép toán.
- Các ví dụ ứng dụng.

Tổ chức sách

Nội dung của cuốn sách được tổ chức thành ba phần. Phần 1 sẽ nghiên cứu các CTDL cơ bản được sử dụng để cài đặt các KDLTT, đó là danh sách liên kết (DSLK), cây tìm kiếm nhị phân (TKNP), cây thứ tự bộ phận (heap), bảng băm. Danh sách, ngăn xếp, hàng đợi sẽ được cài đặt bởi mảng hoặc bởi DSLK. Cây TKNP được sử dụng để cài đặt tập động. Hàng ưu tiên được cài đặt hiệu quả bởi heap. Bảng băm là CTDL rất thích hợp để cài đặt từ điển.

Trong phần 2 chúng ta sẽ nghiên cứu các CTDL cao cấp. Các CTDL này có đặc điểm chung là sự tổ chức dữ liệu và các phép toán trên các CTDL này là khá phức tạp, song bù lại thời gian thực hiện các phép toán lại hiệu quả hơn. Chúng ta sẽ nghiên cứu các loại cây tìm kiếm cân bằng, các CTDL tự điều chỉnh, các CTDL đa chiều, … Đặc biệt, chúng ta sẽ đưa vào kỹ thuật phân tích trả góp, đây là kỹ thuật phân tích hoàn toàn mới, được sử dụng để đánh giá thời gian chạy của một dãy phép toán trên các CTDL tự điều chỉnh.

Phần 3 dành để nói về thuật toán. Chúng ta sẽ trình bày phương pháp đánh giá thời gian chạy của thuật toán bằng ký hiệu ô lớn, và các kỹ thuật để phân tích, đánh giá thời gian chạy của thuật toán. Một nội dung quan trọng của phần này là nghiên cứu các chiến lược thiết kế thuật toán. Chúng ta sẽ trình bày các chiến lược thiết kế thuật toán hay được sử dụng là : chia - để - trị, quy hoạch động, quay lui, … Các thuật toán sắp xếp, các thuật toán đồ thị cũng sẽ được nghiên cứu. Cuối cùng chúng ta trình bày một vấn đề có tính chất lý thuyết, đó là các bài toán NP – khó và NP - đầy đủ.

Sử dụng sách

Để đọc cuốn sách này, độc giả cần phải biết lập trình định hướng đối tượng với C + +. Tuy nhiên, chúng tôi đã đưa vào các chương 2 và 3 để trình bày một số vấn đề quan trọng liên quan tới thiết kế lớp C + +, giúp cho độc giả chưa biết C + + cũng có thể hiểu được các chương tiếp theo.

Nội dung của sách này đề cập tới nhiều vấn đề hơn là nội dung của giáo trình Cấu trúc dữ liệu và thuật toán cho sinh viên công nghệ thông tin. Theo quan điểm của chúng tôi, trong giáo trình Cấu trúc dữ liệu và thuật toán cho sinh viên công nghệ thông tin, chỉ nên đưa vào các chương 1, 4, 5, 6, 7, 8, 9 của phần I và các chương 15, 16, 17, 18 của phần II. Nếu sinh viên chưa được làm quen với sự đánh giá thời gian chạy của thuật toán, thì nội dung chương 15 cần được dạy trước.

Lời cảm ơn

Chúng tôi xin chân thành cảm ơn các đồng nghiệp ở bộ môn Khoa học máy tính, Khoa công nghệ thông tin, Trường Đại học Công nghệ, Đại học Quốc gia Hà Nội, vì những trao đổi bổ ích về các vấn đề được đề cập trong sách, đặc biệt TS. Phạm Hồng Thái, ThS Trần Quốc Long và ThS Ma Thị Châu đã cùng chúng tôi giảng dạy giáo trình Cấu trúc dữ liệu và thuật toán. Chúng tôi cũng xin chân thành cảm ơn Trường Đại học công nghệ, Đại học Quốc gia Hà Nội đã tạo điều kiện tốt nhất cho chúng tôi viết cuốn sách này.

Tháng Giêng, 2007 Đinh Mạnh Tường

MỤC LỤC

PHẦN I

CÁC CẤU TRÚC DỮ LIỆU CƠ BẢN

CHƯƠNG 1

SỰ TRỪU TƯỢNG HOÁ DỮ LIỆU

Khi thiết kế thuật giải cho một vấn đề, chúng ta cần sử dụng sự trừu tượng hoá dữ liệu. Sự trừu tượng hoá dữ liệu được hiểu là chúng ta chỉ quan tâm tới một tập các đối tượng dữ liệu (ở mức độ trừu tượng) và các phép toán (các hành động) có thể thực hiện được trên các đối tượng dữ liệu đó. Với mỗi phép toán chúng ta cũng chỉ quan tâm tới điều kiện có thể sử dụng nó và hiệu quả mà nó mang lại, không cần biết nó được thực hiện như thế nào. Sự trừu tượng hoá dữ liệu được thực hiện bằng cách tạo ra các kiểu dữ liệu trừu tượng. Trong chương này chúng ta sẽ trình bày khái niệm kiểu dữ liệu trừu tượng, các phương pháp đặc tả và cài đặt kiểu dữ liệu trừu tượng.

1.1 BIỂU DIỄN DỮ LIỆU TRONG CÁC NGÔN NGỮ LẬP TRÌNH

Trong khoa học máy tính, dữ liệu được hiểu là bất kỳ thông tin nào được xử lý bởi máy tính. Dữ liệu có thể là số nguyên, số thực, ký tự, … Dữ liệu có thể có cấu trúc phức tạp, gồm nhiều thành phần dữ liệu được liên kết với nhau theo một cách nào đó. Trong bộ nhớ của máy tính, mọi dữ liệu đều được biểu diễn dưới dạng nhị phân (một dãy các ký hiệu 0 và 1). Đó là dạng biểu diễn cụ thể nhất của dữ liệu (dạng biểu diễn vật lý của dữ liệu).

Trong các ngôn ngữ lập trình bậc cao (Pascal, C, C+ +…), dữ liệu được biểu diễn dưới dạng trừu tượng, tức là dạng biểu diễn của dữ liệu xuất phát từ dạng biểu diễn toán học của dữ liệu (sử dụng các khái niệm toán học, các mô hình toán học để biểu diễn dữ liệu). Chẳng hạn, nếu dữ liệu là các điểm trong mặt phẳng, thì chúng ta có thể biểu diễn nó như một cặp số thực (x, y), trong đó số thực x là hoành độ, còn số thực y là tung độ của điểm. Do đó, trong ngôn ngữ C + +, một điểm được biểu diễn bởi cấu trúc:

```
struct   point
{  double    x;
   double    y;
 };
```

Trong các ngôn ngữ lập trình bậc cao, các dữ liệu được phân thành các lớp dữ liệu (kiểu dữ liệu). Kiểu dữ liệu của một biến được xác định bởi một tập các giá trị mà biến đó có thể nhận và các phép toán có thể thực hiện trên các giá trị đó. Ví dụ, có lẽ kiểu dữ liệu đơn giản nhất và có trong nhiều ngôn ngữ lập trình là kiểu boolean, miền giá trị của kiểu này chỉ gồm hai giá trị false và true, các phép toán có thể thực hiện trên các giá trị này là các phép toán logic mà chúng ta đã quen biết.

Mỗi ngôn ngữ lập trình cung cấp cho chúng ta một số **kiểu dữ liệu cơ bản (basic data types).** Trong các ngôn ngữ lập trình khác nhau, các kiểu dữ liệu cơ bản có thể khác nhau. Ngôn ngữ lập trình Lisp chỉ có một kiểu cơ bản, đó là các S-biểu thức. Song trong nhiều ngôn ngữ lập trình khác (chẳng hạn Pascal, C / C + +, Ada, …), các kiểu dữ liệu cơ bản rất phong phú. Ví dụ, ngôn ngữ C + + có các kiểu dữ liệu cơ bản sau:

Các kiểu ký tự (char, signed char, unsigned char)

Các kiểu nguyên (int, short int, long int, unsigned)

Các kiểu thực (float, double, long double)

Các kiểu liệt kê (enum)

Kiểu boolean (bool)

Gọi là các kiểu dữ liệu cơ bản, vì các dữ liệu của các kiểu này sẽ được sử dụng như các thành phần cơ sở để kiến tạo nên các dữ liệu có cấu trúc phức tạp. Các kiểu dữ liệu đã cài đặt sẵn (build-in types) mà ngôn ngữ lập trình cung cấp là không đủ cho người sử dụng. Trong nhiều áp dụng, người lập trình cần phải tiến hành các thao tác trên các dữ liệu phức hợp. Vì vậy, mỗi ngôn ngữ lập trình cung cấp cho người sử dụng một số quy tắc cú pháp để tạo ra các kiểu dữ liệu mới từ các kiểu cơ bản hoặc các kiểu khác đã được xây dựng. Chẳng hạn, C + + cung cấp cho người lập trình các luật để xác

định các kiểu mới: kiểu mảng (array), kiểu cấu trúc (struct), kiểu con trỏ, …

Ví dụ. Từ các kiểu đã có $T_1, T_2, ..., T_n$ (có thể khác nhau), khai báo sau

```
struct   S {
    T1    M1 ;
    T2    M2 ;
    ……
    Tn    Mn ;
}
```

xác định một kiểu cấu trúc với tên là S, mỗi dữ liệu của kiểu này gồm n thành phần, thành phần thứ i có tên là M_i và có giá trị thuộc kiểu T_i (i = 1,…, n).

Các kiểu dữ liệu được tạo thành từ nhiều kiểu dữ liệu khác (các kiểu này có thể là kiểu cơ bản hoặc kiểu dữ liệu đã được xây dựng) được gọi là **kiểu dữ liệu có cấu trúc**. Các dữ liệu thuộc kiểu dữ liệu có cấu trúc được gọi là các **cấu trúc dữ liệu** (data structure). Ví dụ, các mảng, các cấu trúc, các danh sách liên kết, … là các cấu trúc dữ liệu (CTDL).

Từ các kiểu cơ bản, bằng cách sử dụng các qui tắc cú pháp kiến tạo các kiểu dữ liệu, người lập trình có thể xây dựng nên các kiểu dữ liệu mới thích hợp cho từng vấn đề. Các kiểu dữ liệu mà người lập trình xây dựng nên được gọi là các kiểu dữ liệu được xác định bởi người sử dụng (user-defined data types).

Như vậy, một CTDL là một dữ liệu phức hợp, gồm nhiều thành phần dữ liệu, mỗi thành phần hoặc là dữ liệu cơ sở (số nguyên, số thực, ký tự,…) hoặc là một CTDL đã được xây dựng. Các thành phần dữ liệu tạo nên một CTDL được liên kết với nhau theo một cách nào đó. Trong các ngôn ngữ lập trình thông dụng (Pascal, C/ C+ +), có ba phương pháp để liên kết các dữ liệu:

1. Liên kết các dữ liệu cùng kiểu tạo thành mảng dữ liệu.
2. Liên kết các dữ liệu (không nhất thiết cùng kiểu) tạo thành cấu trúc trong C/ C+ +, hoặc bản ghi trong Pascal.

3. Sử dụng con trỏ để liên kết dữ liệu. Chẳng hạn, sử dụng con trỏ chúng ta có thể tạo nên các danh sách liên kết, hoặc các CTDL để biểu diễn cây. (Chúng ta sẽ nghiên cứu các CTDL này trong các chương sau)

Ví dụ. Giả sử chúng ta cần xác định CTDL biểu diễn các lớp học. Giả sử mỗi lớp học cần được mô tả bởi các thông tin sau: tên lớp, số tổ của lớp, danh sách sinh viên của mỗi tổ; mỗi sinh viên được mô tả bởi 3 thuộc tính: tên sinh viên, tuổi và giới tính. Việc xây dựng một CTDL cho một đối tượng dữ liệu được tiến hành theo nguyên tắc sau: từ các dữ liệu có kiểu cơ sở tạo ra kiểu dữ liệu mới, rồi từ các kiểu dữ liệu đã xây dựng tạo ra kiểu dữ liệu phức tạp hơn, cho tới khi nhận được kiểu dữ liệu cho đối tượng dữ liệu mong muốn. Trong ví dụ trên, đầu tiên ta xác định cấu trúc Student

```
struct   Student
{
    string     StName;
    int        Age;
    bool       Sex;
}
```

Danh sách sinh viên của mỗi tổ có thể lưu trong mảng, hoặc biểu diễn bởi danh sách liên kết. Ở đây chúng ta dùng danh sách liên kết, mỗi tế bào của nó là cấu trúc sau:

```
struct     Cell
{
      Student       Infor;
      Cell*         Next;
}
```

Chúng ta sử dụng một mảng để biểu diễn các tổ, mỗi thành phần của mảng lưu con trỏ trỏ tới đầu một danh sách liên kết biểu diễn danh sách các sinh viên của một tổ. Giả sử mỗi lớp có nhiều nhất 10 tổ, kiểu mảng GroupArray được xác định như sau:

```
typedef     Cell*     GroupArray[10];
```

Cuối cùng, ta có thể biểu diễn lớp học bởi cấu trúc sau:

```
struct      StudentClass
{
    string      ClassName;
    int         GroupNumber;
    GroupArray  Group;
}
```

1.2 SỰ TRỪU TƯỢNG HOÁ DỮ LIỆU

Thiết kế và phát triển một chương trình để giải quyết một vấn đề là một quá trình phức tạp. Thông thường quá trình này cần phải qua các giai đoạn chính sau:

1. Đặc tả vấn đề.
2. Thiết kế thuật toán và cấu trúc dữ liệu.
3. Cài đặt (chuyển dịch thuật toán thành các câu lệnh trong một ngôn ngữ lập trình, chẳng hạn C+ +)
4. Thử nghiệm và sửa lỗi.

Liên quan tới nội dung của sách này, chúng ta chỉ đề cập tới hai giai đoạn đầu. Chúng ta muốn làm sang tỏ vai trò quan trọng của sự trừu tượng hoá (abstraction) trong đặc tả một vấn đề, đặc biệt là sự trừu tương hoá dữ liệu (data abstraction) trong thiết kế thuật toán.

Vấn đề được đặt ra bởi người sử dụng thường được phát biểu không rõ ràng, thiếu chính xác. Do đó, điều đầu tiên chúng ta phải làm là chính xác hoá vấn đề cần giải quyết, hay nói một cách khác là mô tả chính xác vấn đề. Điều đó được gọi là đặc tả vấn đề. Trong giai đoạn này, chúng ta phải trả lời chính xác các câu hỏi sau. Chúng ta được cho trước những gì? Chúng ta cần tìm những gì? Những cái đã biết và những cái cần tìm có quan hệ với nhau như thế nào? Như vậy, trong giai đoạn đặc tả, chúng ta cần mô tả chính xác các dữ liệu vào (inputs) và các dữ liệu ra (outputs) của chương trình. Toán học là một ngành khoa học trừu tượng, chính xác, các khái niệm toán học, các mô hình toán học là sự trừu tượng hoá từ thế giới hiện thực. Sử dụng trừu tượng hoá trong đặc tả một vấn đề đồng nghĩa với việc chúng ta sử dụng các khái niệm toán học, các mô hình toán học và logic để biểu diễn chính xác một vấn đề.

Ví dụ. Giả sử chúng ta cần viết chương trình lập lịch thi. Vấn đề như sau. Mỗi người dự thi đăng kí thi một số môn trong số các môn tổ chức thi. Chúng ta cần xếp lịch thi, mỗi ngày thi một số môn trong cùng một thời gian, sao cho mỗi người dự thi có thể thi tất cả các môn họ đã đăng kí.
Chúng ta có thể đặc tả inputs và outputs của chương trình như sau:

Inputs: danh sách các người dự thi, mỗi người dự thi được biểu diễn bởi danh sách các môn mà anh ta đăng kí.

Outputs: danh sách các ngày thi, mỗi ngày thi được biểu diễn bởi danh sách các môn thi trong ngày đó sao cho hai môn thi bất kì trong danh sách này không thuộc cùng một danh sách các môn đăng kí của một người dự thi.

Trong mô tả trên, chúng ta đã sử dụng khái niệm danh sách (khái niệm dãy trong toán học). Các khái niệm toán học, các mô hình toán học hoặc logic được sử dụng để mô tả các đối tượng dữ liệu tạo thành các **mô hình dữ liệu (data models).** Danh sách là một mô hình dữ liệu. Chú ý rằng, lịch thi cần thoả mãn đòi hỏi: người dự thi có thể thi tất cả các môn mà họ đăng kí. Để dễ dàng đưa ra thuật toán lập lịch, chúng ta sử dụng một mô hình dữ liệu khác: đồ thị. Mỗi môn tổ chức thi là một đỉnh của đồ thị. Hai đỉnh có cạnh nối, nếu có một người dự thi đăng kí thi cả hai môn ứng với hai đỉnh đó. Từ

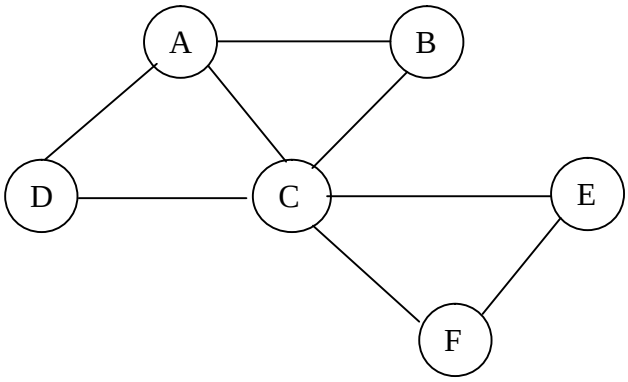
mô hình dữ liệu đồ thị này, chúng ta có thể đưa ra thuật toán lập lịch như sau:

Bước 1: Chọn một đỉnh bất kì, đưa môn thi ứng với đỉnh này vào danh sách các môn thi trong một ngày thi (danh sách này ban đầu rỗng). Đánh dấu đỉnh đã chọn và tất cả các đỉnh kề nó. Trong các đỉnh chưa đánh dấu, lại chọn một đỉnh bất kì và đưa môn thi ứng với đỉnh này vào danh sách các môn thi trong ngày thi. Lại đánh dấu đỉnh vừa chọn và các đỉnh kề nó. Tiếp tục quá trình trên cho tới khi tất cả các đỉnh của đồ thị được đánh dấu, chúng ta nhận được danh sách các môn thi trong một ngày thi.

Bước 2: Loại khỏi đồ thị tất cả các đỉnh đã xếp vào danh sách các môn thi trong một ngày thi ở bước 1 và loại tất cả các cạnh kề các đỉnh đó. Các đỉnh và các cạnh còn lại tạo thành đồ thị mới.

Bước 3: Lặp lại bước 1 và bước 2 cho tới khi đồ thị trở thành rỗng.

Chẳng hạn, giả sử các môn tổ chức thi là A, B, C, D, E, F và đồ thị xây dựng nên từ các dữ liệu vào được cho trong hình sau:



Khi đó lịch thi có thể như sau:

Ngày thi 1: A, F.

Ngày thi 2: D, E, B.

Ngày thi 3: C.

Sau khi đặc tả vấn đề, chúng ta chuyển sang giai đoạn thiết kế thuật toán để giải quyết vấn đề. Ở mức độ cao nhất của sự trừu tượng hoá, thuật toán được thiết kế như là **một dãy các hành động trên các đối tượng dữ**

liệu được thực hiện theo một trình tự logic nào đó. Thuật toán lập lịch thi ở trên là một ví dụ. Các đối tượng dữ liệu có thể là số nguyên, số thực, ký tự; có thể là các điểm trên mặt phẳng; có thể là các hình hình học; có thể là con người, có thể là danh sách các đối tượng (chẳng hạn, danh sách các môn thi trong ví dụ lập lịch thi); có thể là đồ thị, cây, …

Các hành động trên các đối tượng dữ liệu cũng rất đa dạng và tuỳ thuộc vào từng loại đối tượng dữ liệu. Chẳng hạn, nếu đối tượng dữ liệu là điểm trên mặt phẳng, thì các hành động có thể là: quay điểm đi một góc nào đó, tịnh tiến điểm theo một hướng, tính khoảng cách giữa hai điểm, … Khi đối tượng dữ liệu là danh sách, thì các hành động có thể là: loại một đối tượng khỏi danh sách, xen một đối tượng mới vào danh sách, tìm xem một đối tượng đã cho có trong danh sách hay không, …

Khi thiết kế thuật toán như là một dãy các hành động trên các đối tượng dữ liệu, chúng ta cần sử dụng **sự trừu tượng hoá dữ liệu (data abstraction).**

Sự trừu tượng hoá dữ liệu có nghĩa là chúng ta chỉ quan tâm tới một tập các đối tượng dữ liệu (ở mức độ trừu tượng) và các hành động (các phép toán) có thể thực hiện trên các đối tượng dữ liệu đó (với các điều kiện nào thì hành động có thể được thực hiện và sau khi thực hiện hành động cho kết quả gì), chúng ta không quan tâm tới các đối tượng dữ liệu đó được lưu trữ như thế nào trong bộ nhớ của máy tính, chúng ta không quan tâm tới các hành động được thực hiện như thế nào.

Sử dụng sự trừu tượng hoá dữ liệu trong thiết kế thuật toán là phương pháp luận thiết kế rất quan trọng. Nó có các ưu điểm sau:

- Đơn giản hoá quá trình thiết kế, giúp ta tránh được sự phức tạp liên quan tới biểu diễn cụ thể của dữ liệu .
- Chương trình sẽ có tính mođun (modularity). Chẳng hạn, một hành động trên đối tượng dữ liệu phức tạp được cài đặt thành một mođun (một hàm). Chương trình có tính mođun sẽ dễ đọc, dễ phát hiện lỗi, dễ sửa, …