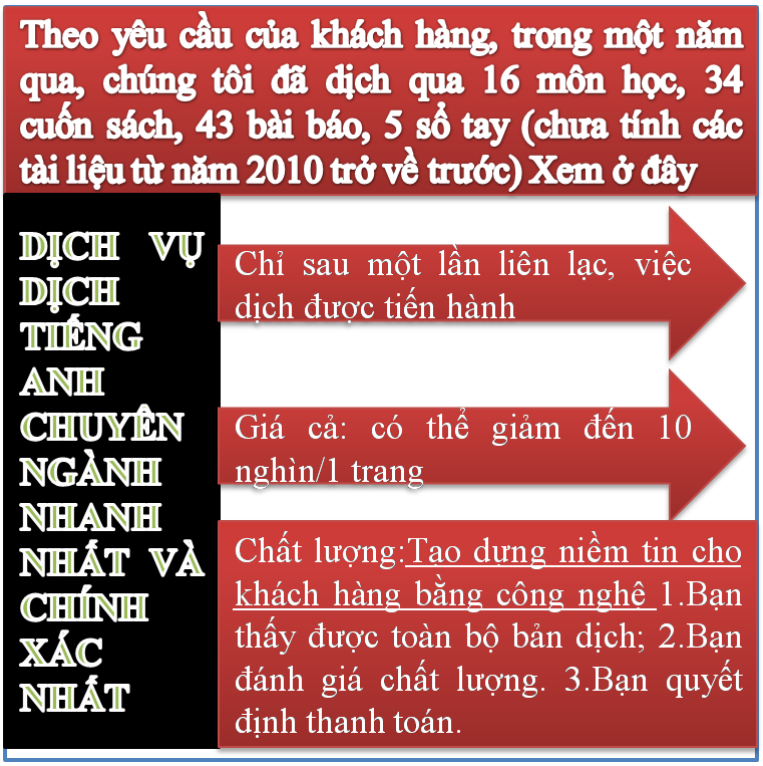

Tài liệu này được dịch sang tiếng việt bởi:

www.mientayvn.com

Tìm bản gốc tại thư mục này (copy link và dán hoặc nhấn Ctrl+Click):

https://drive.google.com/folderview?id=0B4rAPqlxIMRDSFE2RXQ2N3FtdDA&usp=sharing

Liên hệ để mua:

thanhlam1910_2006@yahoo.com hoặc frbwrthes@gmail.com hoặc số 0168 8557 403 (gặp Lâm)

Giá tiền: 1 nghìn /trang đơn (trang không chia cột); 500 VND/trang song ngữ

Dịch tài liệu của bạn: http://www.mientayvn.com/dich_tieng_anh_chuyen_nghanh.html

Chapter 2: Getting Started

Chương 2: Mở Đầu

This chapter will familiarize you with the framework we shall use throughout the book to think about the design and analysis of algorithms. It is self-contained, but it does include several references to material that will be introduced in Chapters 3 and 4. (It also contains several summations, which Appendix A shows how to solve.)

Chương này sẽ giúp bạn làm quen với các kiến thức nền tảng mà chúng ta sẽ sử dụng xuyên suốt sách này để thiết kế và phân tích giải thuật. Nó có tính chất độc lập, nhưng nó cũng đề cập đến một số tài liệu tham khảo cho những phần kiến thức được trình bày trong Chương 3 và 4. (Nó cũng chứa một số tổng, cách tính các tổng này được trình bày ở Phụ lục A.)

We begin by examining the insertion sort algorithm to solve the sorting problem introduced in Chapter 1. We define a "pseudocode" that should be familiar to readers who have done computer programming and use it to show how we shall specify our algorithms. Having specified the algorithm, we then argue that it correctly sorts and we analyze its running time. The analysis introduces a notation that focuses on how that time increases with the number of items to be sorted. Following our discussion of insertion sort, we introduce the divide-and-conquer approach to the design of algorithms and use it to develop an algorithm called merge sort. We end with an analysis of merge sort's running time.

Chúng ta bắt đầu bằng cách xét thuật toán sắp xếp kiểu chèn để giải bài toán sắp xếp được trình bày ở Chương 1. Chúng ta định nghĩa một ‘mã giả”, đây là một thuật ngữ khá quen thuộc với những người đọc đã qua khóa lập trình máy tính và dùng nó để chỉ rõ cách tiến hành thuật toán. Sau khi đã đưa ra thuật toán, chúng ta chỉ rõ nó sắp xếp chính xác và chúng ta phân tích thời gian chạy của nó. Chúng tôi sẽ giới thiệu một khái niệm ghi nhận sự tăng của thời gian theo số chi tiết được sắp xếp. Tiếp theo sau thảo luận về sắp xếp kiểu chèn, chúng tôi sẽ giới thiệu phương pháp chia để trị dùng cho thiết kế các giải thuật và dùng nó để xây dựng một thuật toán được gọi là sắp xếp kiểu trộn. Chúng ta kết thúc với mục phân tích thời gian chạy của thuật toán sắp xếp kiểu trộn.

2.1 Insertion sort

2.1 Sắp xếp kiểu chèn

Our first algorithm, insertion sort, solves the ***sorting problem*** introduced in Chapter 1:

Thuật toán đầu tiên của chúng ta, sắp xếp kiểu chèn, giải bài toán sắp xếp được trình bày ở Chương 1:

- **Input:** A sequence of n numbers $\langle a_1, a_2, \ldots, a_n \rangle$.
- Đầu vào: Một chuỗi n số $\langle a_1, a_2, \ldots, a_n \rangle$.
- **Output:** A permutation (reordering) $\langle a'_1, a'_2, \ldots, a'_n \rangle$ of the input sequence such that $a'_1 \le a'_2 \le \cdots \le a'_n$
- **Đầu ra:** Một hóan vị (sắp xếp lại) $\langle a'_1, a'_2, \ldots, a'_n \rangle$ chuỗi đầu vào sao cho $a'_1 \le a'_2 \le \cdots \le a'_n$

The numbers that we wish to sort are also known as the ***keys***.

Những số mà chúng ta muốn sắp xếp cũng được gọi là các khóa.

In this book, we shall typically describe algorithms as programs written in a ***pseudocode*** that is similar in many respects to C, Pascal, or Java. If you have been introduced to any of these languages, you should have little trouble reading our algorithms. What separates pseudocode from "real" code is that in pseudocode, we employ whatever expressive method is most clear and concise to specify a given algorithm. Sometimes, the clearest method is English, so do not be surprised if you come across an English phrase or sentence embedded within a section of "real" code. Another difference between pseudocode and real code is that pseudocode is not typically concerned with issues of software engineering. Issues of data abstraction, modularity, and error handling are often ignored in order to convey the essence of the algorithm more concisely.

Trong sách này, chúng ta thường mô tả các thuật toán dưới dạng các chương trình được viết bằng mã giả, một loại mã có nhiều khía cạnh giống với C, Pascal, hoặc Java. Nếu bạn đã được học bất kỳ ngôn ngữ nào trong số đó, bạn sẽ dễ dàng hiểu được các thuật toán của chúng tôi. Sự khác biệt giữa mã giả và mã “thực” nằm ở chổ trong mã giả, chúng ta sử dụng bất kỳ phương pháp biểu diễn nào rõ ràng và súc tích nhất để xác định một thuật toán nhất định. Đôi khi, phương pháp rõ ràng nhất là tiếng anh, vì vậy đừng ngạc nhiên kho bạn thấy một cụm từ hoặc câu tiếng anh được nhúng vào một phần của mã “thực”. Một sự khác biệt nữa giữa mã giả và mã thực là mã giả thường không liên quan đến các vấn đề kỹ thuật phần mềm. Nó thường bỏ qua những vấn đề như trừu tượng hóa dữ liệu, tính mô đun, và xử lý lỗi để truyền đạt nội dung của thuật toán cô đọng hơn.

We start with ***insertion sort***, which is an efficient algorithm for sorting a small number of elements. Insertion sort works the way many people sort a hand of playing cards. We start with an empty left hand and the cards face down on the table. We then remove one card at a time from the table and insert it into the correct position in the left hand. To find the correct position for a card, we compare it with each of the cards already in the hand, from right to left, as illustrated in Figure 2.1. At all times, the cards held in the left hand are sorted, and these cards were originally the top cards of the pile on the table. Figure 2.1: Sorting a hand of cards using insertion sort.

Chúng ta bắt đầu với thuật toán sắp xếp kiểu chèn, đây là một thuật toán có hiệu quả trong việc sắp xếp một số lượng nhỏ phần tử. Sắp xếp kiểu chèn hoạt động theo kiểu giống như người ta sắp xếp các lá bài. Đầu tiên, tay trái của chúng ta còn trống và bộ bài úp xuống bàn. Sau đó, chúng ta rút mỗi lần một lá khỏi bàn và đưa nó vào đúng vị trí ở trên tay trái. Để tìm vị trí chính xác cho một lá bài, chúng ta so sánh nó với mỗi lá bài trước đó trong tay, từ trái sang phải, như minh họa trong Hình 2.1. Vào mọi thời điểm, các lá bài được giữ trong tay được sắp xếp, và những lá bài này ban đầu là những lá bài nằm trên của chồng bài trên bàn.

Hình 2.1: Sắp xếp các lá bài dùng sắp xếp kiểu chèn

Our pseudocode for insertion sort is presented as a procedure called INSERTION-SORT, which takes as a parameter an array $A[1 \cdot\cdot n]$ containing a sequence of length n that is to be sorted. (In the code, the number n of elements in A is denoted by $length[A]$.) The input numbers are ***sorted in place***: the numbers are rearranged within the array A, with at most a constant number of them stored outside the array at any time. The input array A contains the sorted output sequence when INSERTION-SORT is finished.

Mã giả của chúng ta để sắp xếp kiểu chèn được trình bày dưới dạng một quy trình (thủ tục) được gọi là SẮP XẾP KIỂU CHÈN, nó chọn một tham số là một mảng $A[1 \cdot\cdot n]$ chứa một chuỗi có chiều dài n cần được sắp xếp. (Trong mã, số n phần tử trong A được kí hiệu bằng $length[A]$.) Các số đầu vào được sắp xếp đúng chổ: các số được sắp xếp lại trong mảng A, với nhiều nhất một lượng không đổi trong số chúng được lưu trữ bên ngoài mảng tại bất kỳ thời điểm nào. Mảng đầu vào A chứa chuỗi đầu ra được sắp xếp khi thuật toán SẮP XẾP KIỂU CHÈN HOÀN THÀNH.