**Chương 1: Các khái niệm căn bản về phân tích và thiết kế giải thuật**

* Các hệ thức truy hồi cơ bản
  + Hàm tính giai thừa có hệ thức truy hồi sau:
  + Bài toán tháp Hà Nội có hệ thức truy hồi sau cho số bước chuyển C(n):
  + Hàm tính số Fibonacci thứ N có hệ thức truy hồi sau:
* Phân tích độ phức tạp của giải thuật
  + Bảng các loại độ phức tạp tiệm cận tiêu biểu

|  |  |  |
| --- | --- | --- |
| **Loại** | **Tên** | **Chú thích** |
| 1 | Hằng số | Nếu trong giải thuật, tác vụ chính được thực thi chỉ một vài lần (với số lần cố định) thì thời gian tính toán là hằng số |
| lgN | Logarit  (lgN=log2N) | Giải thuật có độ phức tạp loại này tăng chậm hơn sự tăng của N |
| N | Tuyến tính | Khi thời gian chạy của giải thuật tỷ lệ tuyến tính với quy mô dữ liệu nhập |
| NlgN | Tuyến tính logarit | Nhiều giải thuật chia để trị, ví dụ Quicksort và xếp thứ tự bằng phương pháp trộn có độ phức tập trung bình rơi vào loại này |
| N2 | Bình phương | Khi giải thuật là vòng lặp lồng hai |
| N3 | Lập phương | Khi giải thuật là vòng lặp lồng ba |
| 2N | Lũy thừa | Ví dụ tiêu biểu là giải thuật sinh ra tất cả các tập con của một tập N phần tử |
| N! | Giai thừa | Ví dụ tiêu biểu là giải thuật sinh ra tất cả các hoán vị của một tập N phần tử |

* + Bảng các công thức tiêu biểu, nắm bắt ý tưởng chính của việc phân tích 1 giải thuật đệ quy

|  |  |  |
| --- | --- | --- |
| **Nhận diện** | **Hệ thức truy hồi** | **Các giải thuật có quan hệ truy hồi này** |
| Công thức 1: Một chương trình đệ quy có vòng lặp duyệt qua bộ dữ liệu nhập để loại đi một phần tử |  |  |
| Công thức 2: Một chương trình đệ quy tách đôi bộ dữ liệu nhập trong một bước làm việc | Giả sử  Ta có vậy |  |
| Công thức 3: Một chương trình đệ quy tách đôi bộ dữ liệu nhập trong một bước làm việc nhưng phải xem xét từng phần tử trong dữ liệu nhập | Giả sử      Suy ra  Ta có vậy |  |
| Công thức 4: Một chương trình đệ quy tách đôi dữ liệu nhập thành hai nửa trong một bước làm việc | Giả sử    Cuối cùng, khi i = n – 1, ta được:  Suy ra  Ta có vậy |  |

* Chuỗi số thông dụng
  + Chuỗi số cộng:
  + Chuỗi số nhân:
  + Chuỗi số điều hòa (Harmonic)
  + Chuỗi số thông dụng khi phân tích độ phức tạp của các thao tác trên cây nhị phân
  + Chuỗi lũy thừa đặc biệt
  + Chuỗi logarit
  + Công thức thông dụng
    - Hoán vị:
    - Tổ hợp:
    - Tập con: tập con của n phần tử = 2n

**Chương 2: Chiến lược chia để trị**

* Giải thuật Quicksort:
  + Trường hợp tốt nhất xảy ra với Quicksort khi mỗi lần phân hoạch tập tin được chia làm 2 phần bằng nhau, điều này làm cho số lần so sánh của Quicksort thỏa mãn hệ thức truy hồi **C(N) = 2C(N/2) + N + 1**. Do đó độ phức tạp trong trường hợp tốt nhất của Quicksort là **NlgN**
  + Trường hợp xấu nhất của Quicksort là khi tập tin đã có thứ tự, tổng số lần so sánh sẽ là **(n2 + 3n)/2**. Độ phức tạp trong trường hợp xấu nhất của Quicksort là **O(n2)**
  + Trong trường hợp trung bình hệ thức truy hồi là . Độ phức tạp trong trường hợp trung bình là **C(N) = 2NlnN**
* Giải thuật Mergesort
  + Sắp thứ tự bằng phương pháp trộn cần khoảng **NlgN** lần so sánh để sắp bất kỳ tập tin N phần tử nào
  + Đối với giải thuật sắp thứ tự bằng phương pháp trộn (đệ quy), số lần so sánh được mô tả bằng hệ thức truy hồi . Vậy
  + Kết quả phân tích trên đúng trong cả 2 trường hợp xấu nhất và trung bình
* Sắp thứ tự ngoại
  + Gọi **br** là tổng số khối của tập tin đầu vào, **M** là số trang (page) của bộ đệm trong bộ nhớ chính. Ta được:
    - Tổng truy đạt khối trong giai đoạn tạo run = **2br**
    - Tổng số run ban đầu =
    - Tổng số chuyến trong giai đoạn trộn run =
    - Chi phí tất cả các chuyến trộn =
    - Tổng số truy đạt khối cho toàn giải thuật =
* Cây tìm kiếm nhị phân
  + Gọi N là số nút nội của cây nhị phân và X là số nút ngoại của cây nhị phân, ta có thể thấy rằng **X = N + 1**
  + Giải thuật tính chiều cao của cây nhị phân (so sánh để kiểm tra xem một cây con có rỗng hay không là tác vụ căn bản), giải thuật có tổng số lần so sánh để kiểm tra xem một cây con có rỗng hay không sẽ bằng tổng số nút mà cây mở rộng có, tức là: . Vậy độ phức tạp của giải thuật tính chiều cao cây nhị phân là **O(N)**
  + Giải thuật duyệt cây tìm kiếm nhị phân bằng thứ tự nội (inorder-tree-walk) có độ phức tạp là **O(N)** với N là số nút trong cây
  + Một tác vụ thêm vào hay tìm kiếm trên một cây nhị phân đòi hỏi chừng **2lnN** lần so sánh trên một cây được tạo ra từ N trị khóa ngẫu nhiên (N nút)
  + Chiều dài trung bình của cây N nút là **.** Vậy chiều dài trung bình của một nút trong cây là **2lnN**
  + Trường hợp xấu nhất xảy ra khi cây nhị phân bị suy biến thành một danh sách liên kết. Độ phức tạp của thao tác tìm kiếm trên cây tìm kiếm nhị phân trong trường hợp xấu nhất là O(N) (cần N lần so sánh cho cây tìm kiếm có N nút)

**Chương 3: Chiến lược giảm để trị**

* Sắp thứ tự bằng phương pháp chèn (Insertion sort)
  + Trường hợp xấu nhất xảy ra khi mảng đã có thứ tự đảo ngược, vòng lặp ngoài của giải thuật được thực thi N – 1 lần và vòng lặp trong được thực thi với tổng số lần **,** số bước dịch chuyển cũng như số lần so sánh trong trường hợp xấu nhất đều xấp xỉ
  + Trường hợp trung bình ta có khoảng i/2 lần so sánh được thực thi ở vòng lặp trong, vì vậy tổng số lần so sánh là
  + Sắp thứ tự bằng phương pháp chèn thực thi khoảng N2/2 so sánh và N2/4 hoán vị trong trường hợp xấu nhất
  + Sắp thứ tự bằng phương pháp chèn thực thi khoảng N2/4 so sánh và N2/8 hoán vị trong trường hợp trung bình
  + Sắp thứ tự bằng phương pháp chèn có độ phức tạp tuyến tính đối với một mảng đã gần có thứ tự
* Các giải thuật duyệt đồ thị
  + Có 2 cách biểu diễn đồ thị: dùng ma trận kế cận, dùng tập danh sách kế cận
  + Đồ thị vô hướng:
    - Khi biểu diễn ở dạng ma trận kề thì mỗi cạnh tương ứng với 2 bit trong ma trận (mỗi cạnh nối giữa x và y được biểu diễn bằng giá trị true tại cả a[x,y] và a[y,x]), do đó đường chéo chính của ma trận kề gồm toàn bit 1. Việc kiểm tra xem tồn tại một cạnh giữa hai đỉnh u và v có độ phức tạp thời gian là O(V)
    - Khi biểu diễn ở dạng tập danh sách kế cận thì mỗi cạnh trong đồ thị tương ứng với hai nút trong tập danh sách kế cận, do đó số nút trong tập danh sách kế cận bằng 2E (E là số cạnh của đồ thị) . Việc kiểm tra xem tồn tại một cạnh giữa hai đỉnh u và v có độ phức tạp thời gian là O(1)
    - Duyệt đồ thị theo chiều sâu trước (depth-first-traversal): sử dụng cấu trúc dữ liệu chồng (stack)
    - Duyệt đồ thị theo chiều rộng trước (breadth-first-traversal): sử dụng cấu trúc dữ liệu hàng đợi (queue)
    - Duyệt đồ thị theo chiều sâu trước hay chiều rộng trước có độ phức tạp tính toán của giải thuật là như nhau:
      * Khi dùng tập danh sách kế cận độ phức tạp là O(V + E)
      * Khi dùng ma trận kề độ phức tạp là O(V2)
  + Đồ thị có hướng
    - Khi biểu diễn ở dạng ma trận kề thì mỗi cạnh tương ứng với 1 bit trong ma trận (mỗi cạnh nối giữa x và y được biểu diễn bằng giá trị true tại a[x,y]), do đó ma trận này không có tính đối xứng qua đường chéo chính.
    - Khi biểu diễn ở dạng tập danh sách kế cận thì mỗi cạnh trong đồ thị tương ứng với một nút trong tập danh sách kế cận, do đó số nút trong tập danh sách kế cận bằng E (E là số cạnh của đồ thị)
    - Giải thuật sắp xếp thứ tự tôpô phương pháp 1 (duyệt theo chiều sâu trước): Nhận diện các đỉnh không có đỉnh đi trước (tức các đỉnh nguồn), đưa chúng vào stack; **while** stack khác rỗng **do { if** phần tử đang ở đỉnh stack là 1 đỉnh có một số đỉnh kề cận **then** đưa tất cả những đỉnh kế cận này vào stack **else** lấy phần tử đang ở đỉnh stack ra khỏi stack, gỡ đỉnh tương ứng ra khỏi đồ thị và đưa vào danh sách kết quả }; Đảo ngược danh sách kết quả để được thứ tự tôpô
    - Giải thuật sắp xếp thứ tự tôpô phương pháp 2 (duyệt theo chiều rộng trước): Nhận diện những đỉnh không có đỉnh đi trước, đưa chúng vào trong hàng đợi; **while** hàng đợi khác rỗng **do {** lấy đỉnh N ở đầu hàng đợi ra; **for** mỗi đỉnh M kế cận với đỉnh N **do {** xóa bỏ cạnh nối từ N đến M; **if** nếu đỉnh M không có đỉnh đi trước **then** đưa đỉnh M vào đuôi hàng đợi **} }**
    - Độ phức tạp tính toán của giải thuật sắp xếp tôpô theo phương pháp 1 và 2 khi dùng tập danh sách kế cận là **O(E + V)**