ĐẠI HỌC QUỐC GIA TPHCM
Trường Đại học Công nghệ Thông tin
Khoa Khoa học Máy tính
Lớp Khoa học Tài năng khóa 2021

BÀI TẬP PHÂN TÍCH VÀ THIẾT KẾ THUẬT TOÁN KHÔNG ĐỆ QUY

Ngày 27 tháng 3 năm 2023

Thành viên 1: 21520117 - Phan Trường Trí
Thành viên 2: 21520373 - Nguyễn Đức Nhân
Lớp: CS112.N21.KHTN
Giảng viên: Nguyễn Thanh Sơn

Mục lục

1 Bài 1:

1.1 Đề bài:

The range of a finite nonempty set of n real numbers S is defined as the difference between the largest and smallest elements of S. For each representation of S given below, describe in English an algorithm to compute the range. Indicate the time efficiency classes of these algorithms using the most appropriate notation (O, θ, Ω)

a. An unsorted array

b. A sorted array

c. A sorted singly linked list

d. A binary search tree

1.2 Bài giải:

Yêu cầu của bài toán có thể được giải quyết bằng việc tìm hiệu của phần tử lớn nhất và phần tử nhỏ nhất.

a. An unsorted array

- **Thuật toán:** Thực hiện duyệt 1 vòng lặp để duyệt qua tất cả các phần tử của mảng để tìm phần tử lớn nhất và nhỏ nhất. Sau đó lấy phần tử lớn nhất trừ cho phần tử nhỏ nhất.
- **Độ phức tạp:** Thuật toán có độ phức tạp ở cả best cases, worst cases, average cases là như nhau: $O(n), \theta(n), \Omega(n)$

b. A sorted array

- **Thuật toán:** Với mảng sắp xếp giảm dần thì lấy phần tử đầu tiên trừ cho phần tử cuối cùng của mảng và thực hiện ngược lại với mảng sắp xếp tăng dần.

- **Độ phức tạp:** Thuật toán có độ phức tạp ở cả best cases, worst cases, average cases là như nhau: $O(1), \theta(1), \Omega(1)$

c. A sorted singly linked list

- **Thuật toán:** Với mảng sắp xếp giảm dần thì lấy phần tử đầu tiên trừ cho phần tử cuối cùng của mảng và thực hiện ngược lại với mảng sắp xếp tăng dần.
- **Độ phức tạp:** Vì đầu vào là 1 linked list nên ta cần duyệt qua toàn bộ danh sách liên kết để tìm thấy phần tử cuối. Thuật toán có độ phức tạp ở cả best cases, worst cases, average cases là như nhau: $O(n), \theta(n), \Omega(n)$

d. A binary search tree

- **Thuật toán:** Dựa trên tính chất của cây tìm kiếm nhị phân, ta sẽ lấy giá trị của node nằm ngoài cùng bên phải trừ cho giá trị của node nằm ngoài cùng bên trái.
- **Độ phức tạp:**
  - **Worst case (O):** Cây nhị phân chỉ có 1 nhánh duy nhất về bên trái hoặc bên phải. Độ phức tạp để tìm phần tử ngoài cùng bên trái và ngoài cùng bên phải là $O(n)$
  - **Average case (θ)** Cây nhị phân có độ cao $h = log_2(N + 1)$. Độ phức tạp để tìm phần tử ngoài cùng bên trái và ngoài cùng bên phải là $\theta(h)$
  - **Worst case (Ω):** Cây nhị phân là cây nhị phân cân bằng với độ cao là $h = log_2(n)$. Độ phức tạp để tìm phần tử ngoài cùng bên trái và ngoài cùng bên phải là $\Omega(log(n))$

2 Bài 2:

2.1 Đề bài:

Lighter or heavier? You have $n > 2$ identical-looking coins and a two-pan balance scale with no weights. One of the coins is a fake, but you do not know whether it is lighter or heavier than the genuine coins, which all weigh the same. Design a $\theta(1)$ algorithm to determine whether the fake coin is lighter or heavier than the others.

2.2 Bài giải:

- **Giải thuật:** Ta chia n đồng xu thành 3 nhóm có số lượng bằng nhau và đặt phần dư thành nhóm thứ 4 nếu có.
  - **Bước 1:** Ta thực hiện tối đa 2 lần cân với mỗi lần cân khác nhau đôi một với 3 nhóm đầu tiên. Sau bước này ta sẽ biết đồng xu có nằm trong 3 nhóm đầu tiên hay không?
  - **Bước 2:** Trong trường hợp đồng xu giả nằm trong nhóm 4 thì ta sẽ lấy thêm k đồng xu ở 3 nhóm đầu để thêm vào nhóm 4 sao cho số đồng xu của nhóm 4 bằng 3. Ta lại thực hiện **Bước 1** để tìm ra đồng xu giả trong nhóm 4.

  Kết quả của bài toán sẽ là kết quả của các phép cân được thực hiện trong 2 bước trên.
- **Độ phức tạp:** Dễ thấy ta cần tối đa 2 bước để thực hiện **Bước 1** và tối đa 2 bước để thực hiện **Bước 2**. Vậy độ phức tạp của thuật toán này là $\theta(1)$

3 Bài 3:

3.1 Đề bài:

ALGORITHM $GE(A[0..n-1, 0..n])$

```
//Input: An n × (n + 1) matrix A[0..n − 1, 0..n] of real numbers
for i ← 0 to n − 2 do
    for j ← i + 1 to n − 1 do
        for k ← i to n do
            A[j, k] ← A[j, k] − A[i, k] ∗ A[j, i] / A[i, i]
```

a. Find the time efficiency class of this algorithm.

b. What glaring inefficiency does this pseudocode contain and how can it be eliminated to speed the algorithm up?

3.2 Bài giải:

a. Find the time efficiency class of this algorithm

Xét độ phức tạp của từng vòng for:

- Vòng for biến i: $O(n)$
- Vòng for biến j: $O(n)$
- Vòng for biến k: $O(n)$

Vậy độ phức tạp của thuật toán trên là $O(n^3)$

b. What glaring inefficiency does this pseudocode contain and how can it be eliminated to speed the algorithm up?

Xét giá trị của $A[j, i]$:

- Khi $k = i$:
  $A[j, k] = A[j, k] - A[i, k] * A[j, i]/A[i, i]$
  $\Leftrightarrow$A[j,i] = A[j,i] - A[i,i]*A[j,i]/A[i,i]

$\Leftrightarrow$A[j,i] = A[j,i] - A[j,i]
$\Leftrightarrow$A[j,i] = 0

- Khi $k > i$:
  $A[j,k] = A[j,k] - A[i,k] * A[j,i]/A[i,i]$
  $\Leftrightarrow$A[j,k] = A[j,k] - A[i,k]*0/A[i,i]**(Vì** $A[j,i] = 0$ **khi** $k = i$**)**
  $\Leftrightarrow$A[j,k] = A[j,k] - 0
  $\Leftrightarrow$A[j,k] = A[j,k]
  Do đó ta có thể cải tiến giải thuật này thành:

```
Function ALGORITHM(A[0...n − 1, 0...n):
    for i ← 0 to n − 2 do
        for j ← i + 1 to n − 1 do
            A[j, i] ←0 end
        end
```

Độ phức tạp của thuật toán giảm còn $O(n^2)$.