Phân Tích Đánh Giá Thuật Giải

Mục tiêu

Bài tập này nhằm mục đích rèn luyện cho sinh viên khả năng phân tích và đánh giá độ phức tạp của thuật giải thông qua các phương pháp toán học như tính toán Big-O, Small-o, Big-Ω, Big-Θ. Nắm được các kĩ thuật này, sinh viên sẽ có khả năng tư duy và tạo ra được các thuật giải với độ phức tạp thấp hoặc có khả năng xử lý hiệu quả đối với các bài toán thực tế.

Lý thuyết

Các qui tắc xác định độ phức tạp tính toán của thuật giải:

- Gọi $T_1(n)$ và $T_2(n)$ là thời gian thực hiện của 2 đoạn chương trình P_1 và P_2 mà $T_1(n)$ là $O(f(n))$ và $T_2(n)$ là $O(g(n))$

  Qui tắc cộng: Khi P_1 và P_2 là 2 đoạn chương trình tiếp nối, nghĩa là P_1 thực hiện xong rồi mới tới P_2 thì ta có tổng thời gian thực hiện của 2 đoạn chương trình P_1 và P_2 là:

  $$T_1(n) + T_2(n) = O(\max(f(n), g(n)))$$

  Qui tắc nhân: Khi P_1 và P_2 là 2 đoạn chương trình lồng nhau, nghĩa là P_1 nằm trong P_2 hoặc P_2 nằm trong P_1 thì ta có tổng thời gian thực hiện của toàn bộ chương trình chứa P_1 và P_2 là:

  $$T_1(n) * T_2(n) = O(f(n) * g(n))$$

- Các phép toán sơ cấp: so sánh, gán, đọc, viết, gọi hàm, return, … có thời gian thực hiện $O(1)$
- Đối với lệnh lựa chọn hay còn gọi là câu điều kiện if – else có dạng

  if (<điều kiện>)

  lệnh 1

  else

  lệnh 2

  thì cách đánh giá độ phức tạp thuật toán hay thời gian thực hiện giải thuật như sau:

  Gọi thời gian đánh giá điều kiện là $T_0(n)$ (trong một điều kiện có thể có nhiều phép so sánh).

  Gọi thời gian thực hiện lệnh 1 là $T_1(n)$, thời gian thực hiện lệnh 2 là $T_2(n)$.

  Thời gian thực hiện cấu trúc điều khiển if-else sẽ là thời gian lớn nhất trong các thời gian $T_0(n) + T_1(n)$ và $T_0(n) + T_2(n)$ tức là $\max(T_0(n) + T_1(n), T_0(n) + T_2(n))$

- Câu lệnh switch được đánh giá tương tự như câu lệnh if-else
- Đánh giá các lệnh lặp: for, while, do-while
  - Gọi số lần lặp tối đa là L(n)
  - Xét i = 1, 2,…,L(n) ta đánh giá thời gian chạy của mỗi lần lặp thứ i là $T_i(n)$
  - Trong mỗi lần lặp, luôn có chi phí kiểm tra điều kiện lặp gọi là $T_0(n)$
  - Như vậy, chi phí hay thời gian thực hiện của một lệnh lặp là:

$$\sum_{i=1}^{L(n)} (T_0(n) + T_i(n))$$

Nội dung

Câu 1 Tính Big-O của các hàm số sau:

1/ $f(n) = 5n^4 + 3n^3 + 2n^2 + 4n + 1$

Đáp án:

Ta có $f(n) = 5n^4 + 3n^3 + 2n^2 + 4n + 1 \leq (5 + 3 + 2 + 4 + 1)n^4 = 15n^4$, $\forall n \geq 1$.

Vậy Big-O của f(n) là $O(n^4)$

2/ $f(n) = 5n^2 + 3n\log n + 2n + 5$ (ghi chú: logn = log2n)

Đáp án:

Vì logn << n (nhỏ hơn rất nhiều) nên ta có:

$f(n) = 5n^2 + 3n\log n + 2n + 5 \leq (5 + 3 + 2 + 5)n^2$, $\forall n \geq 1$.

Vậy Big-O của f(n) là $O(n^2)$

3/ $f(n) = 20n^3 + 10n\log n + 5$

Đáp án:

Vì logn << n nên ta có:

$f(n) = 20n^3 + 10n\log n + 5 \leq (20 + 10 + 5)n^3$, $\forall n \geq 1$.

Vậy Big-O của f(n) là $O(n^3)$

4/ f(n) = 3logn + 2

Đáp án:

Ta có f(n) = 3logn + 2 ≤ (3 + 2)logn, ∀n ≥ 1.

Vậy Big-O của f(n) là O(logn)

5/ f(n) = 2n+2

Đáp án:

Ta có f(n) = 2n+2 ≤ (2 + 2)n, ∀n ≥ 1.

Vậy Big-O của f(n) là O(n)

6/ f(n) = 2n + 100logn

Đáp án:

Vì logn << n nên ta có:

f(n) = 2n + 100logn ≤ (2 + 100)n, ∀n ≥ 1.

Vậy Big-O của f(n) là O(n)

7/ f(n) = 2n

Đáp án:

Ta có: f(n) = 2n ≤ 2n, ∀n ≥ 1.

Vậy Big-O của f(n) là O(n)

8/ f(n) = 10

Đáp án:

f(n) = 10 ≤ 10.n^0, ∀n

Vậy Big-O của f(n) là O(n^0) tức là O(1)

Câu 2 Tính độ phức tạp của thuật giải sau:

```
1    sum = 0;
2    for (int i = 0; i < n; i++){
3        System.out.println(sum);
4        sum = sum + i;
5    }
```

Đáp án:

Thời gian chạy của các câu lệnh:

+ Dòng 1 có 1 phép gán

+ Dòng 2 có 1 vòng lặp với số lần lặp tối đa là L(n) = n, với mỗi lần lặp:

- Có 1 phép so sánh kiểm tra điều kiện lặp
- Có 1 phép xuất
- Có 1 phép gán

Vậy $f(n) = 1 + 3n \leq (1 + 3)n$ với $n \geq 1$

Độ phức tạp của f(n) là O(n).

Câu 3 Tính độ phức tạp của thuật giải sau:

```
1    Bước 1: Gán tổng = 0. Gán i = 0.
2    Bước 2:
3        ▪ Tăng i thêm 1 đơn vị
4        ▪ Gán tổng = tổng + i
5    Bước 3: so sánh i với 10
6        ▪ Nếu i < 10, quay lại bước 2
7        ▪ Ngược lại, nếu i ≥ 10, dừng thuật toán
```

Gợi ý: số phép gán của thuật toán là bao nhiêu ? Số phép so sánh là bao nhiêu ?

Đáp án:

+ Dòng 1 có 2 phép gán

+ Dòng 3-4 có 2 phép gán

+ Dòng 5 có 1 phép so sánh

+ Dòng 6 là một điều kiện lặp trong trường hợp i < 10, số bước lặp tối đa là L(n) = 10, mỗi bước lặp thực hiện số phép gán của dòng 3 và 4, số phép so sánh của dòng 5

+ Dòng 7 là một phép toán trả về (dừng thuật toán) trong trường hợp i ≥ 10

Vậy số phép gán là 2 + 2 * 10 = 22

Số phép so sánh là 10

Câu 4 Tính độ phức tạp của thuật toán tính tổng dãy số sau:

$$S = 1 + \frac{1}{2} + \frac{1}{6} + \ldots + \frac{1}{n!}$$

Đáp án:

Dãy số trên có thể viết lại dưới dạng:

$$S = \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \ldots + \frac{1}{n!}$$

Do đó thuật giải tìm S có thể được lập trình như sau:

```
1    int giaiThua = 1;
2    double sum = 0;
3    for (int i=1; i<=n; i++) {
4        giaiThua = giaiThua * i;
5        sum = sum + (1.0 / giaiThua);
6    }
```

Ta có: f(n) = 2 + 3n ≤ (2 + 3)n với n ≥ 1.

Vậy độ phức tạp của f(n) là O(n)

Câu 5 Cho mảng A có chứa các số nguyên, có n phần tử, giải thuật sau xác định mảng A có chứa số nguyên x hay không

```
1    i = 0;
2    while (i < n && A[i] != x)
3        i++;
```

Tính độ phức tạp của giải thuật trên

Đáp án:

+ Dòng 1 có 1 phép gán

+ Dòng 2 có 2 phép so sánh

+ Dòng 3 có 1 phép toán

Dòng 2-3 lặp tối đa L(n) = n lần trong trường hợp không tìm được (A[i] == x)

Vậy f(n) = 1 + (2 + 1)n ≤ (1 + 2 + 1)n, với n≥ 1.

Nên độ phức tạp của f(n) là O(n)

Câu 6 Cho biết độ phức tạp của các thuật toán sau:

Thuật toán 1:

```
1    for (i=0; i<n; i++)
2       for (j=0; j<n; j++)
3          k++;
```

Đáp án:

+ Dòng 1 thực hiện 1 vòng lặp tối đa $L_1 = n$ (bước)

+ Dòng 2 thực hiện 1 vòng lặp tối đa $L_2 = n$ (bước)

+ Dòng 3 thực hiện 1 phép toán

Vòng lặp 1 và 2 lồng nhau nên theo nguyên lý nhân ta có:

$f(n) = L_1 * L_2 * 1 = n * n * 1 = n^2$

Vậy độ phức tạp của f(n) là $O(n^2)$

Thuật toán 2:

```
1    for (i=0; i<n; i++)
2       k++;
3    for (i=0; i<n; i++)
4       for (j=0; j<n; j++)
5          k++;
```

Đáp án:

+ Dòng 1 thực hiện 1 vòng lặp tối đa $L_1 = n$ (bước)

+ Dòng 2 thực hiện 1 phép toán

+ Dòng 3 thực hiện 1 vòng lặp tối đa $L_2 = n$ (bước)

+ Dòng 4 thực hiện 1 vòng lặp tối đa $L_3 = n$ (bước)

+ Dòng 5 thực hiện 1 phép toán

Dòng 1-2 và dòng 3-5 là 2 đoạn chương trình nối tiếp nên theo nguyên lý cộng ta có:

$f(n) = \max(L_1 * 1, L_2 * L_3 * 1) = \max(n, n * n) = n^2$

Vậy độ phức tạp của f(n) là $O(n^2)$

Thuật toán 3:

```
1    for (int i=0; i<n-1; i++)
2       for (int j=0; j<i; j++)
3          k += 1;
```

Đáp án:

+ Dòng 1 thực hiện 1 vòng lặp tối đa $L_1 = n-1$ (bước)

+ Dòng 2 thực hiện 1 vòng lặp tối đa $L_2 = i$ (bước)

+ Dòng 3 thực hiện 1 phép toán

Xét i = 0: $L_2 = 0$

i = 1: $L_2 = 1$

i = 2: $L_2 = 2$

…

i = n - 1: $L_2 = n - 1$

Vậy $f(n) = 0 + 1 + 2 + \ldots + n - 1 = \frac{(n-1)n}{2} = \frac{n^2-n}{2} \leq \frac{n^2}{2}$

Vậy độ phức tạp của f(n) là $O(n^2)$

Câu 7 Cho biết độ phức tạp của thuật toán sau:

```
1    int MaxSubSum1(const int a[], int n) {
2       int maxSum=0;
3       for (int i=0; i<n; i++)
4          for (int j=i; j<n; j++) {
5             int thisSum=0;
6             for (int k=i; k<=j; k++)    thisSum+=a[k];
7             if (thisSum>maxSum) maxSum=thisSum;
8          }
9       return maxSum;
10   }
```

Đáp án:

+ Dòng 2 thực hiện 1 phép gán

+ Dòng 3 thực hiện 1 vòng lặp tối đa $L_1 = n$ (bước)

+ Dòng 4 thực hiện 1 vòng lặp tối đa $L_2 = n - i$ (bước)

+ Dòng 5 thực hiện 1 phép gán

+ Dòng 6 thực hiện 1 vòng lặp tối đa $L_3 = j - i + 1$ (bước) và 1 phép gán

+ Dòng 7 thực hiện 1 phép so sánh, 1 phép gán

+ Dòng 9 thực hiện 1 phép trả về

Vòng lặp dòng 3, 4 và 6 lồng nhau nên theo nguyên lý nhân ta có:

$f(n) = 1 + \sum_{i=0}^{n-1} \sum_{j=i}^{n-1} (3 + \sum_{k=i}^{j} 1)$

$= 1 + \sum_{i=0}^{n-1} \sum_{j=i}^{n-1} [3 + (n-1)]$

$= 1 + \sum_{i=0}^{n-1} (n-1)[3 + (n-1)]$

$= 1 + n(n-1)[3 + (n-1)]$

$= n^3 + n^2 - 2n + 1 \leq n^3 + n^2 + 1 \leq 3n^3$

Vậy độ phức tạp của f(n) là $O(n^3)$

Câu 8 Cho biết độ phức tạp của thuật toán sau:

```
1   int MaxSubSum4(const int a[], int n) {
2       int maxSum=0, thisSum=0;
3       for (int j=0; j<n; j++) {
4           thisSum+=a[j];
5           if (thisSum>maxSum) maxSum=thisSum;
6           else if (thisSum<0) thisSum=0;
7       }
8       return maxSum;
9   }
```

Đáp án:

+ Dòng 2 thực hiện 2 phép gán

+ Dòng 3 thực hiện 1 vòng lặp tối đa $L_1 = n$ (bước)

+ Dòng 4 thực hiện 1 phép gán

+ Dòng 5-6 là 1 câu điều kiện if-else nên theo nguyên lý cộng thì chương trình thực hiện 1 phép so sánh và tối đa là 1 phép gán

+ Dòng 8 là 1 phép trả về

Vậy $f(n) = 2 + n(1 + 2) + 1 = 3n + 3 \leq 6n$

Vậy độ phức tạp của f(n) là O(n)

Câu 9 Cho biết độ phức tạp của thuật toán sau:

```
1    sum = 0;
2    for (j = 0; j < n; j++)
3       for (k = 0; k < n*n; k++)
4          sum++;
```

Đáp án:

+ Dòng 1 thực hiện 1 phép gán

+ Dòng 2 thực hiện 1 vòng lặp tối đa $L_1 = n$ (bước)

+ Dòng 3 thực hiện 1 vòng lặp tối đa $L_2 = n * n$ (bước)

+ Dòng 4 thực hiện 1 phép gán

Vòng lặp dòng 2 và 3 lồng nhau nên theo nguyên lý nhân ta có:

$f(n) = 1 + n * n^2 * 1 = n^3 + 1 \leq 2n^3$

Vậy độ phức tạp của f(n) là $O(n^3)$