Data Structures and Algorithms

HUS - HKI (2025 - 2026)

Assignment 1

Lecturer: Nguyễn Thị Tâm - Trần Bá Tuấn

§ Algorithm Analysis §

Phần 1: MỤC TIÊU

- Sinh viên cần nắm được các kiến thức cơ bản về thuật toán và phân tích thuật toán.
- Sinh viên hiểu được cách đánh giá độ phức tạp của thuật toán và mối quan hệ theo ký pháp tiệm cận.
- Sinh viên nắm rõ cách chứng minh tính đúng đắn của thuật toán.

Phần 2: Thực hành

- Nộp bài bằng file ảnh chụp scan thành PDF với bài làm viết tay ra giấy.
- Quy tắc đặt tên file là Hw1_MaSinhVien_Hovaten.pdf
- Sinh viên không nộp bài sẽ nhận điểm 0 bài tập tuần.
- Sinh viên CÓ GIAN LẬN trong nộp bài tập sẽ bị ĐÌNH CHỈ môn học (điểm 0 cho tất cả các điểm thành phần).

(1) Trình bày sơ lược về độ phức tạp của thuật toán.

(2) Một số ví dụ

Ví dụ 1. *Tìm giới hạn trên (upper bound) cho:*

a) $f(n) = 3n + 5$

b) $f(n) = 4n^2 + 3$

c) $f(n) = n^4 + 100n^2 + 80$

d) $f(n) = 5n^3 - 5n^2$

e) $f(n) = 502$

Câu hỏi chung là: Độ phức tạp của thuật toán là gì?

Ví dụ 2. *Tính* $S = \dfrac{n * (n-1)}{2}$.

Ví dụ 3.

```python
for i in range(0, n):
    print('Current Number', i)
```

Ví dụ 4.

```
for i in range(0, n):
    print('Current Number', i)
    break
```

Ví dụ 5.

```
def function(n):
    i = 1
    while i <= n:
        i = i * 2
        print(i)
function(100)
```

Ví dụ 6.

```
for i in range(0, n):
    for j in range(0, n):
        print("Value of i, j", i, j)
```

Ví dụ 7.

```
public void function(int n){
        int i, j, k, count = 0;
        for(i = n/2; i <= n; i++)
            for(j = 1; j + n/2 <= n; j++)
                for(k = 1; k <= n; k = k * 2)
                    count++;
 }
```

Ví dụ 8.

```
public void function(int n) {
        int i, j, k, count = 0;
        for(i = n/2; i <= n; i++)
            for(j = 1; j <= n; j = 2 * j)
                for(k = 1; k <= n; k = k * 2)
                    count++;
}
```

(3) Bài tập vận dụng

Bài tập 1. *Độ phức tạp của thuật toán là gì?*

a) $T(n) = nlogn + 3n + 2$

b) $T(n) = nlog(n!) + 5n^2 + 7$

c) $T(n) = 1000n + 0.01n^2$

d) $T(n) = 100nlogn + n^3 + 100n$

e) $T(n) = 0.01nlogn + n(logn)^2$

Bài tập 2. *Độ phức tạp của thuật toán các đoạn code dưới đây là gì?*

a)

```
// Returns the sum of the integers in given array.
 public static int example1(int[ ] arr) {
    int n = arr.length, total = 0;
    for (int j=0; j < n; j++) // loop from 0 to n-1
        total += arr[j];
    return total;
 }
```

b)
```java
// Returns the sum of the integers with even index in given array.
public static int example2(int[ ] arr) {
    int n = arr.length, total = 0;
    for (int j=0; j < n; j += 2) // note the increment of 2
        total += arr[j];
    return total;
}
```

c)
```java
// Returns the sum of the prefix sums of given array.
public static int example3(int[ ] arr) {
    int n = arr.length, total = 0;
    for (int j=0; j < n; j++) // loop from 0 to n-1
        for (int k=0; k <= j; k++) // loop from 0 to j
            total += arr[j];
    return total;
}
```

d)
```java
// Returns the sum of the prefix sums of given array.
public static int example4(int[ ] arr) {
    int n = arr.length, prefix = 0, total = 0;
    for (int j=0; j < n; j++) { // loop from 0 to n-1
        prefix += arr[j];
        total += prefix;
    }
    return total;
}
```

e)
```java
 // Returns the number of times second array stores sum of prefix sums from
  ↪ first.
 public static int example5(int[ ] first, int[ ] second) {
 // assume equal-length arrays
    int n = first.length, count = 0;
    for (int i=0; i < n; i++) { // loop from 0 to n-1
        int total = 0;
        for (int j=0; j < n; j++) // loop from 0 to n-1
            for (int k=0; k <= j; k++) // loop from 0 to j
                total += first[k];
        if (second[i] == total) count++;
    }
    return count;
}
```

(4) Bài tập

- **(Master Theorem)** If n is a power of b, the solution to the recurrence

$$T(n) = \begin{cases} 1 & if \quad n \leq 1 \\ aT\left(\frac{n}{b}\right) + n^d & if \quad n > 1, a \geq 1, b > 1, d \geq 0 \end{cases}$$

is

$$T(n) = \begin{cases} O(n^d) & if \quad a < b^d \\ O(n^d logn) & if \quad a = b^d \\ O(n^{log_b a}) & if \quad a > b^d. \end{cases}$$

Ngoài ra, sinh viên có thể đọc thêm:

NHẮC LẠI: Master Theorem

$$T(n) = aT(\frac{n}{b}) + \Theta(n^k log^p n)$$

với $a \geq 1$, $b > 1$, $k \geq 0$ và số thực p.

1. Nếu $a > b^k$ thì $T(n) = \Theta(n^{\log_b a})$
2. Nếu $a = b^k$ thì
   a) Nếu $p > -1$ thì $T(n) = \Theta(n^{\log_b a} \log^{p+1} n)$
   b) Nếu $p = -1$ thì $T(n) = \Theta(n^{\log_b a} \log \log n)$
   c) Nếu $p < -1$ thì $T(n) = \Theta(n^{\log_b a})$
3. Nếu $a < b^k$ thì
   a) Nếu $p \geq 0$ thì $T(n) = \Theta(n^k \log^p n)$
   b) Nếu $p < 0$ thì $T(n) = \Theta(n^k)$

Ví dụ 9. *Xác định độ phức tạp của công thức dưới đây:*

$$T(n) = \begin{cases} 2T(n-1) - 1, & if \quad n > 0. \\ 1, & otherwise. \end{cases}$$

Ví dụ 10. *Xác định độ phức tạp của công thức dưới đây:*

$$T(n) = \begin{cases} 3T(n-1), & if \quad n > 0. \\ 1, & otherwise. \end{cases}$$

Ví dụ 11. *Giải công thức đệ quy xác định độ phức tạp thuật toán.*

1. $T(1) = 1$, *and for all* $n \geq 2$ *a power of 2*, $T(n) = 2T\left(\frac{n}{2}\right) + 6n - 1$.
2. $T(1) = 2$, *and for all* $n \geq 2$ *a power of 3*, $T(n) = 4T\left(\frac{n}{3}\right) + 3n - 5$.
3. $T(1) = 1$, *and for all* $n \geq 2$ *a power of 2*, $T(n) = 3T\left(\frac{n}{2}\right) + n^2 - n$.

Bài tập 3. *Xác định độ phức tạp thuật toán (Chọn thực hiện ít nhất 5 bài trong các bài dưới đây).*

- $T(1) = 1$, *and for all* $n \geq 2$, $T(n) = 3T(n-1) + 2$.
- $T(1) = 3$, *and for all* $n \geq 2$, $T(n) = T(n-1) + 2n - 3$.
- $T(1) = 1$, *and for all* $n \geq 2$, $T(n) = 2T(n-1) + n - 1$.
- $T(1) = 5$, *and for all* $n \geq 2$, $T(n) = 2T(n-1) + 3n + 1$.
- $T(1) = 4$, *and for all* $n \geq 2$ *a power of 2*, $T(n) = 2T\left(\frac{n}{2}\right) + 3n + 2$.
- $T(1) = 1$, *and for all* $n \geq 2$ *a power of 6*, $T(n) = 6T\left(\frac{n}{6}\right) + 2n + 3$.
- $T(1) = 3$, *and for all* $n \geq 2$ *a power of 6*, $T(n) = 6T\left(\frac{n}{6}\right) + 3n - 1$.
- $T(1) = 3$, *and for all* $n \geq 2$ *a power of 3*, $T(n) = 4T\left(\frac{n}{3}\right) + 2n - 1$.
- $T(1) = 4$, *and for all* $n \geq 2$ *a power of 2*, $T(n) = 3T\left(\frac{n}{2}\right) + n^2 - 2n + 1$.
- $T(1) = 1$, *and for all* $n \geq 2$ *a power of 2*, $T(n) = 3T\left(\frac{n}{2}\right) + n - 2$.