Phân tích thuật toán

Bài 1: Giả sử ta có các thuật toán với thời gian chạy như bên dưới. Hỏi mỗi thuật toán chậm đi bao nhiêu khi (i) gấp đôi kích thước đầu vào, hoặc (ii) tăng kích thước đầu vào một đơn vị?

(a) n^2

(b) n^3

(c) $100n^2$

(d) $n \log n$

(e) 2^n

Bài 2: Giả sử ta có các thuật toán với thời gian chạy như bên dưới, và có một chiếc máy tính có thể thực hiện 10^{10} tháo tác trong một giây. Đối với mỗi thuật toán, hỏi kích thước đầu vào n lớn nhất là bao nhiêu để có thể nhận được kết quả trong một giờ?

(a) n^2

(b) n^3

(c) $100n^2$

(d) $n \log n$

(e) 2^n

Bài 3: Sắp xếp danh sách các hàm sau đây theo thứ tự tăng dần của tốc độ tăng trưởng. Nghĩa là, nếu hàm g(n) được xếp sau hàm f(n) trong danh sách kết quả thì f(n) = O(g(n)).

$f_1(n) = n^{2,5}$

$f_2(n) = \sqrt{2n}$

$f_3(n) = n + 10$

$f_4(n) = 10^n$

$f_5(n) = 100^n$

$f_6(n) = n^2 \log n$

Bài 4: Giả sử ta có các hàm f và g sao cho f(n) = O(g(n)). Hỏi mỗi phát biểu sau đây là đúng hay sai? Hãy chứng minh hoặc nêu ra phản ví dụ.

(a) $\log_2 f(n) = O(\log_2 g(n))$

(b) $2^{f(n)} = O(2^{g(n)})$

(c) $f(n)^2 = O(g(n)^2)$

Bài 5: Phân tích thời gian chạy của thuật toán tìm phần tử lớn nhất (dùng ký hiệu O).

```
max ← a1
for i ← 2 to n
    if (ai > max)
        max ← ai
```

Bài 6: Phân tích thời gian chạy của thuật toán hợp nhất hai danh sách đã sắp xếp A = a_1, a_2, …, a_n và B = b_1, b_2, …, b_n thành một danh sách tổng thể C cũng được sắp xếp.

```
i ← 1, j ← 1
while (cả A và B không rỗng)
{
    if (a_i ≤ b_j)
        thêm a_i vào cuối C và tăng i một đơn vị
    else
        thêm b_j vào cuối C và tăng j một đơn vị
}
thêm phần còn lại của danh sách không rỗng vào cuối C
```

Bài 7: Cho một mảng A gồm n số nguyên A[1], A[2], …, A[n]. Ta muốn xuất ra một mảng hai chiều B có kích thước n x n, trong đó B[i, j] (với i < j) chứa tổng của các phần tử A[i] đến A[j], tức là tổng A[i] + A[i+1] + … + A[j]. Giá trị của các phần tử B[i, j] không xác định khi i ≥ j. Bên dưới là một thuật toán đơn giản giải quyết vấn đề đó. Hãy phân tích thời gian chạy của thuật toán này.

```
for i ← 1 to n {
    for j ← i+1 to n {
        cộng các phần tử từ A[i] đến A[j]
        lưu trữ kết quả vào B[i,j]
    }
}
```

Bài 8: Phân tích thời gian chạy của thuật toán tìm cặp điểm gần nhất trong n điểm (x_1, y_1), (x_2, y_2), …, (x_n, y_n).

```
p1 ← 1, p2 ← 2
min ← (x_1 - x_2)^2 + (y_1 - y_2)^2
for i ← 1 to n - 1 {
    for j ← i + 1 to n {
        d ← (x_i - x_j)^2 + (y_i - y_j)^2
        if (d < min) {
            min ← d
            p1 ← i, p2 ← j
        }
    }
}
```

Bài 9: Phân tích thời gian chạy của các đoạn chương trình C++ sau đây.

(a)
```
sum = 0;
for (i = 0; i < n; i++)
    sum++;
```

(b)
```
sum = 0;
for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
        sum++;
```

(c)
```
sum = 0;
for (i = 0; i < n; i++)
    for (j = 0; j < n*n; j++)
        sum++;
```

(d)
```
sum = 0;
for (i = 0; i < n; i++)
    for (j = 0; j < i; j++)
        sum++;
```

(e)
```
sum = 0;
for (i = 0; i < n; i++)
    for (j = 0; j < i*i; j++)
        for (k = 0; k < j; k++)
            sum++;
```

(f)
```
sum = 0;
for (i = 0; i < n; i++)
    for (j = 0; j < i*i; j++)
        if (j % i == 0)
            for (k = 0; k < j; k++)
                sum++;
```