Nhập môn phân tích độ phức tạp thuật toán – 21TN.

BÀI TẬP LÝ THUYẾT LẦN 3

Câu 1. (*3.5 điểm*) Ký hiệu $d(x)$ là số lượng ước dương của x, xét đẳng thức

$$d(1)+d(2)+\cdots+d(n)=\left\lfloor\frac{n}{1}\right\rfloor+\left\lfloor\frac{n}{2}\right\rfloor+\cdots+\left\lfloor\frac{n}{n}\right\rfloor \text{ với } n\in\mathbb{Z}^+. \ (*)$$

Chẳng hạn với $n=4$ thì ta có

$$d(1)+d(2)+d(3)+d(4)=1+2+2+3=8 \text{ và } \left\lfloor\frac{4}{1}\right\rfloor+\left\lfloor\frac{4}{2}\right\rfloor+\left\lfloor\frac{4}{3}\right\rfloor+\left\lfloor\frac{4}{4}\right\rfloor=4+2+1+1=8.$$

a) Dựa vào đẳng thức (*) ở trên, hãy ước lượng big-Θ theo n cho tổng $d(1)+d(2)+\cdots+d(n)$. Ngoài ra, hãy thử chứng minh đẳng thức (*) đúng với mọi n nguyên dương.

b) Từ tính chất được nêu ở trên, hãy ước lượng độ phức tạp của đoạn code sau (viết bằng C++) dựa trên việc đếm số phép gán:

```
int happy(int n){
    int i = 1, res = 0;
    while(i <= n){
        int k = 1, j = 1;
        while(k <= 2*i){
            res = res - k*i;
            k = k + 1;
        }
        while(j <= i){
            if(i % j == 0){
                int t = 1;
                while(t <= n){
                    res = res + j*t;
                    t = t + 1;
                }
            }
            j = j + 1;
        }
        i = i + 1;
    }
    return res;
}
```

Câu 2. (*3 điểm*) Xét đoạn code sau viết bằng C++ để kiểm tra mảng số nguyên dương $a_0, a_1, a_2, \ldots, a_{n-1}$ có thỏa mãn ràng buộc: với mọi $i=0,1,2,\ldots,n-1$ thì hai số a_i và i đều có cùng tính chẵn lẻ hay không:

```
bool checkParity(int n, int a[]){
    for(int i = 0; i < n; i++){
        if((a[i] % 2) != (i % 2)) return false;
    }
    return true;
}
```

Giả sử rằng mỗi phần tử của mảng được sinh ngẫu nhiên với xác suất là $\frac{1}{2}$ cho số chẵn và $\frac{1}{2}$ cho số lẻ. Xét $f_n(x)$ là hàm sinh xác suất ứng với số lần lặp của thuật toán.

a) Chứng minh rằng $f_n(x) = \frac{x}{2} + \frac{x^2}{4} + \frac{x^3}{8} + \cdots + \frac{x^{n-2}}{2^{n-2}} + \frac{x^{n-1}}{2^{n-1}} + \frac{x^n}{2^{n-1}}$.

b) Hãy tính độ phức tạp trung bình của thuật toán bằng cách xác định kỳ vọng của số lần lặp dựa theo hàm sinh trên.

Bài 3. (*3.5 điểm*) Trong bài này, ta sẽ đánh giá độ phức tạp thuật toán bằng thực nghiệm.

a) Cài đặt bổ sung các biến đếm số phép gán `count_assign` và số phép so sánh `count_compare` vào mã nguồn của các thuật toán sau đây, thực hiện đếm rồi lập bảng thống kê hoặc vẽ đồ thị mô tả sự thay đổi của các giá trị đó theo n (*SV có thể cài đặt lại bằng Python cũng được, bài làm trên notebook hoặc chụp code chèn vào file báo cáo đều được*).

- Thuật toán tính số Fibonacci dùng đệ quy:

```
int fibonacci(int n)
{
    if (n <= 1)
        return 1;
    else
        return fibonacci(n - 1) + fibonacci(n - 2);
}
```

- Thuật toán tính số Fibonacci không đệ quy:

```
int fibonacci(int n)
{
    if(n <= 1)
        return 1;

    int last = 1;
    int nextToLast = 1;
    int answer = 1;

    for(int i = 2; i <= n; i++)
    {
        answer = last + nextToLast;
        nextToLast = last;
        last = answer;
    }

    return answer;
}
```

b) Dùng phương pháp lý thuyết để đếm số phép so sánh được sử dụng ở các câu a) và số phép gán được sử dụng ở câu b), xét với n tổng quát (có thể đối chiếu lại với thực nghiệm).