TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN, ĐHQG-HCM

ĐỀ THI KẾT THÚC HỌC PHẦN

Học kỳ 1 – Năm học 2023-2024

MÃ LƯU TRỮ
(do phòng KT-ĐBCL ghi)
CK2324.1
CSC14007

Tên học phần:	**Nhập môn phân tích độ phức tạp thuật toán**	**Mã HP:**	**CSC14007**
Thời gian làm bài:	**105 phút**	Ngày thi:	10/01/2024

Ghi chú: *Sinh viên được phép sử dụng tài liệu giấy khi làm bài.*

Họ tên sinh viên: .... **MSSV:** .... **STT:** ....

Đề thi có 03 câu với 02 trang, tổng là 10 điểm.

Bài 1. (*3.5 điểm*)

1. (1.5đ) Với n nguyên dương, đặt

$$T(n) = \log(1)\sqrt{1} + \log(2)\sqrt{2} + \cdots + \log(n)\sqrt{n}.$$

Trình bày cách ước lượng độ lớn của $T(n)$ theo ký hiệu Θ.

2. (2.0đ) Cho mảng số nguyên $a[7] = \{4,2,6,7,5,3,1\}$ cần được sắp xếp tăng. Hãy thực hiện các yêu cầu sau:

i. So sánh số chu trình hoán vị của mảng trước và sau khi thực hiện một bước lặp với thuật toán *selection sort*.

ii. Không chạy từng bước cụ thể, hãy nêu cách gián tiếp xác định được số phép lặp của thuật toán *insertion sort* được sử dụng để sắp xếp mảng ban đầu.

Bài 2. (*3.5 điểm*) Cho đoạn chương trình sau đây (viết bằng C++):

```cpp
int process(int n){
    int i = 1, res = 0;
    while(i <= n){
        int j = 1, k = 1;
        while(j <= i * i * i){
            j = j + k;
            k = k + 1;
            res = res + j * k;
        }
        res = res - j * k;
        i = i + i;
    }
    return res;
}
```

a) (1.0đ) Hãy viết lại đoạn chương trình trên rồi thêm biến *count_compare*, *count_assign* vào các vị trí thích hợp để có thể đếm được số phép so sánh, số phép gán bằng thực nghiệm.

b) (2.0đ) Hãy ước lượng độ phức tạp của đoạn chương trình trên bằng lý thuyết, dựa trên việc đếm trực tiếp số phép gán. Bên dưới là gợi ý về số phép gán được thực hiện với một bộ dữ liệu n cụ thể (*SV có thể dùng để đối chiếu với kết quả ước lượng tìm được*):

n	16	64	256	1024	4096	65536	262144
count_assign	439	3387	26918	215101	1720509	110109238	880873431

Bài 3. (*3.0 điểm*) Trong HAI chọn MỘT.

3.1) Thuật toán Strassen là một thuật toán chia để trị, dùng để tìm tích C của hai ma trận vuông A, B có cùng kích thước $n \times n$ được mô tả như bên dưới:

Đầu tiên "cắt" ma trận A, B thành 4 khối kích thước $\frac{n}{2} \times \frac{n}{2}$ (nếu n lẻ thì ta sẽ thêm 1 dòng/cột gồm toàn số 0 vào cuối để chia được) như sau: $A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$, $B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$.

Thuật toán Strassen sẽ tính các ma trận:

$$\begin{cases} M_1 = (A_{11} + A_{22}) \times (B_{11} + B_{22}); \\ M_2 = (A_{21} + A_{22}) \times B_{11}; \\ M_3 = A_{11} \times (B_{12} - B_{22}); \\ M_4 = A_{22} \times (B_{21} - B_{11}); \\ M_5 = (A_{11} + A_{12}) \times B_{22}; \\ M_6 = (A_{21} - A_{11}) \times (B_{11} + B_{12}); \\ M_7 = (A_{12} - A_{22}) \times (B_{21} + B_{22}); \end{cases}$$

Từ đó sẽ tính được ma trận C theo công thức:

$$C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} M_1 + M_4 - M_5 + M_7 & M_3 + M_5 \\ M_2 + M_4 & M_1 - M_2 + M_3 + M_6 \end{bmatrix}.$$

Yêu cầu: giả sử các thao tác cộng, trừ hai ma trận cùng kích thước $m \times m$ sẽ có chi phí là $\Theta(m^2)$ với mọi $m \in \mathbb{Z}^+$, hãy mô tả công thức đệ quy cho $T(n)$ thích hợp để xác định chi phí tính toán ở trên. Từ đó, dùng định lý Master để ước lượng độ phức tạp cho $T(n)$ và so sánh sự hiệu quả của thuật toán này với thuật toán naive thông thường để nhân hai ma trận $n \times n$.

3.2) Cho đoạn code sau đây dùng để kiểm tra một mảng số nguyên gồm n phần tử có được sắp xếp tăng hay không (để đơn giản, ta xét các số trong mảng a là hoán vị tùy ý của các số $1, 2, 3, \ldots, n$). Gọi X là biến ngẫu nhiên mô tả số bước lặp của vòng for trong thuật toán.

```
bool check(int n, int a[]){
    for(int i = 1; i < n; i++){
        if(a[i] < a[i-1])
            return false;
        }
    }
    return true;
}
```

Yêu cầu: bằng các suy luận thích hợp, hãy chứng minh rằng:

a) $E(X) = \frac{1^2}{2!} + \frac{2^2}{3!} + \frac{3^2}{4!} + \cdots + \frac{(n-1)^2}{n!}$.

b) Thuật toán trên có độ phức tạp trung bình là $O(1)$.