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

BÀI TẬP LÝ THUYẾT LẦN 2

Câu 1. (3 *điểm*)

1. Cho mảng $a[7]=\{7,4,1,3,2,6,5\}$, hãy thống kê lại số phần tử cực đại bên trái trong từng bước lặp của thuật toán sắp xếp *insertion sort* (có 6 bước). Hỏi từ đó có thể có nhận xét gì?

2. Cho đoạn code sau đây dùng để kiểm tra xem có hai ký tự liên tiếp giống nhau trong một chuỗi ký tự s hay không. Gọi X là biến ngẫu nhiên mô tả số lần lặp cần dùng ứng với phép thử lấy ngẫu nhiên input là một chuỗi s sinh ra bởi hoán vị các ký tự trong từ "pappa". Hãy tính số lần lặp trung bình của đoạn code này dựa trên việc tính kỳ vọng của X:

```
bool check(string s){
    for(int i = 1; i < s.length(); i++)
        if(s[i-1] == s[i]) return true;
    return false;
}
```

Câu 2. (*4 điểm*) Xét bài toán sau đây: *cho số nguyên* $n \geq 2$ *và mảng* $a_0, a_1, \ldots, a_{n-1}$ *gồm các số nguyên có giá trị tuyệt đối không quá* 10^9. *Tìm giá trị lớn nhất của hiệu* $a_j - a_i$ *với* $0 \leq i < j < n$.

Ví dụ: nếu có mảng $a[6]=\{5,1,-3,0,3,-4\}$ thì đáp số sẽ là 6 (lấy $a_4 - a_2 = 3-(-3) = 6$).

a) Hãy mô tả ngắn gọn cách giải vét cạn brute-force cho bài toán trên (dùng mã giả hoặc code C++/Python) và đánh giá độ phức tạp tương ứng của thuật toán đó.

b) Cho ý tưởng về thuật toán "chia để trị" để giải bài toán trên như sau: chia mảng ban đầu thành 2 mảng con từ vị trí trung vị mid để tính giá trị lớn nhất của $a_j - a_i$ trong các trường hợp sau đây:

(I) $0 \leq i < j < mid$, (II) $mid \leq i < j < n$ và (III) $0 \leq i < mid \leq j < n$.

Đối với trường hợp (I) và (II), ta thực hiện gọi đệ quy giải hai bài toán riêng biệt cho hai mảng con tương ứng. Còn với trường hợp (III), ta chỉ cần lấy giá trị lớn nhất trong các số từ $mid \to n$ (đặt là $\max_R$) rồi trừ cho giá trị nhỏ nhất trong các số từ $0 \to mid$ (đặt là $\min_L$) là được; chú ý rằng để tìm giá trị lớn nhất $(\max_R)$ và nhỏ nhất $(\min_L)$, ta duyệt trực tiếp các mảng con qua một vòng lặp.

Yêu cầu: mô tả công thức đệ quy cho $T(n)$ thích hợp để xác định chi phí tính toán trong thuật toán nêu trên. Từ đó, sử dụng định lý Master để đánh giá độ phức tạp của $T(n)$.

c) Biết rằng có thể sử dụng cấu trúc dữ liệu *segment tree* (cây phân đoạn) để giải bài toán *range minimum/maximum queries* (truy vấn min/max trên nhiều mảng con), trong đó tốn chi phí $O(n)$ cho việc dựng cây một lần duy nhất và chi phí $O(\log n)$ để tìm giá trị nhỏ nhất & lớn nhất cho các phần tử trên mỗi mảng con. Hỏi nếu áp dụng cấu trúc dữ liệu này thì có thể cải tiến độ phức tạp của thuật toán thế nào (mô tả lại công thức cho $T(n)$ và đánh giá)?

Chú ý: không yêu cầu viết code hay mã giả cho câu b) và c).

Câu 3. (*3 điểm*) Xét đoạn code sau đây (viết bằng C++):

```
int compute(int k){
    int tmp = 0;
    while(k > 0){
        if(k%2 == 1) tmp = tmp+k;
        else tmp = tmp-1;
        k = k/2;
    }
    return tmp;
}

int solve(int n){
    int i = 1, res = 0;
    while(i <= n*n){
        res = res + i;
        res = res - compute(i);
        i = i+1;
    }
    return res;
}
```

a) Hãy viết lại đoạn code ứng với hai hàm **`solve()`** và **`compute()`** ở trên, trong đó có chèn thêm biến đếm *count_compare* vào các vị trí thích hợp để có thể đếm được số phép so sánh bằng thực nghiệm.

b) Hãy đánh giá theo n độ phức tạp thời gian của hàm **`solve()`** bằng lý thuyết, dựa trên việc đếm và ước lượng số phép gán cần thực hiện (*ở đây để đơn giản, cho phép dùng xấp xỉ* $\lfloor \log k \rfloor \approx \log k$ *trong các tính toán).*