ÔN TẬP

Contents

1. PHẦN ĐỘ PHỨC TẠP THUẬT TOÁN

1.1 Câu hỏi

a. Trình bày khái niệm độ phức tạp thời gian (Time Complexity) và độ phức tạp không gian (Space Complexity) của một thuật toán. Giải thích ý nghĩa của ký hiệu Big-O trong đánh giá thuật toán.

b. Cho thuật toán tính số Fibonacci theo cách đệ quy cổ điển. Nêu công thức truy hồi và đánh giá độ phức tạp.

c. Cho đoạn mã sau. Hãy phân tích độ phức tạp thời gian và cho biết có thể cải thiện bằng cách nào:

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

1.2 Câu hỏi

a. Trình bày nguyên lý hoạt động của **Heap Sort** và phân tích độ phức tạp của nó.

b. Đánh giá độ phức tạp thời gian thực hiện tốt nhất của hàm fastPower(x, n) sau theo ký hiệu O-lớn:

```
double fastPower(double x, int n)
{
    double fract;
    if(n == 0) return 1;
    fract = fastPower(x, n / 2);

    if(n % 2 == 0) return fract * fract;
    else return fract * fract * x;
}
```

Yêu cầu:

- Viết công thức truy hồi
- Suy ra độ phức tạp thời gian theo Big-O

c. So sánh ưu – nhược điểm của hai phương pháp tìm kiếm:

- Tìm kiếm nhị phân (Binary Search)
- Cây nhị phân tìm kiếm (Binary Search Tree – BST)

Theo các tiêu chí sau (trình bày dưới dạng bảng):

Tiêu chí	**Tìm kiếm nhị phân**	**Cây nhị phân tìm kiếm**
Bộ nhớ dùng lưu trữ các phần tử		
Thời gian tìm kiếm		
Thêm phần tử		
Xóa phần tử		
In ra danh sách các phần tử hiện có		

1.3 Câu hỏi

a. Đánh giá thời gian chạy của giải thuật sắp xếp chèn (Insertion sort). Giải thích ngắn gọn với các trường hợp sau: Mảng đầu vào đã được sắp xếp; Mảng đầu vào đã được sắp xếp nhưng đảo ngược; Mảng đầu vào gồm n phần tử giống nhau.

c. Cho hàm đệ quy sau. Viết công thức truy hồi và đánh giá độ phức tạp:

```
void recur(int n) {
    if (n <= 1) return;
    recur(n - 1);
    for (int i = 0; i < n; i++) {
        cout << i << " ";
    }
}
```

1.4 Câu hỏi

a. Giả sử một thuật toán chia bài toán kích thước n thành 3 bài toán con kích thước n/2 và xử lý mỗi bước mất O(n). Viết công thức truy hồi và đánh giá độ phức tạp.

b. Trình bày nguyên lý hoạt động của **Quick Sort**, phân tích độ phức tạp ở trường hợp tốt nhất và xấu nhất. (1,0 điểm)

c. Cho biết độ phức tạp thời gian và không gian của lời gọi đệ quy sau:

```
int calc(int n) {
    if (n == 0) return 0;
    return n + calc(n - 1);
}
```

2. PHẦN SẮP XẾP & TÌM KIẾM

2.1 Câu hỏi

a. Viết giải thuật BubbleSort để sắp xếp mảng số nguyên theo thứ tự tăng dần sử dụng phương pháp sắp xếp nổi bọt (Bubble sort).

b. Minh họa quá trình sắp xếp dãy số: 25, 16, 47, 33, 39, 66, 15, 3 theo thứ tự tăng dần sử dụng phương pháp sắp xếp nổi bọt (Bubble sort). (1,0 điểm)

c. Trình bày ưu điểm và nhược điểm của phương pháp sắp xếp nổi bọt (Bubble sort).

2.2 Câu hỏi

Cho một dãy số n nguyên a[0], a[1], ..., a[n-1].

a. Cho biết ý tưởng của giải thuật sắp xếp bằng phương pháp sắp xếp nhanh (quick sort).

b. Hãy minh họa quá trình sắp tăng dần cho dãy số 7, 9, 10, 6, 15, 16, 12 theo giải thuật sắp xếp nhanh (quick sort).

c. Giả sử dãy số nguyên n số trên đã được sắp xếp, viết hàm tìm kiếm nhị phân kiểm tra một giá trị nguyên x có nằm trong dãy số hay không? Nếu x có trong dãy số trả về vị trí của x, ngược lại không có trong dãy số trả về -1.

2.3 Câu hỏi

a. Viết giải thuật sortArray1 sắp xếp mảng số nguyên **giảm dần** sử dụng phương pháp sắp xếp chọn trực tiếp (selection sort).

b. Minh họa quá trình sắp xếp giảm dần dãy số 50, 61, 24, 13, 73, 99, 51, 2 theo giải thuật sắp xếp chọn trực tiếp.

c. Giả sử quá trình sắp xếp ở câu a thực thi trong thời gian $T_1(n)$, viết giải thuật sortArray2 sắp xếp mảng số nguyên giảm dần sử dụng phương pháp sắp xếp chọn trực tiếp với thời gian thực thi là $T_2(n)$ sao cho $T_2(n) < T_1(n)$.

2.4 Câu hỏi

a. Viết giải thuật sắp xếp một mảng số nguyên **giảm dần** sử dụng thuật toán sắp nổi bọt (bubble sort).

b. Minh họa nguyên lý hoạt động của thuật toán sắp nổi bọt (bubble sort) để sắp xếp mảng có chứa các phần tử trùng nhau [14, 31, 12, 28, 32, 25, 28, 15] theo thứ tự giảm dần.

3. DANH SÁCH LIÊN KẾT

3.1 Câu hỏi

a. Định nghĩa cấu trúc dữ liệu để biểu diễn danh sách liên kết đơn chứa các giá trị nguyên.

b. Viết giải thuật tìm kiếm một phần tử có giá trị là số nguyên k trong danh sách liên kết đơn đã được định nghĩa ở câu a. Trả về thứ tự của nút chứa k nếu tìm thấy, hoặc -1 nếu không tìm thấy.

c. Viết giải thuật in ra giá trị của các nút trong danh sách liên kết đơn có giá trị lớn hơn số nguyên k. Nếu không có nút nào thỏa mãn, in ra thông báo "Not found".

d. Viết giải thuật chèn một nút mới có giá trị là số nguyên k vào cuối danh sách liên kết đơn. Nếu danh sách rỗng, khởi tạo danh sách với nút mới làm nút đầu tiên.

3.2 Câu hỏi

Cho một danh sách liên kết đơn mỗi nút chứa một số nguyên với NODE như sau:

```
struct NODE {
    int data;
    NODE *next;
};
```

a. Viết hàm thêm một nút có giá trị nguyên x vào vị trí thứ i trong danh sách biết $0 \leq i \leq n$.

b. Viết hàm tìm và trả về vị trí của một nút có giá trị số nguyên x trong danh sách liên kết. Nếu x có trong danh sách trả về vị trí của x, ngược lại không có trong danh sách trả về -1.

c. Viết hàm tính và trả về giá trị trung bình cộng các giá trị nguyên có trong danh sách liên kết.

3.3 Câu hỏi

a. Hãy định nghĩa cấu trúc dữ liệu để biểu diễn danh sách liên kết (DSLK) đơn chứa các giá trị thực.

b. Viết giải thuật **tìm nút có giá trị nhỏ thứ hai** trong DSLK.

c. Viết giải thuật in ra các nút trong DSLK đơn có giá trị lớn hơn số thực k.

d. Viết giải thuật **xóa các nút trong DSLK đơn có giá trị nhỏ hơn số thực k**.

3.4 Câu hỏi

a. Định nghĩa cấu trúc dữ liệu biểu diễn hàng đợi (queue) chứa các giá trị nguyên.

b. Viết giải thuật kiểm tra xem hàng đợi có rỗng hay không.

c. Viết giải thuật thêm một phần tử mới vào cuối hàng đợi.

d. Viết giải thuật tính tổng các phần tử ở những vị trí chẵn của hàng đợi.

e. Viết giải thuật **xóa tất cả các phần tử** của hàng đợi.

4. STACK & QUEUE

4.1 Câu hỏi

a) Biểu thức dạng hậu tố là gì? Ưu điểm của biểu thức dạng hậu tố?

b) Định giá biểu thức dạng hậu tố sau (trình bày rõ các trạng thái trung gian của STACK):

10 2 3 + / 2 ^ 4 12 8 - % +

c) Vẽ cây biểu thức biểu diễn cho biểu thức ở phần b (không cần phải trình bày các bước trung gian)

4.2 Câu hỏi

Stack và biểu thức hậu tố (Postfix)

a. Trình bày cách sử dụng ngăn xếp (Stack) để tính giá trị của một biểu thức hậu tố (postfix).

b. Dùng Stack để tính giá trị biểu thức hậu tố sau và ghi lại thao tác của Stack tại mỗi bước:

```
5 6 2 + * 12 4 / -
```

c. Viết hàm C++ nhận đầu vào là chuỗi biểu thức hậu tố và trả về kết quả tính toán (giả sử các toán tử gồm +, -, *, /).

4.3 Câu hỏi

Biến đổi biểu thức từ trung tố (infix) sang hậu tố (postfix)

a. Trình bày giải thuật chuyển đổi biểu thức trung tố (infix) sang hậu tố (postfix) dùng ngăn xếp. Nêu rõ cách xử lý dấu ngoặc và độ ưu tiên toán tử.

b. Thực hiện chuyển đổi biểu thức sau sang hậu tố, ghi rõ từng bước sử dụng Stack:

(5 + 6) * (7 - 2) / 3

c. Cài đặt chương trình C++ chuyển đổi biểu thức trung tố (chứa các số nguyên và toán tử +, -, *, /, có ngoặc) sang hậu tố.

4.4 Câu hỏi

Ứng dụng hàng đợi: Giả lập hệ thống xử lý tác vụ

a. Trình bày cách sử dụng hàng đợi để mô phỏng hệ thống xử lý tác vụ đến theo thứ tự. Nêu ví dụ thực tế.
b. Mô phỏng hàng đợi xử lý các tác vụ đến theo thời gian sau:

Thời gian đến: 1 2 4 5

Tác vụ: T1 T2 T3 T4

Thời gian xử lý mỗi tác vụ: 3 đơn vị

Cho biết thứ tự hoàn thành các tác vụ và thời điểm hoàn thành.

c. Viết chương trình mô phỏng bài toán trên bằng C++ (sử dụng Queue).

4.5 Câu hỏi

Queue hai đầu (Deque)

a. Trình bày khái niệm hàng đợi hai đầu (Deque) và so sánh với Queue thông thường. Nêu ứng dụng thực tế.
b. Mô phỏng quá trình thực hiện các thao tác sau trên một deque rỗng (ghi rõ trạng thái deque):

push_back(10); push_front(20); pop_back(); push_front(30); pop_front();

c. Cài đặt cấu trúc Deque bằng danh sách liên kết đôi với các hàm push_front(), push_back(), pop_front(), pop_back().

5. CÂY TÌM KIẾM

5.1 Câu hỏi

a. Vẽ cây nhị phân tìm kiếm T với các nút được nhập theo thứ tự sau: 9 (nút gốc), 20, 46, 35, 41, 70, 19, 6, 11, 12, 18, 29, 50.

b. Xác định giá trị của các nút ở mức 3, các nút chỉ có một nút con, các nút lá và chiều cao của cây nhị phân tìm kiếm ở câu a.

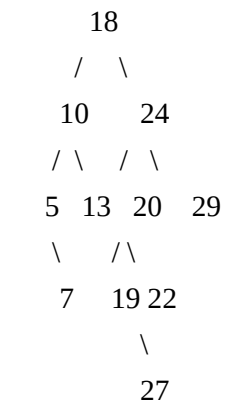
c. Viết giải thuật in ra giá trị của các nút là số nguyên tố trong cây nhị phân tìm kiếm. Nếu cây rỗng hoặc không có giá trị nào là số nguyên tố, in ra thông báo "Not found".

d. Viết giải thuật trả về giá trị của nút nhỏ nhất trong cây nhị phân tìm kiếm. Nếu cây rỗng, in ra thông báo "Not found".

5.2 Câu hỏi

Cho một cây nhị phân tìm kiếm (Binary Search Tree) giá trị tại mỗi nút (node) là số nguyên với NodeTree như sau:

```
struct NodeTree {
    int data;
    Node* left;
    Node* right;
};
```



a. Hãy viết dãy số trên của cây trên với phép duyệt **NRL** (Node-Right-Left).

b. Viết hàm thêm một nút vào cây nhị phân tìm kiếm.

c. Hãy viết hàm đếm số nút lá trên cây.

5.3 Câu hỏi

a. Vẽ cây nhị phân tìm kiếm T với các nút được nhập theo thứ tự sau: 50 (nút gốc), 61, 24, 13, 73, 99, 51, 4, 9, 10, 41, 52, 64.

b. Xác định các nút ở mức 2, các nút chỉ có một nút con, các nút lá, và chiều cao của cây nhị phân tìm kiếm.

c. Viết giải thuật đếm số nút lá trên cây nhị phân tìm kiếm.

d. Viết giải thuật tính tổng các giá trị nút trong cây nhị phân tìm kiếm và số lượng nút lá trên cây.

5.4 Câu hỏi

Cho cây nhị phân tìm kiếm T, mỗi nút trong cây T chứa một số nguyên, hãy thực hiện các nội dung sau:

a. Định nghĩa cấu trúc nút trong cây nhị phân tìm kiếm.

b. Viết giải thuật để xác định **chiều cao** của cây T.

c. Sử dụng giải thuật ở câu b, viết giải thuật kiểm tra xem cây T có phải là **cây nhị phân cân bằng** hay không.

d. Viết giải thuật để kiểm tra xem một cây nhị phân tìm kiếm **A có phải là cây con của cây nhị phân tìm kiếm B** hay không. Nếu A là cây con của B thì trả về true và ngược lại trả về false.

--- Hết ---