

Môn thi: Cấu trúc dữ liệu và giải thuật
Mã lớp: Các lớp IT003 – Hệ đại trà, chất lượng cao
Thời gian làm bài: 90 phút
(Sinh viên không được sử dụng tài liệu)

Câu 1: (2.5 điểm)

- Trình bày các bước giải thuật sắp xếp Quick Sort (không viết chương trình) để sắp xếp mảng số nguyên N phần tử **giảm dần**, cho biết độ phức tạp giải thuật.
- Trình bày các bước (vẽ từng bước) áp dụng giải thuật trong câu 1a để sắp xếp mảng số nguyên {10, 5, 30, 70, 40, 80, 90} giảm dần.

Đáp án tham khảo:

- Bước 1: Chọn một phần tử bất kì trong mảng làm 'pivot'
Bước 2: Chia mảng ra làm 2 phần, phần bên trái gồm những phần tử lớn hơn pivot, phần bên phải gồm những phần tử nhỏ hơn pivot.
Bước 3: Gọi đệ quy (lặp lại bước 1, 2) để sắp xếp 2 dãy con đã chia.
Bước 4: Đệ quy sẽ kết thúc khi không còn phần tử nào để phân chia; ta được dãy số đã được sắp xếp giảm dần.
→ Độ phức tạp của Quick Sort:
+ Tốt nhất: $O(n \log n)$
+ Trung bình: $O(n \log n)$ //khi mảng được chia đều
+ Xấu nhất: $O(n^2)$ //khi mảng đã sắp xếp hoặc đảo ngược.

- Mảng đã cho gồm $n = 7$ phần tử

10	5	30	70	40	80	90
----	---	----	----	----	----	----

Bước 1: Chọn phần tử 70 làm pivot

90	80	70	30	40	5	10
----	----	----	----	----	---	----

Bước 2: Chọn phần tử 80 làm pivot, phân hoạch đoạn [90, 80]

90	80	70	30	40	5	10
----	----	----	----	----	---	----

Bước 3: Chọn phần tử 5 làm pivot, phân hoạch đoạn [30, 10]

90	80	70	30	40	10	5
----	----	----	----	----	----	---

Bước 4: Chọn phần tử 40 làm pivot, phân hoạch đoạn [40, 30]

90	80	70	40	30	10	5
----	----	----	----	----	----	---

Bước 5: Kết thúc thuật toán, in ra màn hình dãy được sắp xếp

90	80	70	40	30	10	5
----	----	----	----	----	----	---

Câu 2: (4 điểm)

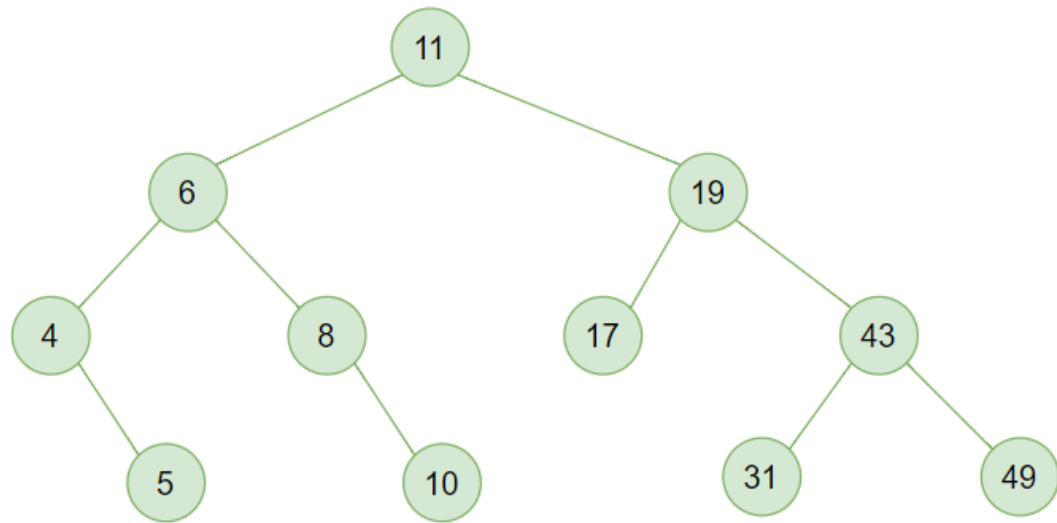
Cho dãy số sau: 11, 6, 8, 19, 4, 10, 5, 17, 43, 49, 31

Hãy thực hiện các yêu cầu sau:

- Xây dựng **cây nhị phân tìm kiếm** từ dãy số đã cho vào cây theo thứ tự thêm các số từ trái sang phải của dãy số.
- Duyệt cây trong câu 2a theo Node – Left – Right, Right – Left – Node.
- Xóa khỏi cây **lần lượt các nút 8, 11, 43, 6** (vẽ hình từng trường hợp) sao cho cây vẫn là cây nhị phân tìm kiếm sau khi xóa nút.
- Viết hàm in ra màn hình các nút trên cây có duy nhất một nút con.

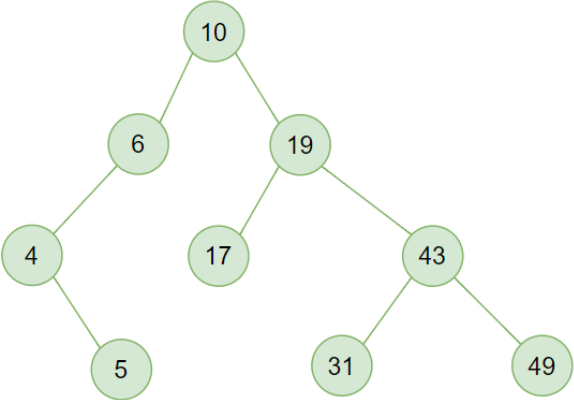
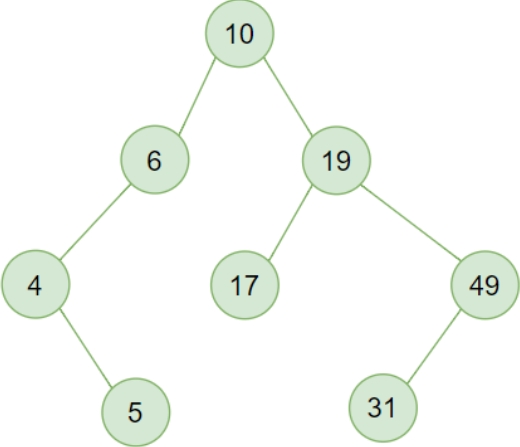
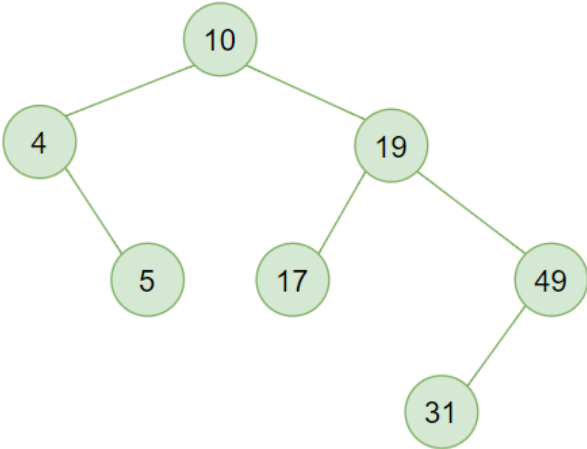
Đáp án tham khảo:

- Cây nhị phân tìm kiếm là cây nhị phân đảm bảo các nguyên tắc bố trí mỗi khóa, mỗi nút sao cho:
 - Các nút trong cây con bên trái nhỏ hơn nút hiện hành
 - Các nút trong cây con bên phải lớn hơn nút hiện hành.



- Duyệt cây theo Node – Left – Right: 11, 6, 4, 5, 8, 10, 19, 17, 43, 31, 49
Duyệt cây theo Right – Left – Node: 43, 31, 43, 17, 19, 10, 8, 5, 4, 6, 11
- Xóa 8, 11, 43, 6

Nút cần xóa	Dạng nút	Vẽ lại cây
8	nút có 1 cây con	<pre> graph TD 11((11)) --> 6((6)) 11 --> 19((19)) 6 --> 4((4)) 6 --> 10((10)) 4 --> 5((5)) 19 --> 17((17)) 19 --> 43((43)) 43 --> 31((31)) 43 --> 49((49)) </pre>

11	nút có 2 cây con (chọn nút thể mạng là nút 10)	
43	nút có đầy đủ 2 cây con (chọn nút thể mạng là nút 49)	
6	nút có 1 cây con	

```

d. void PrintNode(TREE T) {
    if (T != NULL) {
        if (T->pLeft == NULL && T->pRight != NULL || T->pLeft !=
        NULL && T->pRight == NULL)
            cout << T->key;
        PrintNode(T->pLeft);
        PrintNode(T->pRight);
    }
}

```

Câu 3: (2 điểm)

Cho bảng băm A kích thước 13 phần tử và tập khóa $K = \{10, 26, 52, 76, 13, 8, 3, 33, 60, 42\}$, ta cần nạp các giá trị khóa K vào bảng băm A sử dụng hàm băm $H(K) = K \% 13$.

Hãy vẽ bảng băm khi **thêm từng khóa K vào bảng A**, trong trường hợp xảy ra đụng độ, sử dụng phương pháp dò tuyến tính để giải quyết đụng độ.

Đáp án tham khảo:

Thêm 10: $H(10) = 10 \% 13 = 10$. Thêm '10' vào vị trí 10.

Thêm 26: $H(26) = 26 \% 13 = 0$. Thêm '26' vào vị trí 0.

Thêm 52: $H(52) = 52 \% 13 = 0$. Vị trí 0 chứa '26' → xảy ra đụng độ.

+ Băm lại lần 1: $H(52,1) = (52 + 1) \% 13 = 1$. Thêm '52' vào vị trí 1.

Thêm 76: $H(76) = 76 \% 13 = 9$. Thêm '76' vào vị trí 9.

Thêm 13: $H(13) = 13 \% 13 = 0$. Vị trí 0 chứa '26' → xảy ra đụng độ.

+ Băm lại lần 1: $H(13,1) = (13 + 1) \% 13 = 1$ → xảy ra đụng độ

+ Băm lại lần 2: $H(13,2) = (13 + 2) \% 13 = 2$. Thêm '13' vào vị trí 2.

Thêm 8: $H(8) = 8 \% 13 = 8$. Thêm '8' vào vị trí 8.

Thêm 3: $H(3) = 3 \% 13 = 3$. Thêm '3' vào vị trí 3.

Thêm 33: $H(33) = 33 \% 13 = 3$. Thêm '33' vào vị trí 7

Thêm 60: $H(60) = 60 \% 13 = 8$. Vị trí 8 chứa '8' → xảy ra đụng độ.

+ Băm lại lần 1: $H(60,1) = (60 + 1) \% 13 = 9$ → xảy ra đụng độ

+ Băm lại lần 2: $H(60,2) = (60 + 2) \% 13 = 10$ → xảy ra đụng độ

+ Băm lại lần 3: $H(60,3) = (60 + 3) \% 13 = 11$. Thêm '60' vào vị trí 11.

Thêm 42: $H(42) = 42 \% 13 = 3$. Vị trí 3 chứa '3' → xảy ra đụng độ

+ Băm lại lần 1: $H(42,1) = (42 + 1) \% 13 = 4$. Thêm '42' vào vị trí 4.

Bảng kết quả sau khi băm:

	<i>Key</i>
0	26
1	52
2	13
3	3
4	42
5	
6	
7	33
8	8

9	76
10	10
11	60
12	

Câu 4: (1.5 điểm)

Mạng xã hội là dịch vụ kết nối các thành viên, người dùng trên Internet lại với nhau dựa theo những tiêu chí, sở thích nào đó, với nhiều mục đích khác nhau, là nơi trao đổi thông tin, chia sẻ suy nghĩ, ý tưởng mà không bị giới hạn về không gian và thời gian. Các thành viên giao tiếp với các thành viên khác trong mạng, mỗi thành viên sẽ là một chủ thể trao đổi thông tin và tương tác với người khác.

Bạn được giao nhiệm vụ xây dựng một mạng xã hội thu nhỏ với các chức năng: kết bạn với nhau, xem thông tin của bạn của mình post trên mạng và like một post của bạn giống như cách trên Facebook, hãy đề xuất ý tưởng:

- Mô tả cấu trúc dữ liệu bạn nghĩ sẽ sử dụng phù hợp với các chức năng của mạng đã mô tả ở phần trên.
- Mô tả các giải thuật (các bước thực hiện giải thuật, không cài đặt) để hiện thực các chức năng của mạng đã mô tả ở phần trên và phù hợp với cấu trúc dữ liệu bạn đã đề nghị trong câu 4a.

Đáp án tham khảo:

```
a. struct person { //cấu trúc lưu trữ thông tin người
    string name;
    int ID;
};
struct post { //cấu trúc lưu trữ 1 post
    string content; //nội dung post
    string date; //thời gian post
    int likes; //số lượt thích của post
    int ID_like[100]; //danh sách các ID bạn đã thích
};
struct member { //cấu trúc lưu trữ 1 thành viên
    person user_profile; //thông tin thành viên
    post user_post; //danh sách post
    int number_post; //số lượng post
    person friend[100]; //danh sách bạn
    int number_friend; //số lượng bạn
};
member list[100]; //danh sách thành viên trong mạng
```

b. **Chức năng kết bạn:**

- Input: I của thành viên, ID của người cần kết bạn
- Output: Thông báo đã kết bạn thành công
- Ý tưởng:
 - + Bước 1: Kiểm tra ID của bạn mới đã có trong danh sách bạn bè chưa

- + Bước 2: Sau khi kết bạn, cập nhật trong danh sách bạn của mình (user) có thêm ID của bạn mới (thông qua cấu trúc dữ liệu mảng)
- + Bước 3: In ra màn hình thông báo đã kết bạn thành công.

Chức năng thích một post:

- Input: ID thành viên, danh sách bạn bè
- Output: Đã like post thành công
- Ý tưởng:
 - + Bước 1: Hiển thị danh sách các post trong danh sách bạn bè của thành viên
 - + Bước 2: Duyệt qua lần lượt các post cho đến khi hết danh sách post, nếu thích post nào đó chuyển qua bước 3, ngược lại bước 4.
 - + Bước 3: Sau khi like, cập nhật số lượt like trong post của bạn bè qua ID của bạn bè.
 - + Bước 4: quay lại bước 2.

Chức năng duyệt các post:

- Input: ID thành viên, danh sách bạn bè
- Output: Thông báo đã duyệt các post của bạn bè thành công
- Ý tưởng:
 - + Bước 1: Hiển thị danh sách các post trong danh sách bạn bè của thành viên
 - + Bước 2: Duyệt qua lần lượt các post cho đến khi hết danh sách post.

HẾT