VÕ TIẾN
Thảo luận kiến thức CNTT trường BK về KHMT(CScience), KTMT(CEngineering)
https://www.facebook.com/groups/khmt.ktmt.cse.bku

Cấu Trúc Dữ Liệu và Giải Thuật (DSA)

DSA3 - HK242

Cuối Kì

Thảo luận kiến thức CNTT trường BK
về KHMT(CScience), KTMT(CEngineering)
https://www.facebook.com/groups/khmt.ktmt.cse.bku

TP. HỒ CHÍ MINH, THÁNG 4/2025

Mục lục

1 Lý thuyết cơ bản Tree

1.1 Tree là gì?

Tree (Cây) là cấu trúc dữ liệu dạng phân cấp, mỗi phần tử gọi là **nút (node)**. Một cây có:

- **Nút gốc (Root)**: nút không có cha.
- **Nút lá (Leaf)**: nút không có con.
- **Nút cha - Nút con**: liên kết bởi cạnh.

1.2 Binary Tree

Cây nhị phân là cây mà mỗi nút có tối đa **2 con** (trái và phải). Các dạng cây nhị phân phổ biến:

- **Full Binary Tree**: mọi nút đều có 0 hoặc 2 con.
- **Complete Binary Tree**: các mức đều đầy, ngoại trừ mức cuối.

1.3 Tree Traversals (Duyệt cây)

Depth-First Search (DFS):

- **Preorder**: Nút cha → Con trái → Con phải.
- **Inorder**: Con trái → Nút cha → Con phải.
- **Postorder**: Con trái → Con phải → Nút cha.

Breadth-First Search (BFS): Duyệt theo từng tầng từ trên xuống dưới.

1.4 Binary Search Tree (BST)

Cây nhị phân có quy tắc:

$$\textit{Giá trị nút trái} \leq \textit{Cha} \leq \textit{Giá trị nút phải}$$

Dùng cho: tìm kiếm, thêm, xóa hiệu quả.

1.5 AVL Tree

BST có thêm **cân bằng độ cao**:

- Chênh lệch chiều cao giữa 2 nhánh ≤ 1.
- Dùng các phép **Rotate**: trái, phải, trái-phải, phải-trái.

1.6 Splay Tree

Cây nhị phân tự **đưa nút truy cập lên gốc** qua phép quay:

- Zig, Zag, Zig-Zig, Zag-Zag, Zig-Zag, Zag-Zig.

Thích hợp khi **truy vấn lặp lại**.

1.7 B-Trees

Cây nhiều nhánh, dùng cho bộ nhớ ngoài:

- Mỗi nút có từ t đến $2t$ con.
- Cân bằng chiều cao, tối ưu cho **database, filesystem**.

2 Câu hỏi trong các đề thi

1. Lần lượt chèn vào các giá trị 8 2 12 3 4 7 vào một AVL đang trống. Kết quả duyệt hậu thứ tự (Postorder) của cây sẽ là

   a) 2 3 7 12 8 4
   b) 2 3 4 7 8 12
   c) 3 2 4 8 7 12
   d) 4 3 2 8 7 12

2. Sai biệt chiều cao tối đa giữa các lá trong cây AVL có thể là bao nhiêu?

   a) 0 hoặc 1
   b) Nhiều nhất là 1
   c) n, trong đó n là số node
   d) log2 n trong đó n là số node

3. Cho cây tìm kiếm nhị phân tạo ra bởi việc chèn lần lượt 3, 7, 12, 5, 10, 2, 8. Chèn node 9 vào cây, nó sẽ ở vị trí nào?

   a) Là node con trái của node 5
   b) Là node cha của node 8
   c) Là node con phải của node 8
   d) Là node con trái của node 10

4. Số nút tối đa của một cây nhị phân có chiều cao h?

   a) h
   b) $2^{h-1}+1$
   c) $2^{h}-1$
   d) $2^{h+1}+1$

5. Làm thế nào để kiểm tra xem một cây nhị phân có phải là cây tìm kiếm nhị phân hay không?

   a) Thực hiện việc duyệt pre-order và kiểm tra xem kết quả có được sắp xếp hay không
   b) Thực hiện việc duyệt in-order và kiểm tra xem kết quả có được sắp xếp hay không
   c) Thực hiện việc duyệt post-order và kiểm tra xem kết quả có được sắp xếp hay không
   d) Thực hiện việc duyệt breadth-first và kiểm tra xem kết quả có được sắp xếp hay không

6. Làm thế nào để kiểm tra xem một cây nhị phân có phải là cây tìm kiếm nhị phân hay không?

   a) Thực hiện việc duyệt pre-order và kiểm tra xem kết quả có được sắp xếp hay không
   b) Thực hiện việc duyệt in-order và kiểm tra xem kết quả có được sắp xếp hay không
   c) Thực hiện việc duyệt post-order và kiểm tra xem kết quả có được sắp xếp hay không
   d) Thực hiện việc duyệt breadth-first và kiểm tra xem kết quả có được sắp xếp hay không

7. Trong thực tế, khi số lượng record trong cơ sở dữ liệu quá nhiều, và mỗi record lại chứa nhiều khoá khác nhau, thì giải pháp nào là phù hợp để phục vụ mục tiêu tìm kiếm trong điều kiện xử lý nhanh trong real-time?

   a) Sử dụng nhiều cây cân bằng AVL, hoặc RedBlack
   b) Sử dụng nhiều cây B-Tree hoặc các biến thể cao hơn của nó
   c) Sử dụng giải thuật băm với khoá là hàm heuristic tổ hợp các khoá
   d) Sử dụng k-D tree

8. Bất lợi của việc dùng cây Splay?

   a) Thao tác splay khó
   b) Không có bất lợi nào đáng kể
   c) Cây splay thực hiện các bước splay không cần thiết khi một node được đọc
   d) Chiều cao cây splay có thể tuyến tính khi truy xuất phần tử theo thứ tự tăng dần

9. Bất lợi của việc dùng cây Splay?

   a) Thao tác splay khó
   b) Không có bất lợi nào đáng kể
   c) Cây splay thực hiện các bước splay không cần thiết khi một node được đọc
   d) Chiều cao cây splay có thể tuyến tính khi truy xuất phần tử theo thứ tự tăng dần

   Chọn hiện thực đúng của hàm `prtIn` xuất cây nhị phân theo duyệt cây trung thứ tự (in-order traversal):

a) `if (root != NULL){ cout « root->data; prtIn(root->left); prtIn(root->right);}`
b) `if (root != NULL){ cout « root->data; prtIn(root->right); prtIn(root->left);}`
c) `if (root != NULL){ prtIn(root->left); cout « root->data; prtIn(root->right);}`
d) `if (root != NULL){ prtIn(root->right); prtIn(root->left); cout « root->data;}`

10. Độ phức tạp thời gian trong trường hợp xấu nhất khi chèn n^2 phần tử vào cây AVL ban đầu có n phần tử là:

a) $O(n \log_2 n)$ b) $O(n^2 \log_2 n)$ c) $O(n^3)$ d) $O(n^3 \log_2 n)$

11. Chọn phát biểu đúng

a) Cây tìm kiếm nhị phân luôn cho phép tìm kiếm với độ phức tạp O(log2 N)
b) Cây Heap cho phép tìm kiếm với độ phức tạp O(log2 N)
c) Giải thuật Hashing cho phép tìm kiếm với độ phức tạp O(1)
d) Ba phát biểu trên đều đúng

12. Chọn phát biểu đúng

a) Cây tìm kiếm nhị phân luôn cho phép tìm kiếm với độ phức tạp O(log2 N)
b) Cây Heap cho phép tìm kiếm với độ phức tạp O(log2 N)
c) Giải thuật Hashing cho phép tìm kiếm với độ phức tạp O(1)
d) Ba phát biểu trên đều đúng

13. Trong cây nhị phân, độ sâu (depth) của một node là:

a) Số lượng nút con trực tiếp của nút đó
b) Số lượng nút cha trực tiếp của nút đó
c) Số lượng nút con trực tiếp và gián tiếp của nút đó
d) Số lượng nút cha trực tiếp và gián tiếp của nút đó

14. Lần lượt chèn vào các giá trị 2 8 12 3 4 vào một AVL đang trống. Kết quả duyệt tiền thứ tự (Preorder) của cây sẽ là

a) 8 2 3 4 12
b) 2 4 3 12 8
c) 8 3 12 2 4
d) 8 3 2 4 12

15. Làm thế nào để tìm tổ tiên chung thấp nhất (lowest common ancestor - LCA) của hai nút trong một cây tìm kiếm nhị phân?

a) Bắt đầu từ gốc và duyệt cây cho đến khi cả hai nút đều ở cùng một phía của nút hiện tại
b) Bắt đầu từ gốc và duyệt cây cho đến khi cả hai nút bằng với nút hiện tại
c) Bắt đầu từ gốc và duyệt cây cho đến khi cả hai nút nhỏ hơn hoặc lớn hơn nút hiện tại
d) Bắt đầu từ gốc và duyệt cây cho đến khi cả hai nút ở hai phía khác nhau của nút hiện tại hoặc một trong số chúng là nút hiện tại

16. Chèn lần lượt các key sau vào một cây B-Tree (m=3) rỗng: 50, 40, 70, 55, 45, 43, 56, 58. Trong cây trên có bao nhiêu node có nhiều hơn 1 entry?

a) 0 b) 1 c) 2 d) 3

17. Cho cây tìm kiếm nhị phân, tiến hành chèn các giá trị lần lượt là 5, 10, 15, 20, 30, 35, 40. Chèn 27 vào cây, ở độ sâu 3 (root ở độ sâu 0) ta thấy node:

a) 20 b) 27 c) 30 d) 35

18. Giả sử các số 7, 5, 1, 8, 3, 6, 0, 9, 4, 2 theo thứ tự được chèn vào cây tìm kiếm nhị phân (BST) rỗng ban đầu. Cây nhị phân tìm kiếm sử dụng thứ tự thông thường trên các số tự nhiên. Kết quả duyệt cây trung thứ tự là gì?

a) 0 1 2 3 4 5 6 7 8 9
b) 0 2 4 3 1 6 5 9 8 7
c) 7 5 1 0 3 2 4 6 8 9
d) 9 8 6 4 2 3 0 1 5 7

19. Dùng giải thuật tìm kiếm nhị phân (binary search) để tìm vị trí của 48 trong dãy sau: [1,5,8,11,19,22,31,35,40,45,48,49,50] Cần phải thực hiện bao nhiêu lần so sánh?

a) 2 b) 3 c) 4 d) 5

20. Số node tối thiểu của một cây nhị phân có chiều cao h là:

a) h b) $h+1$ c) 2^{h-1} d) $2^{h-1}+1$

21. Duyệt trung thứ tự cây nhị phân cho kết quả là (A,B,C,D,E,F,G), duyệt hậu thứ tự cho kết quả là (B,D,C,A,F,G,E), hãy cho biết cây con bên phải của nút gốc có tổng cộng bao nhiêu nút

a) 2 b) 3 c) 4 d) 5

22. Cho cây AVL với 2 nút, trong đó nút gốc có giá trị 15, nút còn lại có giá trị 20. Khi chèn thêm nút có giá trị 19, cần phải

a) Thực hiện phép quay đơn (single rotation) về bên trái
b) Thực hiện phép quay đơn về bên phải
c) Thực hiện phép quay kép (double rotation)
d) Không cần thực hiện phép quay

23. Xây dựng cây B (B-Tree) bậc 3 từ dãy khóa sau (18, 4, 2, 46, 48, 29, 30). Hãy cho biết chèn các khóa nào dưới đây vào cây B ở trên, thì sẽ xảy ra việc tách nút (split node)

a) 1, 32, 53 b) 1, 19, 20 c) 1, 19, 32 d) 1, 19, 32, 53

24. Cho biết nhận định nào dưới đây về cây B (BTree) bậc m là KHÔNG đúng

a) Nút gốc có nhiều nhất m cây con
b) Tất cả các nút lá nằm trên cùng một mức
c) Các khóa trong nút gốc được sắp xếp theo thứ tự
d) Nút trong có ít nhất m/2 + 1 cây con khác rỗng (nếu m chẵn) hoặc m/2 - 1 cây con khác rỗng (nếu m lẻ)

25. Lần lượt thêm các số sau vào 1 cây AVL rỗng: 3, 10, 20, 9, 1, 5. Liệt kê các Node nằm ở tầng 2 (level 2) trong cây cuối cùng thu được theo thứ tự tăng dần.

a) 3, 5, 20 b) 3, 10
c) 1, 5, 20 d) 3, 5, 10

26. Biết rằng duyệt tiền thứ tự một cây nhị phân cho kết quả là (A,B,C,D,E,K,F,G), kết quả duyệt trung thứ tự là (B,C,A,D,K,E,F,G). Hãy cho biết kết quả duyệt hậu thứ tự

a) C,B,K,G,F,E,D,A b) C,B,K,G,E,F,D,A
c) C,B,G,K,F,E,D,A d) Cả ba đáp án còn lại đều sai

27. Cho cây AVL có kết quả duyệt tiền thứ tự là (13, 10, 5, 4, 6, 11, 15, 16), kết quả duyệt trung thứ tự là (4, 5, 6, 10, 11, 13, 15, 16). Chèn thêm nút có giá trị 7 vào cây AVL ở trên, kết quả duyệt tiền thứ tự sau khi chèn giá trị 7 là

a) 13, 5, 4, 6, 10, 7, 11, 15, 16 b) 13, 6, 5, 4, 10, 7, 11, 15, 16
c) 6, 13, 5, 4, 10, 7, 11, 15, 16 d) 6, 5, 4, 10, 7, 11, 13, 15, 16

28. Lần lượt thêm các số sau vào một câu AVL rỗng: 50, 23, 70, 19, 29, 65, 83, 25, 35, 53. Sau khi thêm xong, lần lượt xoá các số sau ra khỏi cây AVL: 19, 23, 83. Liệt kê các Node nằm ở tầng 2 (level 2) trong cây cuối cùng thu được theo thứ tự từ trái sang phải.

a) 25, 29, 50, 70 b) 25, 35, 53, 70
c) 25, 50, 53, 70 d) 29, 35, 65, 70

29. Cây B (B-Tree) bậc m = 4 và chiều cao bằng 4, có tối đa bao nhiêu entry

a) 255 b) 160 c) 127 d) 64

30. Cho biết nhận định nào sau đây là SAI về cây AVL
i) Cây AVL là cây tìm kiếm nhị phân (Binary Search Tree)
ii) Cây AVL là cây cân bằng
iii) Cây AVL là cây đầy đủ hoặc gần đầy đủ

a) Nhận định i) sai
b) Nhận định ii) sai
c) Nhận định iii) sai
d) Không có nhận định nào sai

31. Cho biết những nhận định nào sau đây là ĐÚNG về cây B (B-Tree) bậc m i) Mỗi nút phải có ít nhất 2 cây con không rỗng
ii) Mỗi nút có nhiều nhất m – 1 dữ liệu (entry)
iii) Tất cả các nút lá nằm trên cùng một mức

a) Chỉ có i) và iii) đúng
b) Chỉ có i) và ii) đúng
c) Chỉ có ii) và iii) đúng
d) Cả ba nhận định trên đều đúng

32. Biết rằng duyệt tiền thứ tự và duyệt hậu thứ tự một cây nhị phân cho kết quả ngược nhau. Ví dụ, duyệt tiền thứ tự là (A,B,C,D), duyệt hậu thứ tự là (D,C,B,A), nhận định nào sau đây đúng nhất

a) Bất cứ nút nào cũng không có cây con bên trái
b) Bất cứ nút nào cũng không có cây con bên phải
c) Cây có chiều cao đúng bằng tổng số nút
d) Cây nhị phân hoặc là rỗng hoặc chỉ có một nút

33. Cho cây nhị phân có các nút A, B, C, D, E, F, G. Duyệt tiền thứ tự (PreOrder) trên cây nhị phân thu được (A,B,C,D,E,F,G). Khi duyệt theo trung thứ tự (InOrder), kết quả có thể là

a) (C,A,B,D,E,F,G)
b) (A,B,C,D,E,F,G)
c) (D,A,C,E,F,B,G)
d) (A,D,C,F,E,G,B)

34. Cho cây nhị phân có các nút A, B, C, D, E, F, G. Duyệt tiền thứ tự (PreOrder) trên cây nhị phân thu được (A,B,C,D,E,F,G). Khi duyệt theo trung thứ tự (InOrder), kết quả có thể là

a) (C,A,B,D,E,F,G)
b) (A,B,C,D,E,F,G)
c) (D,A,C,E,F,B,G)
d) (A,D,C,F,E,G,B)

35. Biết rằng duyệt tiền thứ tự một cây nhị phân cho kết quả là A,B,C,D,E,K,F,G, kết quả duyệt trung thứ tự là B.C,A,D,K,E,F,G. Hãy cho biết kết quả duyệt hậu thứ tự

a) C, B, K. G, F, E, D, A
b) C, B, K, G, E, F, D, A
c) C, B, G, K, F, E, D, A
d) 3 đáp án trên đều sai

36. Biết rằng duyệt tiền thứ tự và duyệt hậu thứ tự một cây nhị phân cho kết quả ngược nhau (Ví dụ, duyệt tiền thứ tự là ABCD, duyệt hậu thứ tự là DCBA), nhận định nào sau đây đúng

a) Cây nhị phân hoặc là rỗng hoặc chỉ có một nút
b) Bất cứ nút nào cũng không có cây con bên trái
c) Cây có chiều cao đúng bằng tổng số nút
d) Bất cứ nút nào cũng không có cây con bên phải

37. Cho một cây nhị phân có chiều cao $H = 5$. Gọi N là số nút trên cây thì $A \le N \le B$ Giá trị của $B + A$

a) 36
b) 55
c) 63
d) 94

38. Cho một cây nhị phân tìm kiếm có hậu thứ tự lần lượt là 28, 30, 29, 33, 35, 34, 31. Cây nhị phân trên có tổng các khóa trên các nút lá là.

a) 126
b) 127
c) 125
d) 128

39. Cho một cây AVL được biểu diễn dưới dạng dấu ngoặc như sau: 44(38(31(27,34), 41(39,N)),109(89, 114)) với N là $NULL$ Lần lượt xóa các nút 109 và 89 ra khỏi cây (luôn lấy các nút lớn trên cây con bên trái), tổng các khóa trên nút ở mức (level) 0 và 1 là.

a) 83 b) 81 c) 85 d) 84

40. Một cây nhị phân có:

- Hậu thứ tự là 32, 38, 42, 40, 34, 25.
- Trung thứ tự là 32, 25, 38, 34, 42, 40.

Tổng các nút trên mức (level) 0 và 1 của cây là.

a) 91 b) 90 c) 92 d) 93

41. Cho các phát biểu sau đây:

(a) Tất cả các nút trong một cây đều có bậc nội (indegree) bằng 1.

(b) Trong một cây nhị phân, một nút bất kỳ có thể có ít hơn 2 con.

(c) Tất cả các nút (ngoại trừ nút lá) trong một cây đều có bậc ngoại lớn hơn hoặc bằng 1.

(d) Trong một cây nhị phân, nếu biết chiều cao của cây là 4 thì số nút của cây là 15.

Số phát biểu đúng là:

a) 0 b) 1 c) 2 d) 3

42. Cây nhị phân có 100 node, tính chiều cao nhỏ nhất

a) 7 b) 5 c) 6 d) 8

43. Cây nhị phân có 100 node, chiều cao lớn nhất

a) 100 b) 99 c) 101 d) 98

44. Cây nhị phân có 100 node, số node lá của cây

a) 100 b) 99 c) 50 d) 51

45. Cây nhị phân có 100 node, số lượng cây có thể là cây có chiều cao lớn nhất

a) 2^{100} b) 2^{99} c) 1 d) 100

46. Hoàn thiện đoạn code sau cho hàm insert một giá trị mới vào cây AVL, vị trí trống là lời gọi hàm nào? Các đặc tả khác như trong slide bài giảng và bài thí nghiệm.

```
Node* insert(Node*& subroot, const T& value) {
    Node* pNew = new Node(value);
    if (!subroot) return pNew;
    if (value < subroot->data) {
        subroot->pLeft = inseOrt(subroot->pLeft, value);
    } else if (value >= subroot->data) {
        subroot->pRight = insert(subroot->pRight, value);
    } else return subroot;
    int balance = getBalance(subroot);
    if (balance > 1 && value < subroot->pLeft->data) return /*Code*/;
}
```

a) rotateLeft(subroot) b) rotateLeft(subroot->pLeft)
c) rotateRight(subroot) d) rotateRight(root)

Cho cấu trúc dữ liệu của một node trên cây nhị phân như sau:

```
template<typename E>
class BNode {
    virtual E &element() = 0; // return the node value
    virtual BNode<E> *left() const = 0; // return the node's left child
    virtual BNode<E> *right() const = 0; // return the node's right child
};
```

Hãy chọn mã thích hợp để điền vào các chỗ trống trong hàm 'checkNode' để hàm 'isComplete' có thể xác định được cây nhị phân có nút gốc là 'root' có phải là cây hoàn chỉnh (complete) không?

```
template<typename E>
bool checkNode(BNode<E> *node, bool &flag, Queue<BNode<E>> &q) {
    if (node) {
        if (/*Code1*/) return false;
        /*Code2*/;
    } else flag = true;
    return true;
}

template<typename E>
bool isComplete(BNode<E> *root) {
    if (root == nullptr) return true;
    Queue<BNode<E>*> q;
    q.push(root);
    bool flag = false;
    while (!q.empty()) {
        BNode<E> *temp = q.front();
        q.pop();
        if (!checkNode(temp->left(), flag, q))
            return false;
        if (!checkNode(temp->right(), flag, q))
            return false;
    }
}
```

47. Điền vào Code1:

a) flag b) flag == false c) q.empty() d) q.empty() == false

48. Điền vào Code2:

a) node = q.front() b) flag = false c) q.push(node) d) q.pop()

49. Cho hàm chèn node vào cây tìm kiếm nhị phân, trả về gốc của cây.

```
TreeNode* bstInsert(TreeNode* root, int val) {
    if (root == nullptr) { return new TreeNode(val); }
    TreeNode* curr = root;
    TreeNode* prev = nullptr;
    while (curr != nullptr) {
        prev = curr;
        // Code
    }
    if (val < prev->val) { prev->left = new TreeNode(val); }
    else { prev->right = new TreeNode(val); }
    return root;
}
```

Không chấp nhận khoá trùng trên cây, điền vào Code để hoàn thành hàm trên.

a) if (val < curr->val) curr = curr->left; else curr = curr->right;
b) if (val < curr->val) curr = curr->left; else if (val > curr->val) curr = curr->right; else break;
c) if (val < curr->val) curr = curr->left; else if (val > curr->val) curr = curr->right;
d) if (val < curr->val) curr = curr->left; else if (val > curr->val) curr = curr->right; else return root;

50. Cho hàm sau:

```
int countL0(Node *root) {
    if (!root == NULL) return 0;
    // Code
    return countL0(root->left) + countL0(root->right) + 1;
}
```

Điền vào Code để hoàn thành hàm trả về số node lá trong cây nhị phân.

a) if (!root->left && !root->right) return 1;
b) if (!root->left || !root->right) return 1;
c) if (!root->left && !root->right) return 0;
d) if (!root->left || !root->right) return 0;

Cho cấu trúc dữ liệu của một node trên cây nhị phân như sau:

```
template<typename E>
class BNode {
    virtual E &element() = 0; // return the node value
    virtual BNode<E> *left() const = 0; // return the node's left child
    virtual BNode<E> *right() const = 0; // return the node's right child
};
```

lowestCommonAncestor với kiểu BNode, ta cần định nghĩa hàm tìm tổ tiên trong cây nhị phân của BNode

```
template<typename E>
BNode<E>* lowestCommonAncestor(BNode<E>* root, BNode<E>* p, BNode<E>* q) {
    if (root == nullptr || root == p || root == q)
        return root;

    BNode<E>* left = /*Code 1*/;
    BNode<E>* right = /*TODO*/;

    if (left != nullptr && right != nullptr)
        return root;

    return /*Code 2*/;
}
```

51. Điền vào Code1:

a) lowestCommonAncestor(root->left(), p->left, q)
b) lowestCommonAncestor(root->left(), p, q)
c) lowestCommonAncestor(root->left(), p, q->left)
d) lowestCommonAncestor(root->left(), p->left, q->left)

52. Điền vào Code2:

a) left != nullptr ? left : right
b) left == nullptr ? left : right
c) left != nullptr ? right : left
d) cả 3 đều sai.

kiểm tra độ xâu của mỗi node. Nếu độ xâu của một node là chẵn, bạn cộng giá trị của nó vào tổng.

```
template<typename E>
int sumEvenDepthNodes(BNode<E>* root) {
    if (root == nullptr) return 0;
    std::queue<std::pair<BNode<E>*, int>> q;
    q.push({root, 0});
    int sum = 0;

    while (!q.empty()) {
        auto [current, depth] = q.front();
```

```
            q.pop();

            if (/*Code 1*/) {
                //TODO
            }

            if (current->left()) {
                /*Code 2*/
            }
            if (current->right()) {
                 //TODO
            }
        }

        return sum;
    }
```

53. Điền vào Code1:

a) depth % 2 == 0
b) depth % 2 != 0
c) depth == 0
d) depth != 0

54. Điền vào Code2:

a) q.push(current->left(), depth + 1);
b) q.push(current->right(), depth - 1);
c) q.push(current->left(), depth + 1);
d) q.push(current->right(), depth - 1);