Cau truc du lieu va giai thuat_ Nhom 08

Bắt đầu vào lúc	Saturday, 11 December 2021, 4:40 PM
Trạng thái	Đã xong
Kết thúc lúc	Saturday, 11 December 2021, 10:51 PM
Thời gian thực hiện	6 giờ 11 phút

Câu hỏi 1

Hoàn thành

Đạt điểm 1,00

Xây dựng bảng Hash sử dụng phương pháp Linear Probing

Note:

K: số lượng key trong bảng hash

N số lượng phần tử của mảng arr cần add vào bảng hash

```
#include <stdio.h>
struct HashTable {
	int K;
	int* table;
	void construct(int arr[], int N) {
		// your code here
	int key;
		table = new int[K];
		for (int i = 0; i < K; i++)
			table[i] = NULL;
		for (int i = 0; i < N; i++)
		{
			key = arr[i] % K;
			while (table[key] != NULL)
			{
				if ((key + 1) < K)
					key++;
				else
				{
					key = 0;
				}
			}
			table[key] = arr[i];
		}
		// end your code
	}
};
```

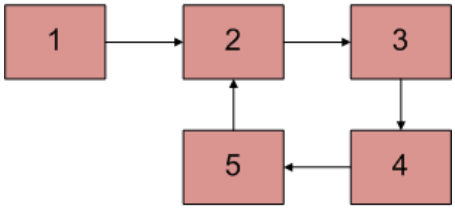
Câu hỏi 2

Hoàn thành

Đạt điểm 1,00

xóa liên kết vòng trong danh sách liên kết đơn:

Danh sách liên kết đơn tạo thành vòng như hình:



Hãy viết hàm để xóa liên kết vòng. Kết quả: `1 --> 2 ---> 3 ----> 4 ----> 5 ---> NULL`

Gợi ý: sử dụng hàm hash với key là địa chỉ của mỗi node trong danh sách liên kết đơn

Node head là phần tử đầu tiên của danh sách

Kết quả của hàm là contro tới Node đầu tiên của danh sách mới

```
#include <stdio.h>
struct HashTable {
	int K;
	int* table;
	void construct(int arr[], int N) {
		// your code here
		// end your code
	}
};
struct Node {
	int data;
	Node* next;
};
Node*  breakCycle(Node* head)
{
}
```

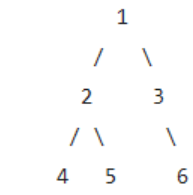
Câu hỏi 3

Hoàn thành

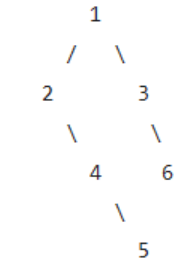
Đạt điểm 1,00

Cho cây nhị phân Tree. Tìm tất cả giá trị của node bên trái của cây. Kết quả trả về hàm là một mảng các giá trị và N là số lượng phần tử của mảng

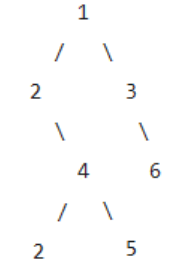
Ví dụ:



Output: 1,2,4 N =3



Output: 1,2,4,5 N=4



Output: 1,2,4,2 N=4

```
#include <stdio.h>
struct BNode { //BST --> AVL
    int data;
    BNode* left, * right;
    // your code here
void init() {

        left = right = nullptr;

    }
    // end your code
};
struct Tree {
    BNode* root= nullptr;
    // your code here
void init(BNode*& node) {
```

```
            node = new BNode;

            node->init();

        }
        void insert(int value, BNode*& parent) {
            if (parent == nullptr) {
                init(parent);
                parent->data = value;
                return;
            }
            if (value < parent->data) {
                insert(value, parent->left);

            }
            else {
                insert(value, parent->right);

            }
            return;
        }
        //end your code
};
int* leftView(Tree * tree,  int &N)
{
    // your code here
N = 0;

    BNode* temp = tree->root;
    int* res = new int[100];

    while (temp != nullptr)

    {

        res[N] = temp->data;

        if (temp->left != nullptr)  temp = temp->left;

        else   temp = temp->right;

        N++;

    }

    return res;
```

```
        //end your code
}
```

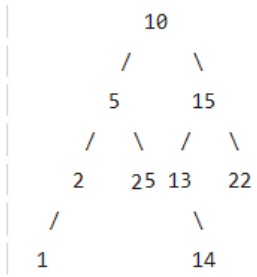
Câu hỏi 4

Hoàn thành

Đạt điểm 1,00

Hiện thực thuật toán tìm kiếm nhị phân trên cây nhị phân tìm kiếm (BST)

Input



Chiều cao của cây height tại node lá bằng 0, các node không phải lá có chiều cao height là số node tới node lá bên trái + 1. Ví dụ: node 2 có height=1, node 1 có height=0, node 22 có height=0, node 10 có height=3

Viết hàm tìm chiều cao height của node có giá trị K trong cây

```
#include <stdio.h>
struct BNode {
    int data;
    BNode* left, * right;
    int height;
    // your code here
    // end your code
};
struct BSTree {
    BNode* root= nullptr;
   // your code here
int max(int a, int b)
  {
    return (a > b) ? a : b;
  }
  struct BNode* newNode(int item)
  {
    struct BNode* temp = new BNode();
    temp->data = item;
    temp->left = temp->right = NULL;
    return temp;
  }
  void inorder(struct BNode* root)
  {
    if (root != NULL) {
      inorder(root->left);
      printf("%d \n", root->data);
      inorder(root->right);
    }
```

```
    }
    void insert(int key) {
        root = insertNode(root, key);
    }
    BNode* insertNode(BNode* node, int key)
    {
        if (node == NULL)
            return newNode(key);
        if (key < node->data)
            node->left = insertNode(node->left, key);
        else if (key > node->data)
            node->right = insertNode(node->right, key);
        return node;
    }
    int findHeightUtil(BNode* root, int x,int& height)
    {
        if (root == NULL) {
            return -1;
        }
        int leftHeight = findHeightUtil(
            root->left, x, height);

        int rightHeight
            = findHeightUtil(
                root->right, x, height);
        int ans = max(leftHeight, rightHeight) + 1;
        if (root->data == x)
            height = ans;

        return ans;
    }
      //end your code
};
int search(BSTree * tree, int K)
{
      // your code here
int h = -1;
    int maxHeight = tree->findHeightUtil(tree->root, K, h);
    return h;
      //end your code
}
```

Câu hỏi 5

Hoàn thành

Đạt điểm 1,00

Xây dựng bảng Hash sử dụng phương pháp Seperate chaining

Note:

K: số lượng key trong bảng hash

N số lượng phần tử của mảng arr cần add vào bảng hash

```
#include <stdio.h>
struct Node {
    int data;
    Node* next;
};
struct HashTable {
    int K;
    Node* table;
    void construct(int arr[], int N) {
        // your code here
int key, i, j;
        table = new Node[K];
        for (i = 0; i < K; i++) {
            table[i].next = NULL;
            table[i].data = NULL;
        }
        for (i = 0; i < N; i++) {
            key = arr[i] % K;
            Node* temp = &table[key];
            while (temp->next != NULL)
            {
                temp = temp->next;
            }

            Node* newNode = new Node();
            newNode->data = arr[i];
            newNode->next = NULL;
            if (temp->data == NULL) {
                temp->data = arr[i];
                temp->next = NULL;
            }
            else
                temp->next = newNode;
        }
        // end your code
    }
};
```

Câu hỏi 6

Hoàn thành

Đạt điểm 1,00

Kết quả đoạn chương trình sau là gì

```
#include <stdio.h>
#include <iostream>
#include <queue>
#include <map>
#include <string>

std::map<std::string, int> price;

price["Xoai"] = 10;

while (price["Xoai"]) {
    price["Xoai"]--;
}

if (price.count("Xoai") {
    cout << "Hash Xoai";
}
else {
    cout << "No item";
}
```

a. Error

b. No item

c. Hash Xoai

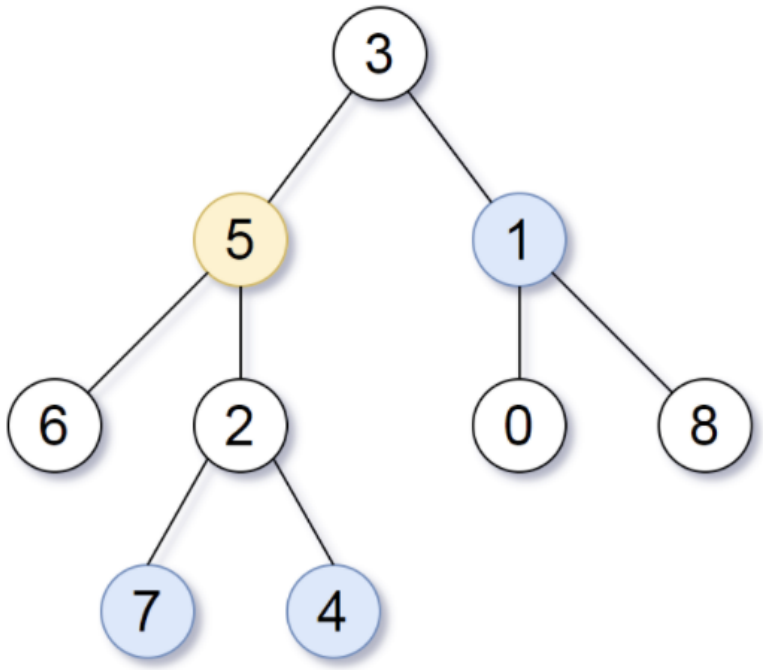
d. Không hiển thị nội dung gì

Câu hỏi 7

Hoàn thành

Đạt điểm 1,00

Cho cây nhị phân Tree, giá trị n, k. Hãy tìm giá trị của tất cả các node có khoảng cách tới node giá trị n là k



n=5, k=2

Output: 1,4,7

Giải thích: các node có giá trị 1,4,7 cách node giá trị 5 1 node

Giá trị N lưu số node tìm được (ví dụ: N=3)

```
#include <stdio.h>
struct BNode { //BST --> AVL
	int data;
	BNode* left, * right;
	// your code here
BNode(int data) {
		this->data = data;
		left = right = nullptr;
	}
	// end your code
};
struct Tree {
	BNode* root= nullptr;
	// your code here
int arr[1000] = { 0 };
	int count = 0;
	bool empty() {
		return root == nullptr;
```

```
}

void insert(int data, BNode*& node) {
	BNode* newNode = new BNode(data);
	if (node == nullptr) {
		node = newNode;
	}
	else {
		if (node->data > data)
			insert(data, node->left);
		else
			insert(data, node->right);
	}
}

void insert(int val) {
	insert(val, root);
}
void searchNodeDown(BNode* root, int k)
{
	if (root == NULL || k < 0)  return;
	if (k == 0)
	{
		arr[count++] = root->data;
		return;
	}
	searchNodeDown(root->left, k - 1);
	searchNodeDown(root->right, k - 1);
}
int searchNode(BNode* root, int target, int k)
{
	if (root == NULL) return -1;
	if (root->data == target)
	{
		searchNodeDown(root, k);
		return 0;
	}

	int temp1 = searchNode(root->left, target, k);
	if (temp1 != -1)
	{
		if (temp1 + 1 == k)
			arr[count++] = root->data;
		else
			searchNodeDown(root->right, k - temp1 - 2);
		return 1 + temp1;
	}
```

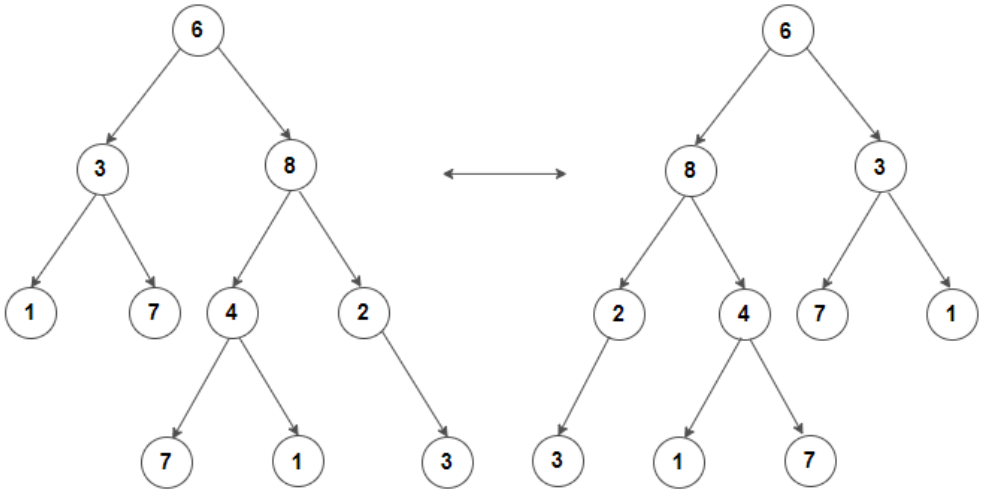
```
            int temp2 = searchNode(root->right, target, k);
            if (temp2 != -1)
            {
                if (temp2 + 1 == k)
                    arr[count++] = root->data;
                else
                    searchNodeDown(root->left, k - temp2 - 2);
                return 1 + temp2;
            }
            return -1;
        }
        //end your code
};
int* getDistance(Tree * tree, int n, int k, int &N)
{
    // your code here
tree->searchNode(tree->root, n, k);
     N = tree->count;
     int* result = &(tree->arr[0]);
     return result;
    //end your code
}
```

Câu hỏi 8

Hoàn thành

Đạt điểm 1,00

Cho cây nhị phân Tree. Hoán đổi nhánh trái với nhánh phải để có cây nhị phân mới




```
#include <stdio.h>
struct BNode {
	int data;
	BNode* left, * right;
	// your code here
void init() {
		left = right = nullptr;
	}
	// end your code
};
struct Tree {
	BNode* root= nullptr;
	// your code here
void init(BNode*& node) {
		node = new BNode;
		node->init();
	}
	void swapTree(struct BNode* node)

	{

		if (node == NULL) return;

		struct BNode* temp = node->left;
```

```
            node->left = node->right;

            node->right = temp;

            swapTree(node->left);

            swapTree(node->right);

        }
        void insert(int value, BNode*& parent) {
            if (parent == nullptr) {
                init(parent);
                parent->data = value;
                return;
            }
            if (value < parent->data) {
                insert(value, parent->left);

            }
            else {
                insert(value, parent->right);

            }
            return;
        }
        //end your code
};
Tree* swap(Tree * tree)
{
    // your code here
Tree t;

     t.swapTree(tree->root);

     return tree;
    //end your code
}
```

Câu hỏi 9

Hoàn thành

Đạt điểm 1,00

Xây dựng bảng Hash sử dụng phương pháp Quadratic Probing

Note:

K: số lượng key trong bảng hash

N số lượng phần tử của mảng arr cần add vào bảng hash

```
#include <stdio.h>
struct HashTable {
    int K;
    int* table;
    void construct(int arr[], int N) {
        // your code here
int key, key2, i,j;
        table = new int[K];
        for (i = 0; i < K; i++)
            table[i] = NULL;
        for (i = 0; i < N; i++)
        {
            j = 0;
            key = arr[i] % K;
            key2 = key;
            while (table[key] != NULL)
            {
                key = (key2 + j * j) % K;
                j++;
            }
            table[key] = arr[i];
        }
        // end your code
    }
};
```

◄ Nhóm câu hỏi 2

Chuyển tới...