CÂY NHỊ PHÂN TÌM KIẾM

MỤC TIÊU

Hoàn tất bài thực hành này, sinh viên có thể:

- Hiểu được các thành phần của cây nhị phân tìm kiếm.
- Thành thạo các thao tác trên cây nhị phân tìm kiếm: tạo cây, thêm phần tử, xóa phần tử, duyệt cây nhị phân tìm kiếm.
- Áp dụng cấu trúc dữ liệu cây nhị phân tìm kiếm vào việc giải quyết một số bài toán đơn giản.

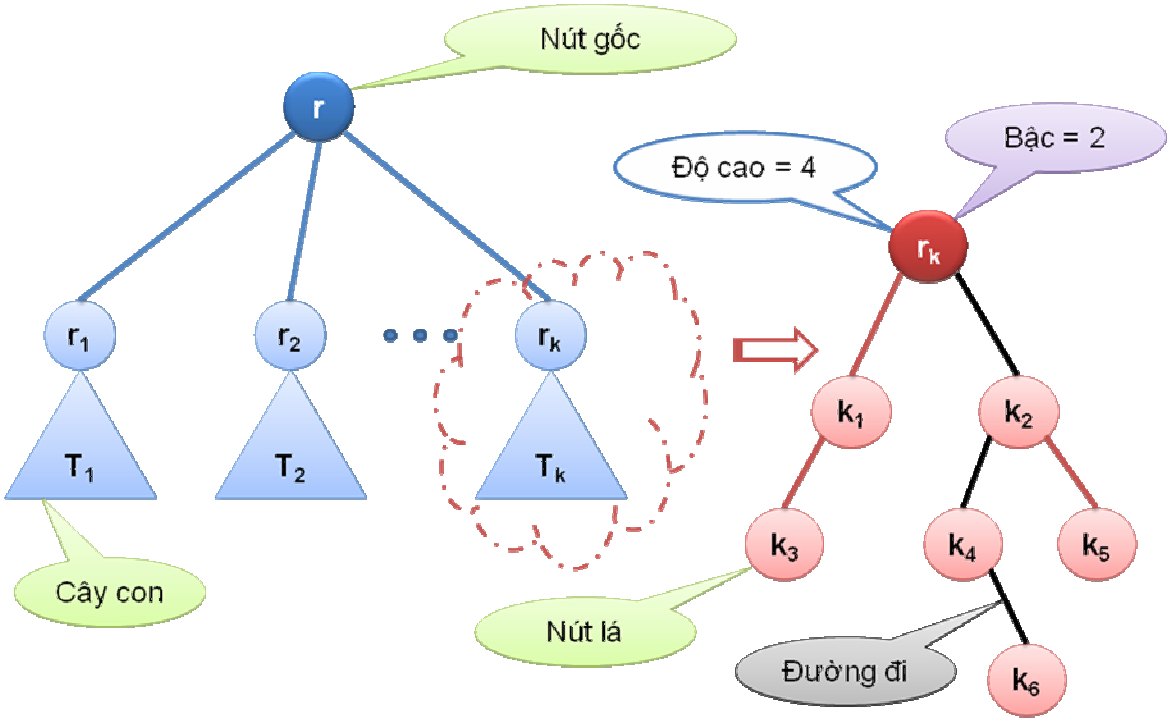
Thời gian thực hành: **từ 120 phút đến 400 phút**

TÓM TẮT

Cây nhị phân tìm kiếm là cây có tối đa 2 nhánh (cây con), nhánh trái và nhánh phải. Cây nhị phân tìm kiếm có các tính chất sau:

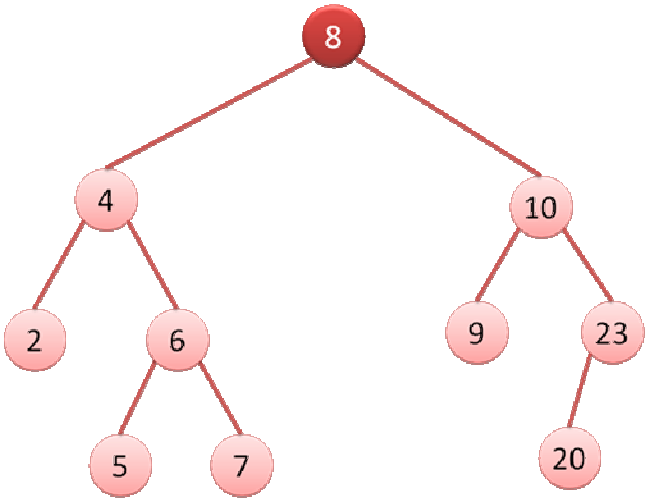
- Khóa của tất cả các nút thuộc cây con trái nhỏ hơn khóa nút gốc.
- Khóa của nút gốc nhỏ hơn khóa của tất cả các nút thuộc cây con phải.
- Cây con trái và cây con phải của nút gốc cũng là cây nhị phân tìm kiếm

Một số khái niệm:



- Nút lá có độ cao bằng 1

Ví dụ cây nhị phân tìm kiếm:



Trong mỗi nút của cây nhị phân tìm kiếm, thông tin liên kết là vô cùng quan trọng. Chỉ cần một xử lý không cẩn thận có thể làm mất phần liên kết này thì cây sẽ bị '**gãy**' cây con liên quan ứng với liên kết đó (không thể truy xuất tiếp tất cả các nút của nhánh con bị mất).

Các thao tác cơ bản trên cây nhị phân tìm kiếm:

- Thêm 1 nút: dựa vào tính chất của cây nhị phân tìm kiếm để tìm vị trí thêm nút mới.
    - Tạo cây: từ cây rỗng, lần lượt thêm các nút vào cây bằng phương thức thêm nút vào cây nhị phân tìm kiếm
- Xóa 1 nút: là nút lá, là nút có 1 nhánh con, là nút có 2 nhánh con.
- Duyệt cây nhị phân tìm kiếm: để có thể đi được hết các phần tử trên cây nhị phân tìm kiếm: duyệt trước (NLR), duyệt giữa (LNR), duyệt sau (LRN). Do tính chất của cây nhị phân tìm kiếm, phép duyệt giữa cho phép duyệt các khóa của cây theo thứ tự tăng dần

NỘI DUNG THỰC HÀNH

Cơ bản

Sinh viên đọc kỹ phát biểu bài tập và thực hiện theo hướng dẫn:

Tổ chức một cây nhị phân tìm kiếm trong đó mỗi phần tử chứa thông tin dữ liệu là số nguyên.

*Người dùng sẽ nhập các giá trị nguyên từ bàn phím. Với mỗi giá trị nguyên được nhập vào, giá trị đó được thêm vào cây nhị phân tìm kiếm mà vẫn đảm bảo cây sau khi thêm vẫn là cây nhị phân tìm kiếm. Nếu người dùng nhập vào giá trị **-1**, quá trình nhập dữ liệu sẽ kết thúc. Cây ban đầu là cây rỗng (chưa có nút nào)*

Sau đó, in ra các phần tử đang có trên cây bằng phương pháp duyệt trước.

Cho người dùng nhập vào 1 giá trị nguyên từ bàn phím, cho biết giá trị này có trong cây hay không. Nếu có, cho biết nút đó có độ cao bao nhiêu. Sau đó, xóa nút khỏi cây, xuất cây sau khi xóa bằng phương pháp duyệt trước

Phân tích

- Cây nhị phân tìm kiếm có mỗi nút chứa dữ liệu nguyên. Thông tin của mỗi nút được khai báo theo ngôn ngữ C/C++ như sau:

```
struct NODE{
	int Key;
	NODE *pLeft;
	NODE *pRight;
};
```

- Thao tác cần thực hiện:
  - Khai báo, khởi tạo cây
  - (lặp) **thêm** nút có khóa nguyên vào **cây nhị phân tìm kiếm** (**Insert**),
  - **in** các nút của cây nhị phân tìm kiếm (**NLR**),
  - **tìm** 1 giá trị, nếu có:
    - tính **độ cao** của nút đó (**Height**)
    - **xóa** nút khỏi cây (**RemoveNode**)
    - **in** các nút của cây sau khi xóa (**NLR**)

Chương trình mẫu

```
#include "stdio.h"

struct NODE{
	int Key;
	NODE *pLeft;
	NODE *pRight;
};

void Init(NODE *&TREE)
{
	TREE = NULL;
}

void  Insert (NODE *&pRoot, int x)
{
	if (pRoot == NULL)
```

```
	{
		NODE *q;
		q = new NODE;
		q->Key = x;
		q->pLeft = q->pRight = NULL;
		pRoot = q;
	}
	else
	{
		if (x < pRoot->Key)
			Insert (pRoot->pLeft, x);
		else if (x > pRoot->Key)
			Insert (pRoot->pRight, x);
	}
}

void CreateTree(NODE *&pRoot)
{
	int Data;
	do{
		printf("Nhap vao du lieu, -1 de ket thuc: ");
		scanf("%d", &Data);
		if (Data == -1)
			break;
		Insert(pRoot, Data);
	} while(1);
}

void NLR(NODE* pTree)
{
	if(pTree != NULL)
	{
		printf("%4d", pTree->Key);
		NLR(pTree->pLeft);
		NLR(pTree->pRight);
	}
}

NODE* Search(NODE* pRoot, int x)
{
	if(pRoot == NULL)
		return NULL;
	if(x < pRoot->Key)
		Search(pRoot->pLeft, x);
	else
		if(x > pRoot->Key)
			Search(pRoot->pRight, x);
		else
		{
			//Ghi chú: Trong trường hợp nào dòng bên dưới được thực hiện?
			return pRoot;
		}
}

int Height(NODE* pNode)
{
	if(pNode == NULL)
		return 0;
	int HL, HR;
	HL = Height(pNode->pLeft);
	HR = Height(pNode->pRight);
	if(HL > HR)
		return (1 + HL);
	return (1 + HR);
```

```
}

void SearchStandFor(NODE* &Tree, NODE* &q)
{
	if (Tree->pRight)
		SearchStandFor(Tree->pRight,q);
	else
	{
		q->Key = Tree->Key;
		q = Tree;
		Tree = Tree->pLeft;
	}
}

void RemoveNode(NODE* &Tree, int x)
{
	NODE* p;
	if(Tree == NULL)
		printf("%d khong co trong cay", x);
	else
	{
		if (x < Tree->Key)
			RemoveNode(Tree->pLeft,x);
		else
			if (x > Tree->Key)
				RemoveNode(Tree->pRight,x);
			else
			{
				//Ghi chú: Mục đích phép gán này là gì?
				p = Tree;
				if(p->pRight == NULL)
					Tree = p->pLeft;
				else
					if (p->pLeft == NULL)
						Tree = p->pRight;
					else {
						//Ghi chú: Hàm bên dưới dùng để làm gì?
						SearchStandFor(Tree->pLeft,p);
					}
				delete p;
			}
	}
}

void main()
{
	NODE* pTree, *p;
	int x;

	Init(pTree);
	CreateTree(pTree);
	NLR(pTree);

	printf("Nhap vao 1 gia tri de tim: ");
	scanf("%d", &x);
	p = Search(pTree, x);
	if(p != NULL)
	{
		printf ("%d co xuat hien trong cay.\n", x);
		printf("Chieu cao cua nut %d la %d\n", x, Height(p));
		RemoveNode(pTree, x);
		NLR(pTree);
	}
	else
```

```
            printf("%d khong co trong cay.\n", x);
}
```

Yêu cầu

1. Biên dịch đoạn chương trình nêu trên.
2. Cho biết kết quả in ra màn hình khi người dùng nhập vào các dữ liệu sau:

-1

-1

---

5 -1

-1

---

7 10 -23 -25 -4 1 -1

-23

-23 7 10 -25 -4 1 -1

-23

---

7 10 -23 -4 1 -25 -1

-23

---

1 2 3 4 -1 5

3

3. Nêu nhận xét ngắn gọn mối liên hệ giữa thứ tự nhập dữ liệu vào (với cùng tập dữ liệu – dữ liệu thứ 3, 4, và 5) với thứ tự in dữ liệu ra màn hình.
4. Vẽ hình cây nhị phân tìm kiếm theo dữ liệu được nhập ở câu 2.
5. Nếu bỏ `Init(pTree)` trong hàm `main` kết quả có thay đổi hay không? Giải thích lý do?
6. Nếu trong hàm `CreateTree` vòng lặp `do…while` được thay đổi như dưới đây thì kết quả kết xuất ra màn hình sẽ như thế nào đối với dữ liệu câu 2? Giải thích lý do?

```
      do
      {
            printf("Nhap vao du lieu, -1 de ket thuc: ");
            scanf("%d", &Data);
            Insert(pRoot, Data);
            if (Data == -1)
                  break;
      }while (1);
```

7. Trong hàm `NLR` nếu ta đổi trật tự như bên dưới thì kết quả thế nào?

```
void NLR(NODE* pTree)
{
      if(pTree != NULL)
      {
            NLR(pTree->pLeft);
            printf("%4d", pTree->Key);
            NLR(pTree->pRight);
```

```
        }
}
```

8. Hãy ghi chú các thông tin bằng cách trả lời các câu hỏi ứng với các dòng lệnh có yêu cầu ghi chú (`//Ghi chú`) trong các hàm `Search`, `RemoveNode`.

9. Nếu trong hàm `RemoveNode` thay đổi như sau, kết quả có thay đổi không? Nếu có, chỉ ra cách để kết quả không thay đổi. Nếu không, giải thích lý do

```
                                else {
                                        //Ghi chú: Hàm bên dưới dùng để làm gì?
                                        SearchStandFor(Tree->pRight,p);
                                }
```

10. Trong hàm `RemoveNode` nếu không có dòng `delete p;` thì kết quả có gì khác? Dòng đó dùng để làm gì.

Áp dụng – Nâng cao

1. Bổ sung chương trình mẫu cho phép tính **tổng giá trị** các nút trên cây nhị phân gồm các giá trị nguyên.

Gợi ý: tham khảo hàm **`NLR`** để viết hàm **`SumTree`**.

2. Bổ sung chương trình mẫu cho phép tìm **giá trị nguyên lớn nhất và nhỏ nhất** trong số các phần tử nguyên trên cây nhị phân tìm kiếm gồm các giá trị nguyên.

Gợi ý: dựa vào tính chất 1, 2 của cây nhị phân tìm kiếm.

3. Bổ sung chương trình mẫu cho phép tính **số lượng các nút** của cây nhị phân gồm các giá trị nguyên.

Gợi ý: tham khảo hàm **`NLR`** để viết hàm **`CountNode`**.

4. Bổ sung chương trình mẫu cho biết **số lượng các nút lá** trên cây nhị phân.

Gợi ý: tham khảo thao tác duyệt cây nhị phân **`NLR`**.

5. Sử dụng cây nhị phân tìm kiếm để giải bài toán:

   a. Đếm có bao nhiêu giá trị phân biệt trong dãy số cho trước

   b. Với mỗi giá trị phân biệt, cho biết số lượng phần tử

BÀI TẬP THÊM

1. Sử dụng cây nhị phân tìm kiếm để giải bài toán đếm (thống kê) số lượng ký tự có trong văn bản (Không dấu).

   a. Xây dựng cây cho biết mỗi ký tự có trong văn bản xuất hiện mấy lần

   b. Nhập vào 1 ký tự. Kiểm tra ký tự đó xuất hiện bao nhiêu lần trong văn bản

2. Bài toán tương tự như trên nhưng thống kê số lượng tiếng có trong văn bản (không dấu)

   Ví dụ:

   Văn bản có nội dung như sau: “hoc sinh di hoc mon sinh hoc”

   Kết quả cho thấy như sau:

   di: 1

   hoc: 3

   mon: 1

sinh: 2