Nội dung

- Cây nhị phân tổng quát và một số thao tác
    - Thêm, xóa
    - Tìm kiếm
    - Duyệt cây: thứ tự trước, thứ tự sau, thứ tự giữa

==========

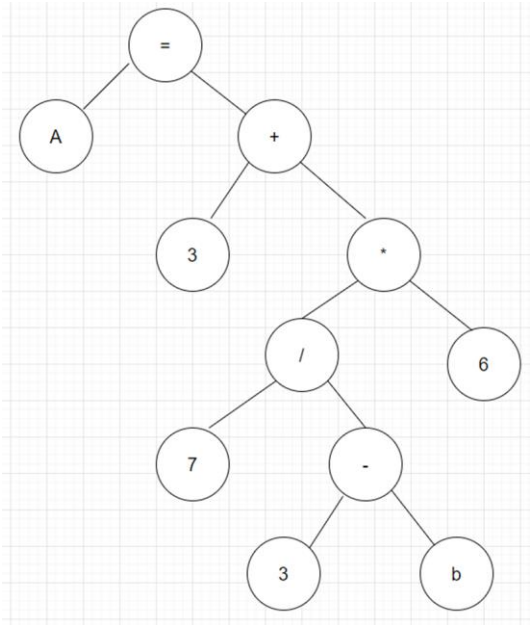
Lý thuyết về cây nhị phân

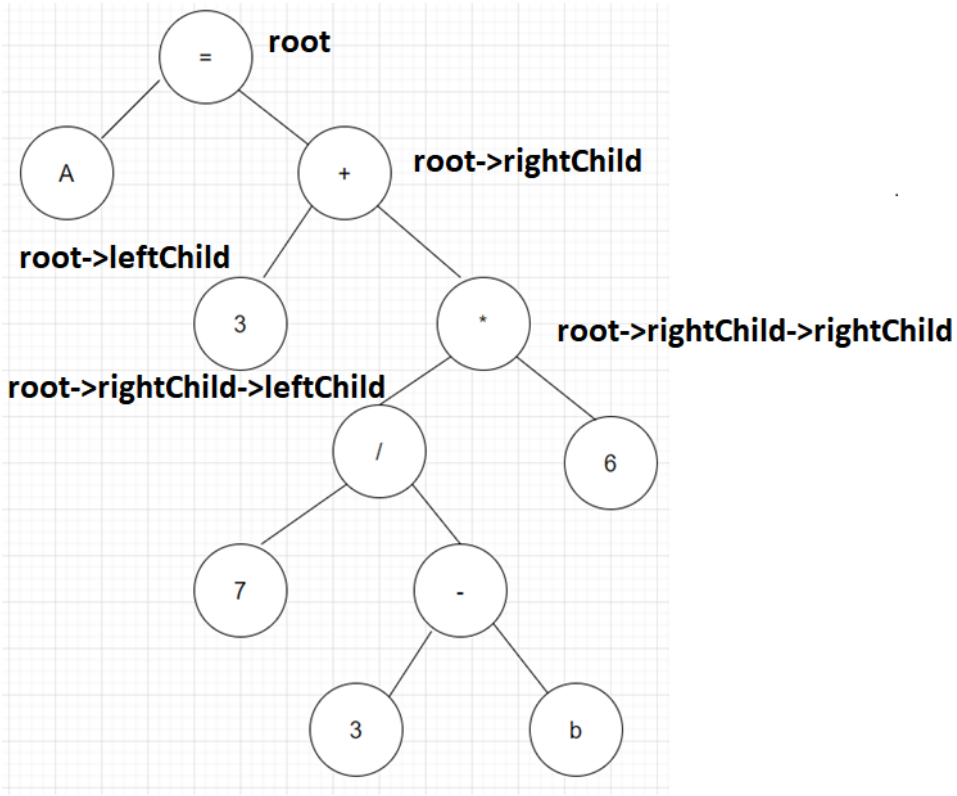
- 1 nút có tối đa 2 con
    - Con trái và con phải

Các thao tác cơ bản đối với cây nhị phân tổng quát

- Tạo cây mới
- Thêm nút vào cây
- Xóa nút
- Duyệt cây: theo thứ tự
- Tính chiều cao/độ sâu
- In ra các nút ở vị trí đặc biệt
- Kiểm tra cây có một so đặc tính đặc biệt: VD. Cây đối xứng,…

Vì cây có hình dáng tùy vào cách thêm, người dùng sẽ cần chỉ rõ ra vị trí thêm (vào con trái hay con phải của nút hiện tại)



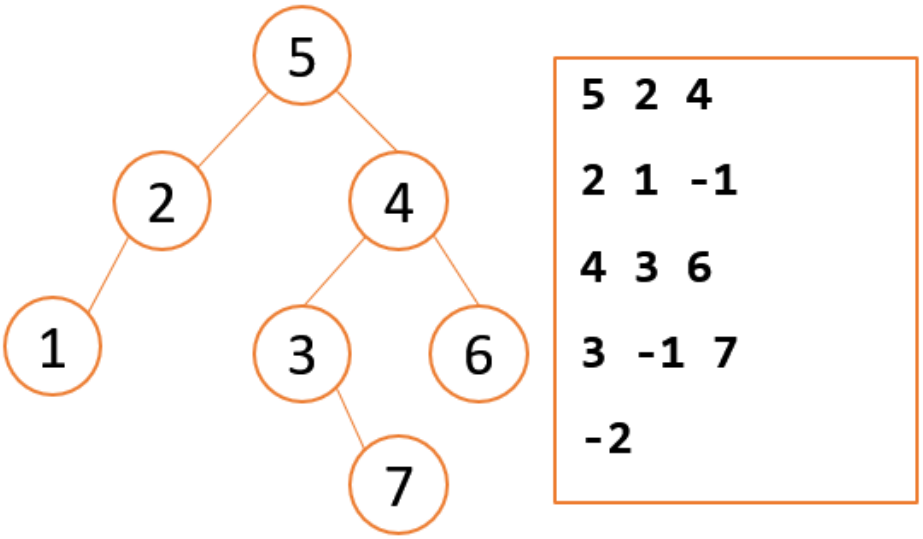



```
#include <stdio.h>
#include <stdlib.h>
#include <queue>
using namespace std;
typedef struct Node{
   char label; // identifier of the node
   struct Node* leftChild;// pointer to the left child
   struct Node* rightChild;// pointer to the right child
}Node;

// ham tao nut moi
// cap phat bo nho dong cho nut moi
Node* makeNode(char label)
{
   Node* newNode = (Node*)malloc(sizeof(Node));
   newNode->label = label;
   newNode->leftChild =NULL;
   newNode->rightChild =NULL;
   return newNode;
}

// them nut moi co nhan la Label thanh con trai/con phai cua nut hien tai
// isLeftChild = 1 --> them vao con trai
```

```
// isLeftChild = 0 --> them vao con phai
// nut root phai khac NULL
void insertNode(Node* root, char label, int isLeftChild)
{
   Node* newNode = makeNode(label);
   if(isLeftChild) root->leftChild = newNode;
   else root->rightChild = newNode;
}

// ham duyet cay theo muc de in ra cac nut lan luot theo muc
// duyet cay theo muc
void levelTraversal(Node* root)
{
if (NULL == root) return;
queue<Node*> Q;
Q.push(root);

while (Q.size() > 0)
{
Node* p = Q.front();
Q.pop();
printf("%c, ", p->label);
if (NULL != p->leftChild) Q.push(p->leftChild);
if (NULL != p->rightChild) Q.push(p->rightChild);
}
printf("\n");
}

// duyet theo thu tu giua
void inorderTraversal(Node* root)
{
   if(NULL==root) return;
   inorderTraversal(root->leftChild);
   printf("%c, ", root->label);
   inorderTraversal(root->rightChild);
}
int main()
{
   Node* root = makeNode('=');
   insertNode(root,'A',1);
   insertNode(root,'+',0);
   insertNode(root->rightChild,'3',1);
   insertNode(root->rightChild,'*',0);
   insertNode(root->rightChild->rightChild,'/',1);
   insertNode(root->rightChild->rightChild,'6',0);
   insertNode(root->rightChild->rightChild->leftChild,'7',1);
   insertNode(root->rightChild->rightChild->leftChild,'-',0);
   insertNode(root->rightChild->rightChild->leftChild->rightChild,'3',1);
```

```
    insertNode(root->rightChild->rightChild->leftChild->rightChild,'b',0);
    levelTraversal(root);
    printf("\n");
    inorderTraversal(root);
    printf("\n");
    return 0;
}
```

Bài tập 1.sửa lại để có thể đọc cây nhị phân từ file văn bản
Sử dụng lại hàm đã có + thêm hàm find



Dòng đầu tiên là gốc

Mỗi dòng nhãn đầu tiên sẽ là gốc, và các nhãn tiếp theo lần lượt là con trái và con phải

Nếu con trái rỗng thì giá trị tương ứng sẽ là -1

Kết thúc là giá trị -2

Nhãn của các nút là giá trị >0

```
// tim va tra ve nut co nhan la label
Node* find(Node* root, char label)
{
    if(NULL==root) return NULL;
    queue<Node*> Q;
Q.push(root);

while (Q.size() > 0)
{
Node* p = Q.front();
Q.pop();
```

```
if(label==p->label) return p;
if (NULL != p->leftChild) Q.push(p->leftChild);
if (NULL != p->rightChild) Q.push(p->rightChild);
}
return NULL;
}
```

CODE

```
#include <stdio.h>
#include <stdlib.h>
#include <queue>
using namespace std;
typedef struct Node{
   int label; // identifier of the node
   struct Node* leftChild;// pointer to the left child
   struct Node* rightChild;// pointer to the right child
}Node;

// ham tao nut moi
// cap phat bo nho dong cho nut moi
Node* makeNode(int label)
{
   Node* newNode = (Node*)malloc(sizeof(Node));
   newNode->label = label;
   newNode->leftChild =NULL;
   newNode->rightChild =NULL;
   return newNode;
}

// them nut moi co nhan la Label thanh con trai/con phai cua nut hien tai
// isLeftChild = 1 --> them vao con trai
// isLeftChild = 0 --> them vao con phai
// nut root phai khac NULL
void insertNode(Node* root, int label, int isLeftChild)
{
   Node* newNode = makeNode(label);
   if(isLeftChild) root->leftChild = newNode;
   else root->rightChild = newNode;
}

// ham duyet cay theo muc de in ra cac nut lan luot theo muc
// duyet cay theo muc
void levelTraversal(Node* root)
{
if (NULL == root) return;
queue<Node*> Q;
```

```
Q.push(root);

while (Q.size() > 0)
{
Node* p = Q.front();
Q.pop();
printf("%d, ", p->label);
if (NULL != p->leftChild) Q.push(p->leftChild);
if (NULL != p->rightChild) Q.push(p->rightChild);
}
printf("\n");
}



// duyet theo thu tu giua
void inorderTraversal(Node* root)
{
   if(NULL==root) return;
   inorderTraversal(root->leftChild);
   printf("%d, ", root->label);
   inorderTraversal(root->rightChild);
}

// tim va tra ve nut co nhan la label
Node* findLabel(Node* root, int label)
{
   if(NULL==root) return NULL;
   queue<Node*> Q;
Q.push(root);

while (Q.size() > 0)
{
Node* p = Q.front();
Q.pop();
if(label==p->label) return p;
if (NULL != p->leftChild) Q.push(p->leftChild);
if (NULL != p->rightChild) Q.push(p->rightChild);
}
return NULL;
}

// nap cay moi tu file
Node *load(const char* filename){
   Node *root = NULL, *p;
   FILE* f = fopen(filename,"r");
   if (f==NULL) {
      printf("Khong mo duoc file");
```

```
        return 0;
    }
    while(1)
    {
        int u;
        fscanf(f,"%d",&u);
        if(u == -2) break;// termination
        if(root == NULL) root = makeNode(u);// create the root
        int l,r;
        p = findLabel(root,u);
        fscanf(f,"%d%d",&l,&r);
        if(l > -1) insertNode(p,l,1);
        if(r > -1) insertNode(p,r,0);
    }

    fclose(f);
    return root;
}
int main()
{
    /*
    Node* root = makeNode('=');
    insertNode(root,'A',1);
    insertNode(root,'+',0);
    insertNode(root->rightChild,'3',1);
    insertNode(root->rightChild,'*',0);
    insertNode(root->rightChild->rightChild,'/',1);
    insertNode(root->rightChild->rightChild,'6',0);
    insertNode(root->rightChild->rightChild->leftChild,'7',1);
    insertNode(root->rightChild->rightChild->leftChild,'-',0);
    insertNode(root->rightChild->rightChild->leftChild->rightChild,'3',1);
    insertNode(root->rightChild->rightChild->leftChild->rightChild,'b',0);
    */
    Node * root = load("tree.txt");
    levelTraversal(root);
    printf("\n");
    inorderTraversal(root);
    printf("\n");
    return 0;
}
```

LOAD file ký tự

```
#include <stdio.h>
#include <stdlib.h>
#include <queue>
```

```
using namespace std;
typedef struct Node{
   char label; // identifier of the node
   struct Node* leftChild;// pointer to the left child
   struct Node* rightChild;// pointer to the right child
}Node;

// ham tao nut moi
// cap phat bo nho dong cho nut moi
Node* makeNode(char label)
{
   Node* newNode = (Node*)malloc(sizeof(Node));
   newNode->label = label;
   newNode->leftChild =NULL;
   newNode->rightChild =NULL;
   return newNode;
}

// them nut moi co nhan la Label thanh con trai/con phai cua nut hien tai
// isLeftChild = 1 --> them vao con trai
// isLeftChild = 0 --> them vao con phai
// nut root phai khac NULL
void insertNode(Node* root, char label, int isLeftChild)
{
   Node* newNode = makeNode(label);
   if(isLeftChild) root->leftChild = newNode;
   else root->rightChild = newNode;
}

// ham duyet cay theo muc de in ra cac nut lan luot theo muc
// duyet cay theo muc
void levelTraversal(Node* root)
{
if (NULL == root) return;
queue<Node*> Q;
Q.push(root);

while (Q.size() > 0)
{
Node* p = Q.front();
Q.pop();
printf("%c, ", p->label);
if (NULL != p->leftChild) Q.push(p->leftChild);
if (NULL != p->rightChild) Q.push(p->rightChild);
}
printf("\n");
}
```

```
// duyet theo thu tu giua
void inorderTraversal(Node* root)
{
   if(NULL==root) return;
   inorderTraversal(root->leftChild);
   printf("%c, ", root->label);
   inorderTraversal(root->rightChild);
}

// tim va tra ve nut co nhan la label
Node* findLabel(Node* root, char label)
{
   if(NULL==root) return NULL;
   queue<Node*> Q;
Q.push(root);

while (Q.size() > 0)
{
Node* p = Q.front();
Q.pop();
if(label==p->label) return p;
if (NULL != p->leftChild) Q.push(p->leftChild);
if (NULL != p->rightChild) Q.push(p->rightChild);
}
return NULL;
}

// nap cay moi tu file
Node *load(const char* filename){
   Node *root = NULL, *p;
   FILE* f = fopen(filename,"r");
   if (f==NULL) {
      printf("Khong mo duoc file");
      return 0;
   }
   char u;
   while(1)
   {
      fscanf(f,"%c",&u);
      // bo ky tu xuong dong trong file
      if(u=='\r' ||u=='\n') continue;
      if(u == '$') break;// termination
      if(root == NULL) root = makeNode(u);// create the root
      char l,r;
      p = findLabel(root,u);
      fscanf(f,"%c%c",&l,&r);
```

```
    if(l !='#') insertNode(p,l,1);
    if(r != '#') insertNode(p,r,0);
  }

  fclose(f);
  return root;
}
int main()
{
  /*
  Node* root = makeNode('=');
  insertNode(root,'A',1);
  insertNode(root,'+',0);
  insertNode(root->rightChild,'3',1);
  insertNode(root->rightChild,'*',0);
  insertNode(root->rightChild->rightChild,'/',1);
  insertNode(root->rightChild->rightChild,'6',0);
  insertNode(root->rightChild->rightChild->leftChild,'7',1);
  insertNode(root->rightChild->rightChild->leftChild,'-',0);
  insertNode(root->rightChild->rightChild->leftChild->rightChild,'3',1);
  insertNode(root->rightChild->rightChild->leftChild->rightChild,'b',0);
  */
  Node * root = load("tree2.txt");
  levelTraversal(root);
  printf("\n");
  inorderTraversal(root);
  printf("\n");
  return 0;
}
```

Bài tập 2. tìm và in ra các nút lá ở mức nông nhất trên cây

Nút lá ở mức nông nhất?

Là nút lá đầu tiên khi **duyệt theo mức**

Vì khi duyệt theo mức ta sẽ đi lần lượt từ mức 0 (gốc) tới mức 1 rồi mức 2 --> gặp nút lá đầu tiên thì nó sẽ là nút lá nông nhất!

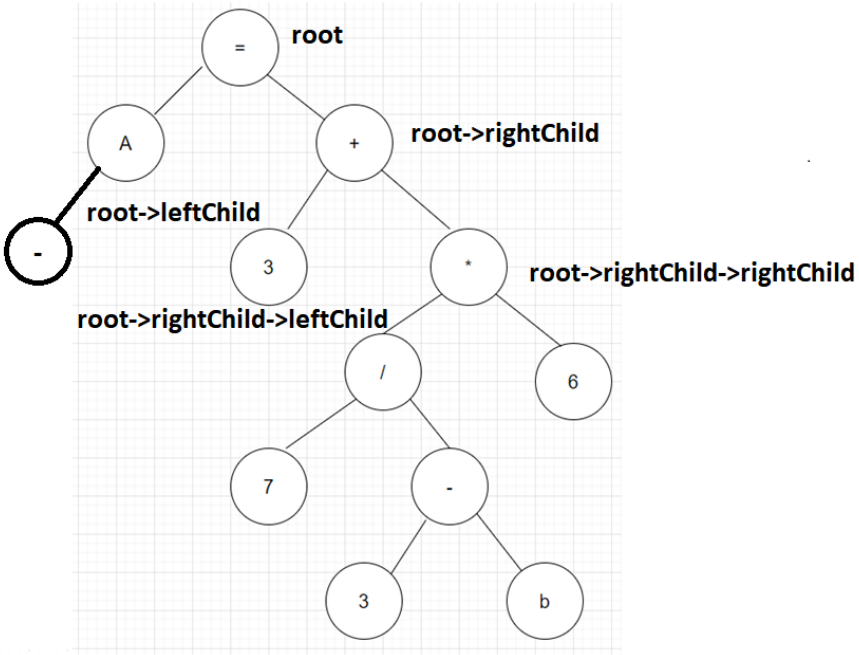
Nếu chỉ cần in ra 1 nút lá nông nhất --> sửa hàm duyệt theo mức là xong

```
Node* shallowestLeaf(Node* root)
{
if (NULL == root) return NULL;
queue<Node*> Q;
Q.push(root);
```

```
while (Q.size() > 0)
{
Node* p = Q.front();
Q.pop();
if(p->leftChild==NULL && p->rightChild==NULL) return p;
if (NULL != p->leftChild) Q.push(p->leftChild);
if (NULL != p->rightChild) Q.push(p->rightChild);
}
}
```

In ra tất cả các nút lá nông nhất?

- Tìm mức nông nhất
- In ra tất cả các nút lá ở mức này




```
=A+
+3*
*/6
/7-
-3b
A-#
$
```

```
void printShallowestLeaves(Node* root)
{
if (NULL == root) return;
queue<Node*> Q;
Q.push(root);
```

```
            int check =0;
while (Q.size() > 0)
{
Node* p = Q.front();
Q.pop();
if(p->leftChild==NULL && p->rightChild==NULL)
{
   printf("Nut la %c\n",p->label);
   check =1; // Khong lay them cac nut o muc sau hon nua,
                                    // sau khi ma tim duoc nut la o muc nong nhat
}
if(check==1) continue;
if (NULL != p->leftChild) Q.push(p->leftChild);
if (NULL != p->rightChild) Q.push(p->rightChild);
}
}
```

Nút lá sâu nhất?

- Nếu **duyệt theo mức** thì nó là nút được duyệt sau dùng khi duyệt theo mức

Có 2 hình thức duyệt nút

- Duyệt theo mức (duyệt theo chiều rộng): đi lần lượt các mức
- Duyệt theo chiều sâu (ưu tiên đi 1 nhánh - thường là nhánh trái trước): thứ tự trước, giữa và sau

Nếu chỉ cần in ra 1 nút lá ở mức sâu nhất --> đây chính là nút lá cuối cùng khi duyệt theo mức

Nếu cần in ra **tất cả các nút lá** ở mức sâu nhất thì làm thế nào?

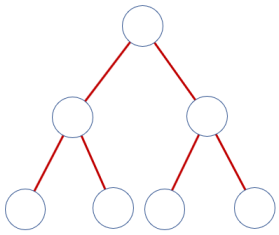
Cách 1. Cần 2 lần duyệt

- Tìm chiều cao/độ sâu của cây
- Duyệt và in ra tất cả các nút lá ở mức h này

Cách 2. 1 Lần duyệt liệu có làm được?

Bài tập 3. kiểm tra 1 cây nhị phân có phải là cây đầy đủ
cây có chiều cao h thì sẽ có số nút là $2^{h+1}-1$

Các nút lá cùng mức, mỗi mức có đủ số lượng nút



Ý tưởng

- Tính chiều cao h
- Đếm số nút n
- Kiểm tra xem n == 2^{h+1}-1

Bài tập về nhà 2. lưu cây nhị phân (cây biểu thức) vào file

Mỗi nút có giá trị kiểu char

Toán tử và toán hạng chỉ là ký tự

Cây trái/phải mà NULL thì tương ứng với ký hiệu #

Kết thúc file tương ứng với ký hiệu $

Tạo ra 1 cây biểu thức và lưu cây đó vào file

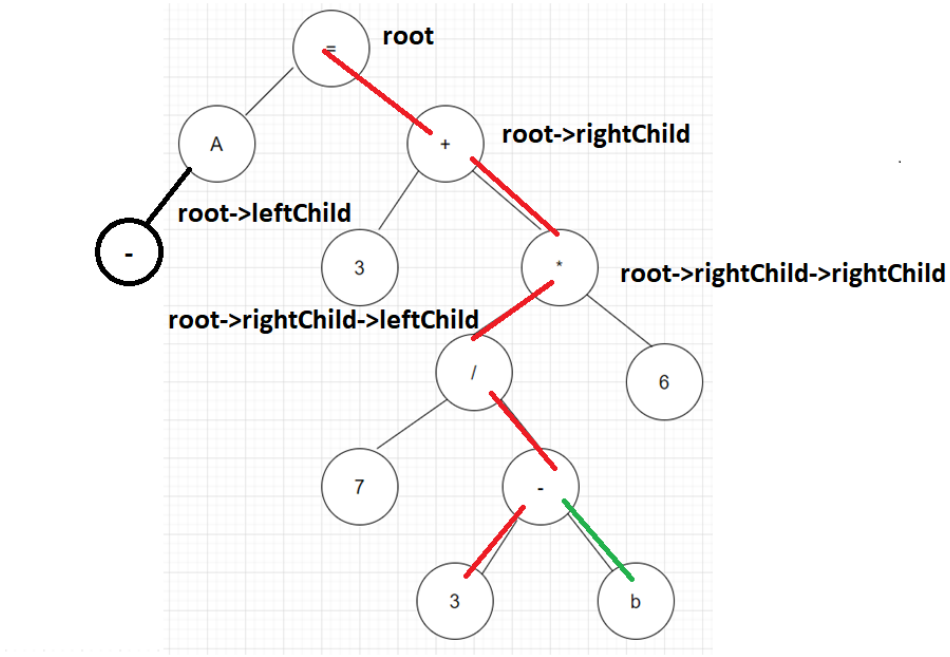
Cây có thể được nhập từ bàn phím hoặc fixed code

```
Node* root = makeNode('=');
insertNode(root,'A',1);
insertNode(root,'+',0);
insertNode(root->rightChild,'3',1);
insertNode(root->rightChild,'*',0);
insertNode(root->rightChild->rightChild,'/',1);
insertNode(root->rightChild->rightChild,'6',0);
insertNode(root->rightChild->rightChild->leftChild,'7',1);
insertNode(root->rightChild->rightChild->leftChild,'-',0);
insertNode(root->rightChild->rightChild->leftChild->rightChild,'3',1);
insertNode(root->rightChild->rightChild->leftChild->rightChild,'b',0);
```

```
=A+
+3*
*/6
/7-
-3b
A-#
$
```

Bài tập về nhà 3. Tính giá trị của cây biểu thức

Bài tập về nhà 4. in ra 1 đường đi dài nhất từ gốc tới lá


=
root
A
+
root->rightChild
-
root->leftChild
3
*
root->rightChild->rightChild
root->rightChild->leftChild
/
6
7
-
3
b