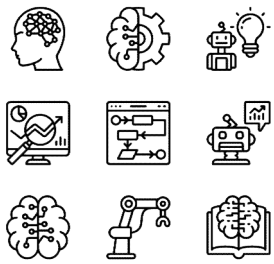
Computer Science for Practicing Engineers

Cây AVL



TS. Huỳnh Bá Diệu
Email: dieuhb@gmail.com
Phone: 0914146868

Cây AVL

Nội dung

1. Các tính chất trên cây AVL
2. Các phép xoay trên cây AVL
3. Chèn một nốt vào cây AVL
4. Xoá một nốt từ cây AVL
5. Một số ứng dụng của cây nhị phân

2

Cây AVL [**A**delson – **V**elskii] – [**L**andis]

Cây AVL là cây nhị phân tìm kiếm tự cân bằng

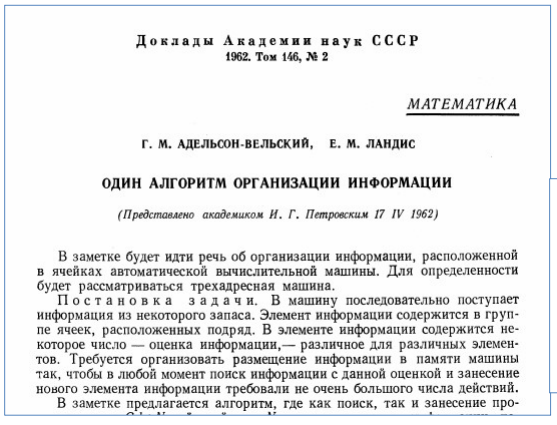
Georgy Maximovich Adelson-Velsky

January 8, 1922 - April 26, 2014

Evgenii Mikhailovich Landis

October 6, 1921 – December 12, 1997

Cây AVL [**A**delson – **V**elskii] – [**L**andis]



Доклады Академии наук СССР
1962. Том 146, № 2

МАТЕМАТИКА

Г. М. АДЕЛЬСОН-ВЕЛЬСКИЙ, Е. М. ЛАНДИС

ОДИН АЛГОРИТМ ОРГАНИЗАЦИИ ИНФОРМАЦИИ

(Представлено академиком И. Г. Петровским 17 IV 1962)

В заметке будет идти речь об организации информации, расположенной в ячейках автоматической вычислительной машины. Для определенности будет рассматриваться трехадресная машина.

Постановка задачи. В машину последовательно поступает информация из некоторого запаса. Элемент информации содержится в группе ячеек, расположенных подряд. В элементе информации содержится некоторое число — оценка информации,— различное для различных элементов. Требуется организовать размещение информации в памяти машины так, чтобы в любой момент поиск информации с данной оценкой и занесение нового элемента информации требовали не очень большого числа действий.

В заметке предлагается алгоритм, где как поиск, так и занесение про-

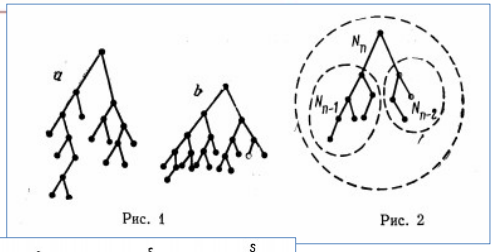


Рис. 1 Рис. 2

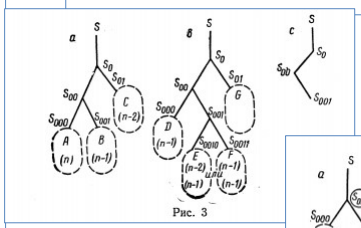


Рис. 3

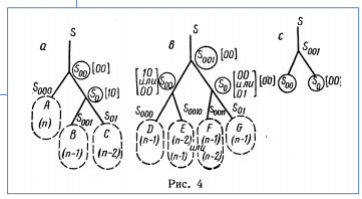


Рис. 4

Cây AVL [**A**delson – **V**elskii] – [**L**andis]

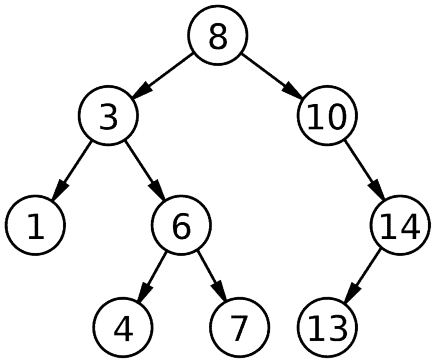
+ Nếu tìm kiếm trên cây BST thì độ phức tạp là O(h), trong đó h là chiều cao của cây. Trong trường hợp cây là cây lệch thì độ phức tạp là O(n).

+ Cây AVL là cây nhị phân tìm kiếm nhưng tự cân bằng nên khi tìm kiếm trên cây thì độ phức tạp luôn là O(h), với h là chiều cao và bằng $\log_2(n)$.

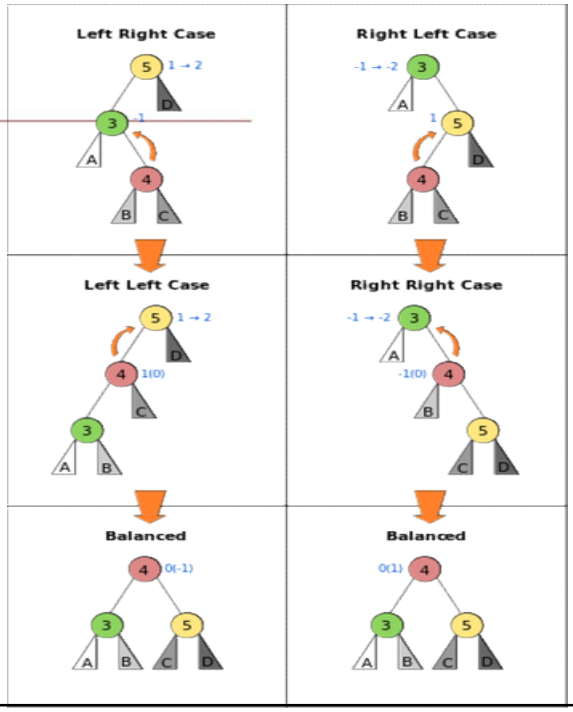
Cây cân bằng: là cây mà mọi cây con thuộc nó đều thỏa mãn tính chất là chiều cao cây con trái và cây con phải không lệch nhau quá 1.

Cây AVL

- Gọi h1 là chiều cao của cây con trái
  - h11 là chiều cao con trái của con trái
  - h12 là chiều cao con phải của con trái
- Gọi h2 là chiều cao của cây con phải
  - h21 là chiều cao con trái của con phải
  - h22 là chiều cao con phải của con phải



Cây AVL



Cây AVL [**A**delson – **V**elskii] – [**L**andis]

Cây AVL sẽ thực hiện 4 phép quay (xoay) để biến cây không cân bằng thành cây cân bằng.

Quay nhánh trái *(khi chiều cao con trái lớn hơn)* : **h1>h2+1**

+ Quay con trái **qL** *(khi chiều cao con trái của con trái lớn hơn chiều cao con phải của con trái)* ***(h11>h12)***

+ Quay con phải của con trái **qLR** *(khi chiều cao con phải của con trái lớn hơn chiều cao con trái của con trái)* ***(h12>h11)***

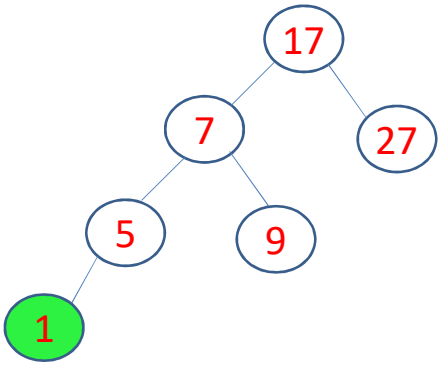
Quay nhánh phải *(khi chiều cao con phải lớn hơn)* : **h2>h1+1**

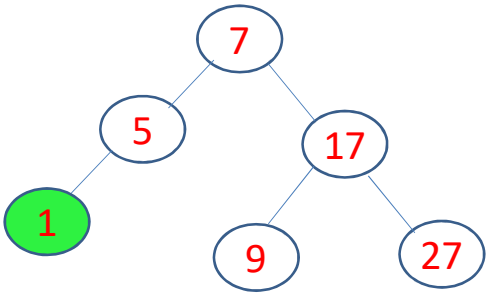
+ Quay con phải **qR** *(h22> h21)*

+ Quay con trái của con phải **qRL** *(h21>h22)*

Phép quay con trái trên cây AVL

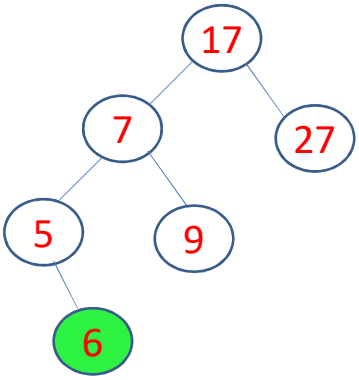
Chèn giá trị 1 vào cây thì cây không cân bằng➔ thực hiện quay trái





Cây AVL

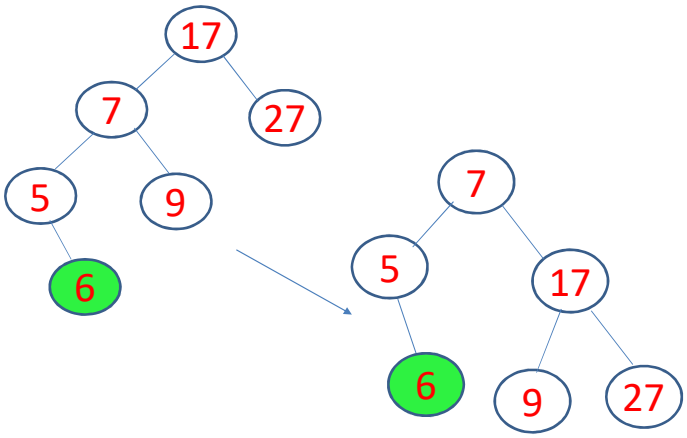
Chèn giá trị 6 vào cây thì cây không cân bằng



Cây kết quả như thế nào???

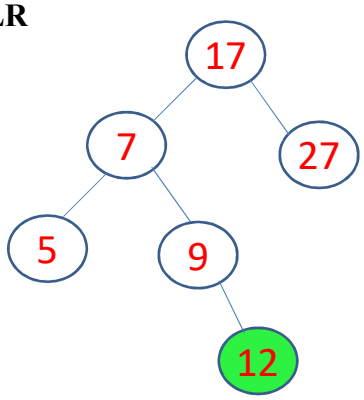
Cây AVL

Chèn giá trị 6 vào cây thì cây không cân bằng ➔ thực hiện quay trái



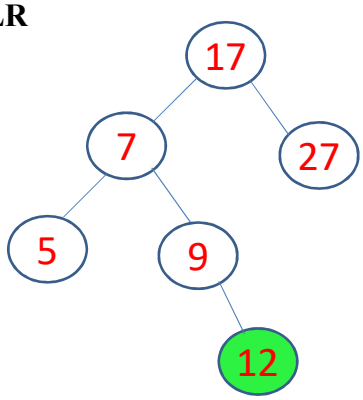
Phép quay con phải của con trái trên cây AVL

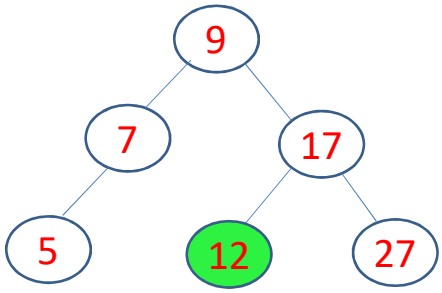
Chèn giá trị 12 vào cây thì cây không cân bằng: quay con phải của con trái qLR



Phép quay con phải của con trái trên cây AVL

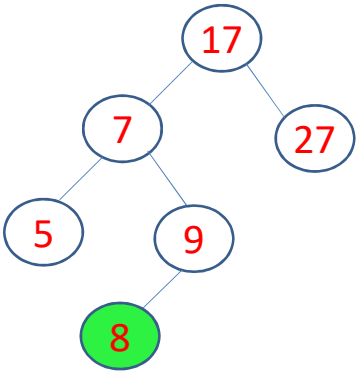
Chèn giá trị 12 vào cây thì cây không cân bằng: quay con phải của con trái qLR





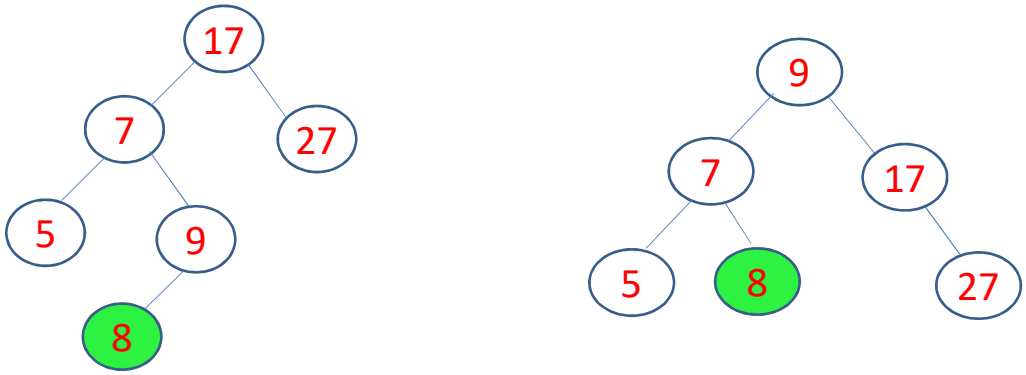
Phép quay con phải của con trái trên cây AVL

Chèn giá trị 8 vào cây thì cây không cân bằng: quay con phải của con trái

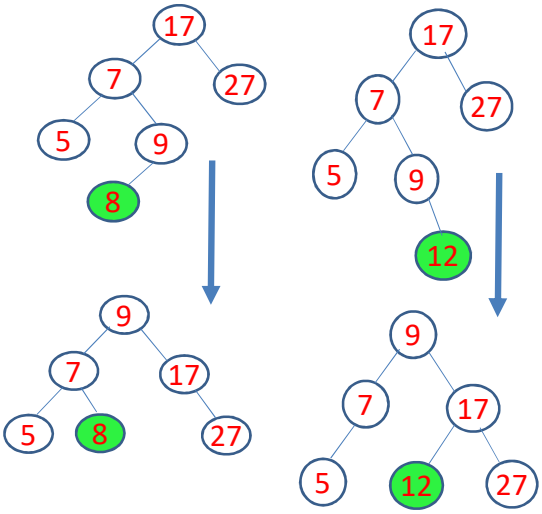


Phép quay con phải của con trái trên cây AVL

Chèn giá trị 8 vào cây thì cây không cân bằng: quay con phải của con trái

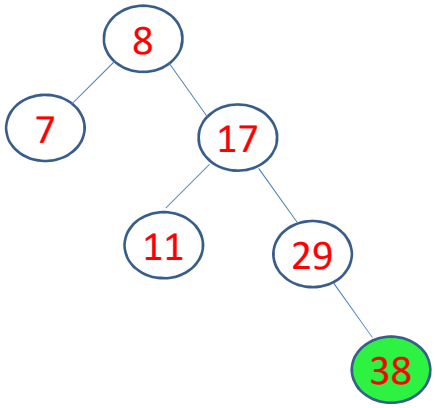


Phép quay con phải của con trái trên cây AVL



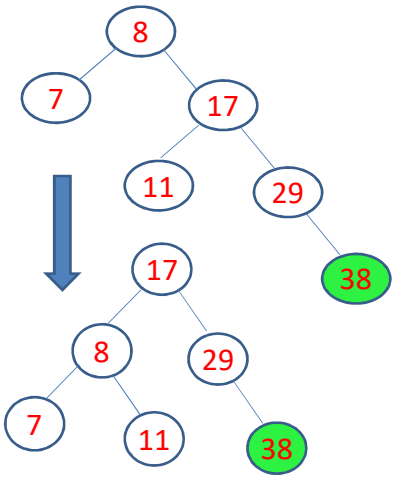
Phép quay con phải trên cây AVL

Chèn giá trị 38 vào cây thì cây không cân bằng: quay con phải



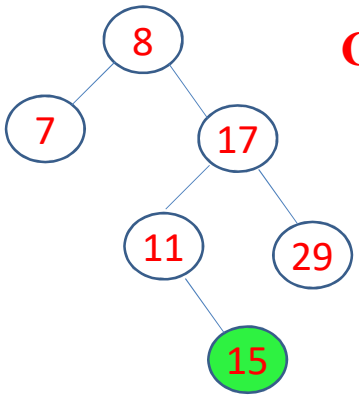
Phép quay con phải trên cây AVL

Chèn giá trị 38 vào cây thì cây không cân bằng: quay con phải



Phép quay con trái của con phải trên cây AVL

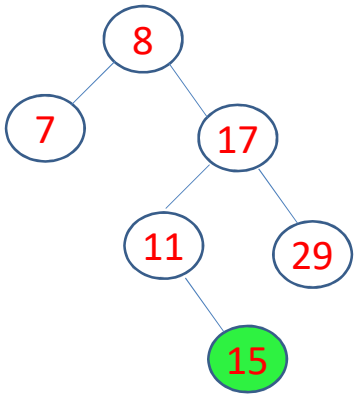
Chèn giá trị 15 vào cây thì cây không cân bằng: quay con trái của con phải

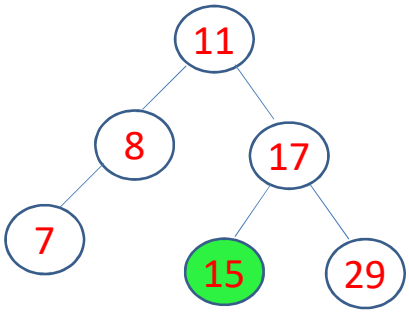


Cây kết quả như thế nào???

Phép quay con trái của con phải trên cây AVL

Chèn giá trị 15 vào cây thì cây không cân bằng: quay con trái của con phải

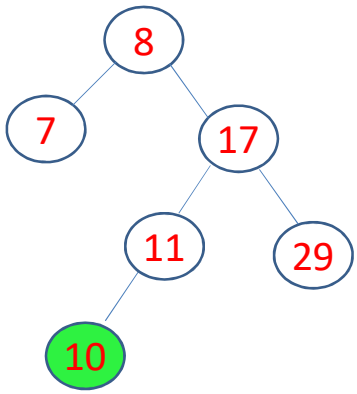




Phép quay con trái của con phải trên cây AVL

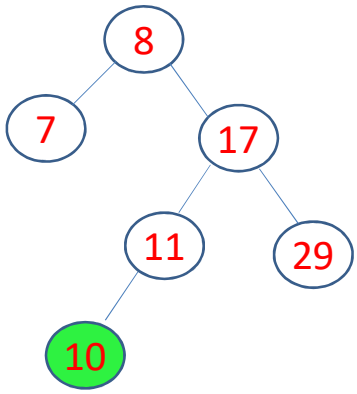
Chèn giá trị 10 vào cây thì cây không cân bằng: quay con trái của con phải

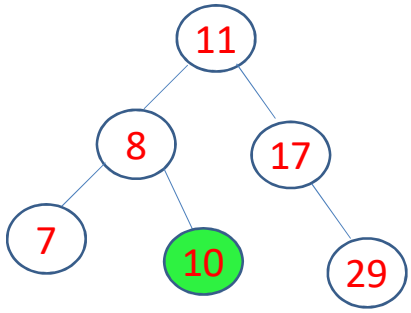
Cây kết quả như thế nào???



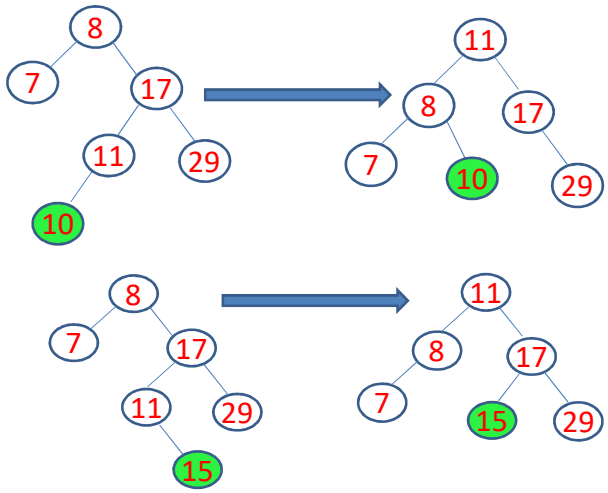
Cây AVL

Chèn giá trị 10 vào cây thì cây không cân bằng: quay con trái của con phải





Cây AVL **quay con trái của con phải**

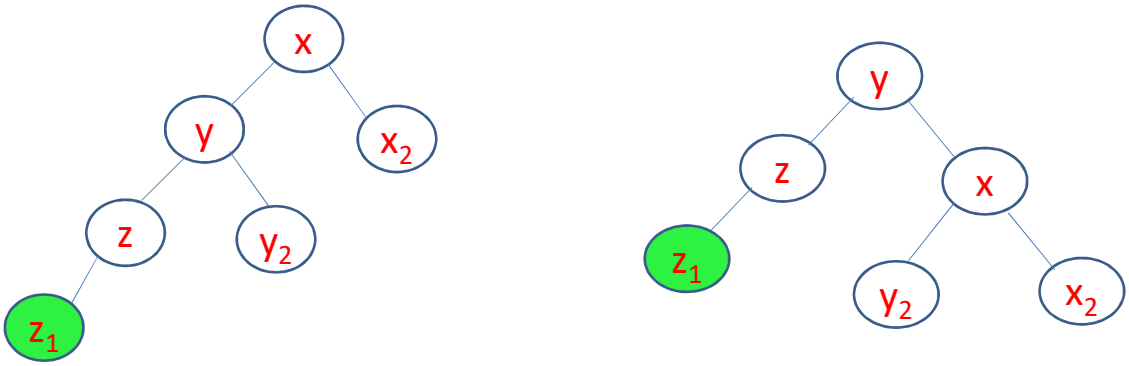


Chèn x vào cây AVL *TNode *chenx(TNode *&T, int x)*

```
Nếu T= rỗng thì T= new TNode(x);
Ngược lại nếu x= T->data thì báo đã có trong cây
Ngược lại  {
        if(x<T->data) T->left= chenx(T->left,x); else T->right= chenx(T->right,x);
        int h1= cao(T->left); int h2= cao(T->right);
        if(h1>h2+1)  {
                int h11= cao(T->left->left); int h12= cao(T->left->right);
                if(h11>h12) T= qL(T); else T= qLR(T);
        }
        else if (h2>h1+1 ) {
                int h22= cao(T->right->right);   int h21= cao(T->right->left);
                if(h22>h21) T= qR(T); else T= qRL(T); }
      }
return T;
```

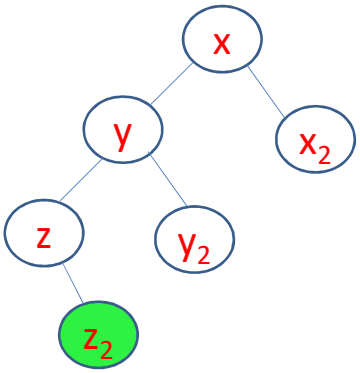
Các phép xoay trên cây AVL

Quay trái qL

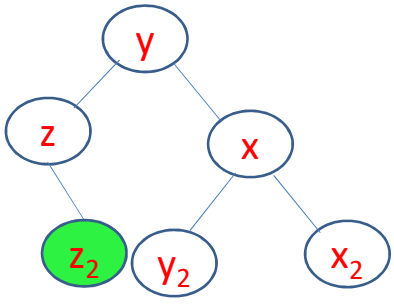


Các phép xoay trên cây AVL

Quay trái qL

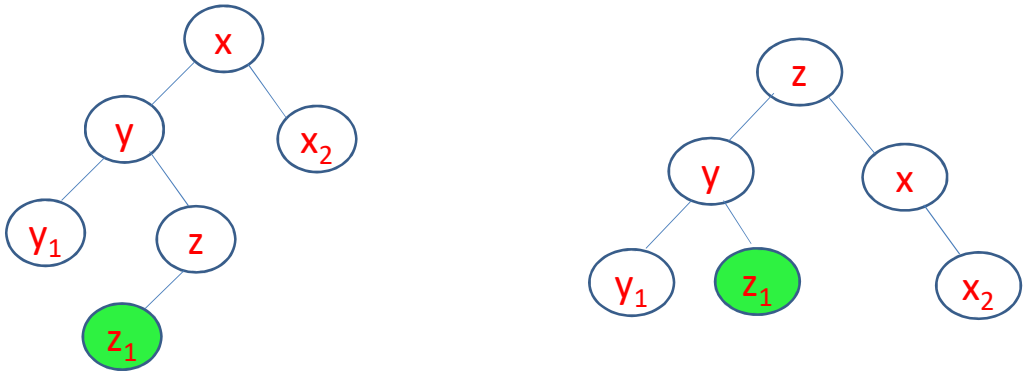



```
TNode* qL(TNode *&T)
{
    TNode *x= T;
    TNode *y= x->left;
    x->left = y->right; //y2
    y->right= x;
    T= y;
     return T;
}
```



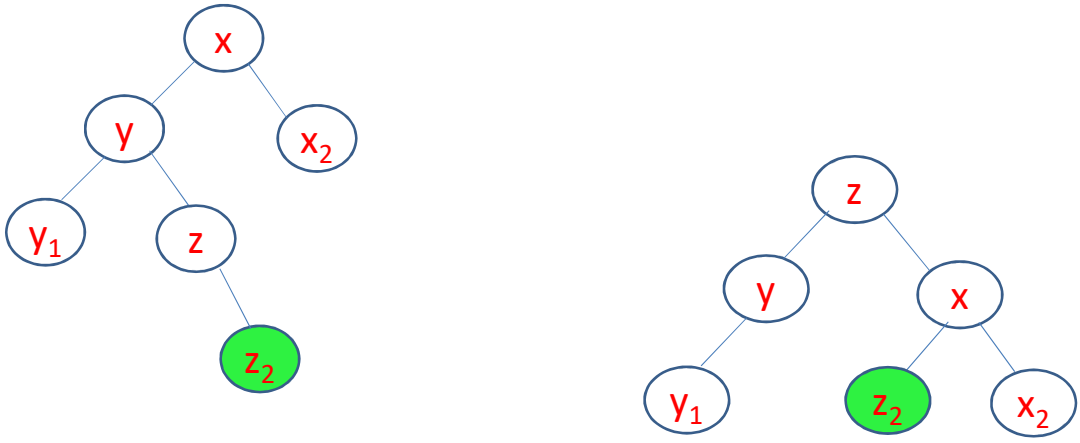
Các phép xoay trên cây AVL

Quay con phải của con trái: qLR



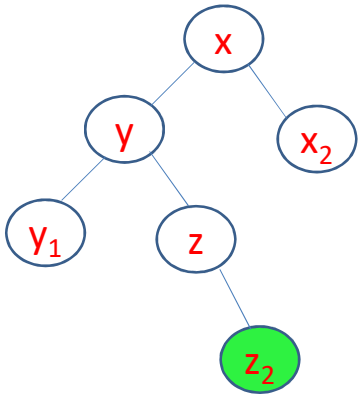
Các phép xoay trên cây AVL

Quay con phải của con trái: qLR

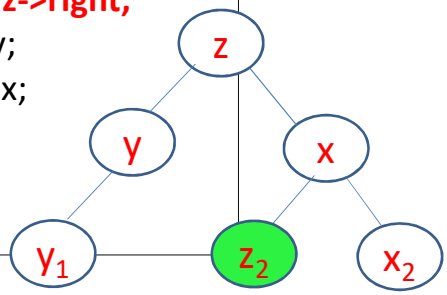


Các phép xoay trên cây A

Quay con phải của con trái: qLR



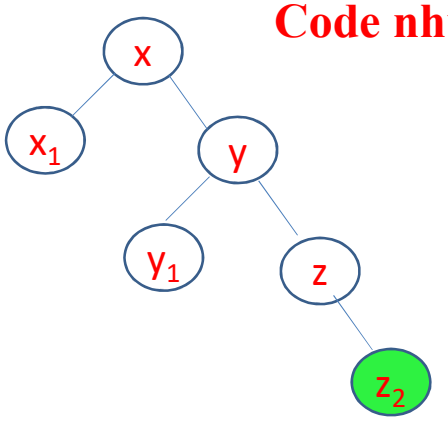

```
TNode* qLR(TNode *&T)
{
    TNode *x= T;
    TNode *y= x->left;
    TNode *z= y->right;
    y->right= z->left;
    x->left = z->right;
    z->left= y;
    z->right=x;
    T=z;
    return T;
}
```

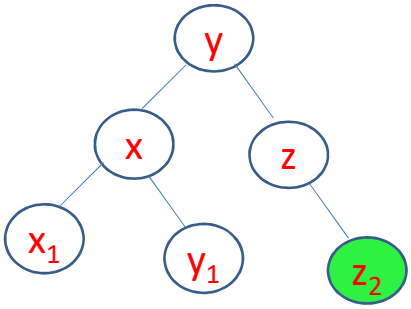


Các phép xoay trên cây AVL

Quay con phải qR

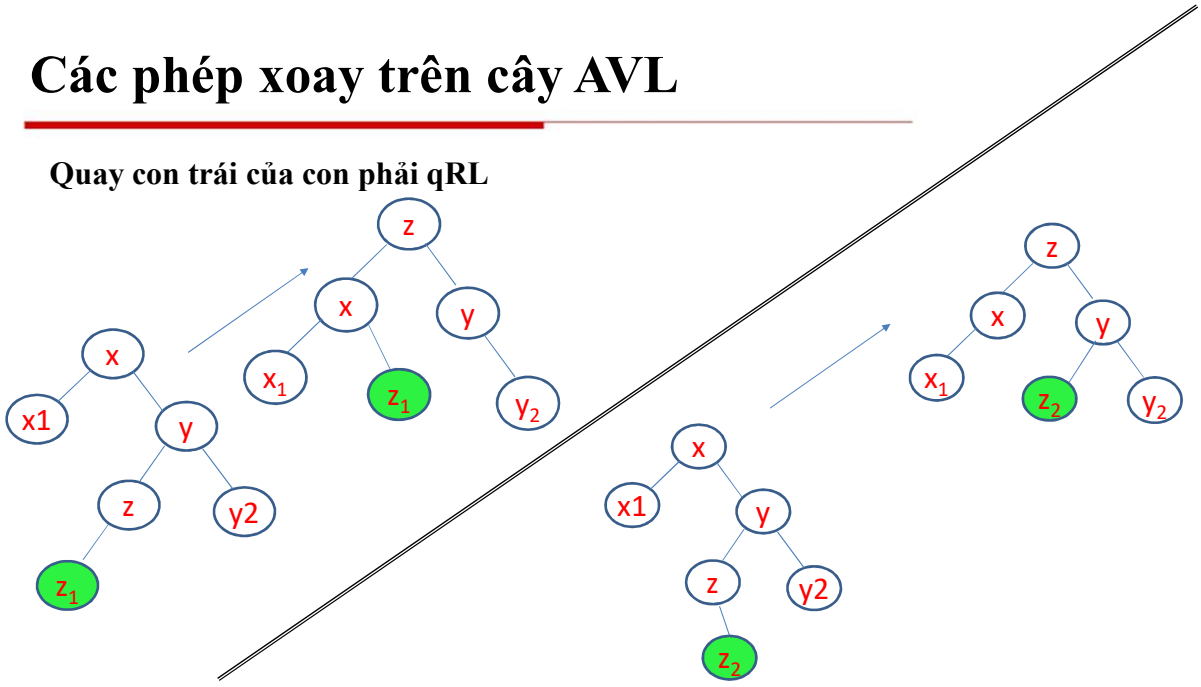
Code như thế nào???





Các phép xoay trên cây AVL

Quay con trái của con phải qRL



Tạo cây AVL

Vẽ cây sau khi chèn lần lượt các giá trị sau vào cây rỗng???

3 7 8 6 5 20 9 15 17

Tạo cây AVL

Viết chương trình:
Nhập các giá trị và chèn vào cây AVL: nhapcayAVL()
Duyệt cây theo chiều rộng
Thử nghiệm với các giá trị sau: 3 7 8 6 5 20 9 15 17

Cần viết các hàm
- **chenx(int x, TNode *&T)**
- **cao(TNode *T)**
- **Chieurong(TNode *T)**

Chèn giá trị vào cây AVL

Vẽ cây sau khi chèn lần lượt các giá trị:

a/ 20, 37, 3, 34, 36, 10, 5, 40, 50
b/ 20 10 17 41 55 5 60 72 4 18 27 35
c/ 5 9 11 3 7 6 13
d/ 27 11 5 29 24 26 3 4 10 16 13

Vẽ cây sau khi chèn lần lượt các giá trị:

a/ 20, 37, 3, 34, 36, 10, 5, 40, 50
b/ 20 10 17 41 55 5 60 72 4 18 27 35
c/ 5 9 11 3 7 6 13
d/ 27 11 5 29 24 26 3 4 10 16 13

Xem kết quả mô phỏng tại:

http://www.cs.armstrong.edu/liang/animation/web/AVLTree.html

Deletion

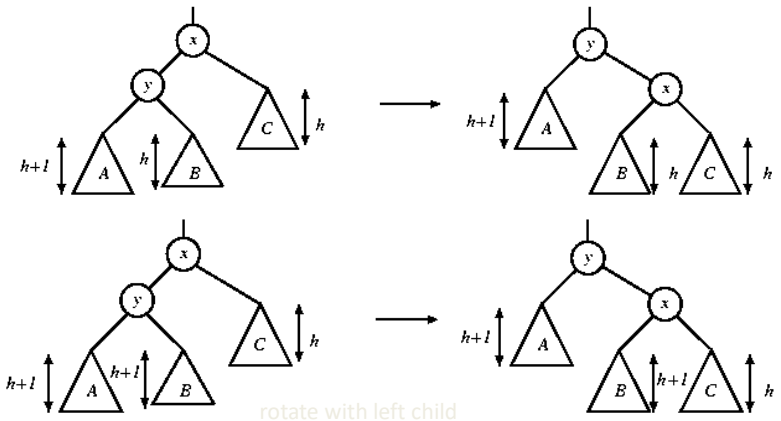
- Delete a node x as in ordinary binary search tree. Note that the last node deleted is a leaf.
- Then trace the path from the new leaf towards the root.
- For each node x encountered, check if heights of left(x) and right(x) differ by at most 1. If yes, proceed to parent(x). If not, perform an appropriate rotation at x. There are 4 cases as in the case of insertion.
- For deletion, after we perform a rotation at x, we may have to perform a rotation at some ancestor of x. Thus, we must continue to trace the path until we reach the root.

Deletion

- On closer examination: the single rotations for deletion can be divided into 4 cases (instead of 2 cases)
  - Two cases for rotate with left child
  - Two cases for rotate with right child

Single rotations in deletion

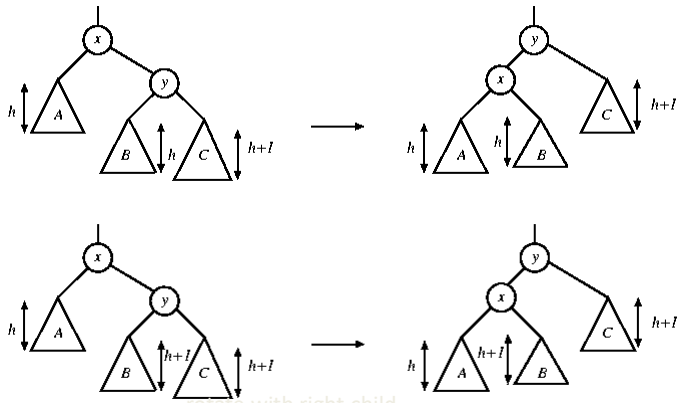
In both figures, a node is deleted in subtree C, causing the height to drop to h. The height of y is h+2. When the height of subtree A is h+1, the height of B can be h or h+1. Fortunately, the same single rotation can correct both cases.



rotate with left child

Single rotations in deletion

In both figures, a node is deleted in subtree A, causing the height to drop to h. The height of y is h+2. When the height of subtree C is h+1, the height of B can be h or h+1. A single rotation can correct both cases.



rotate with right child

Rotations in deletion

- There are 4 cases for single rotations, but we do not need to distinguish among them.
- There are exactly two cases for double rotations (as in the case of insertion)
- Therefore, we can reuse exactly the same procedure for insertion to determine which rotation to perform

Ứng dụng của cây: Cây Huffman

Cho file văn bản a.txt chứa các ký tự như sau:

Ký tự	A	B	C	D	E	F	G	H	I
Số lần xuất hiện trong văn bản	2000	200	50	180	1000	600	420	400	1020

Cây Huffman (dùng để nén dữ liệu)

Mỗi ký tự chiếm 1 byte

Ký tự	A	B	C	D	E	F	G	H	I
Số lần xuất hiện trong văn bản	2000	200	50	180	1000	600	420	400	1020

Để lưu trữ file như vậy thì cần bao nhiêu byte???

Cây Huffman (dùng để nén dữ liệu)

Cho file văn bản a.txt chứa các ký tự như sau:
Mỗi ký tự chiếm 1 byte
Kích thước file: 5870 byte

Ký tự	A	B	C	D	E	F	G	H	I
Số lần xuất hiện trong văn bản	2000	200	50	180	1000	600	420	400	1020

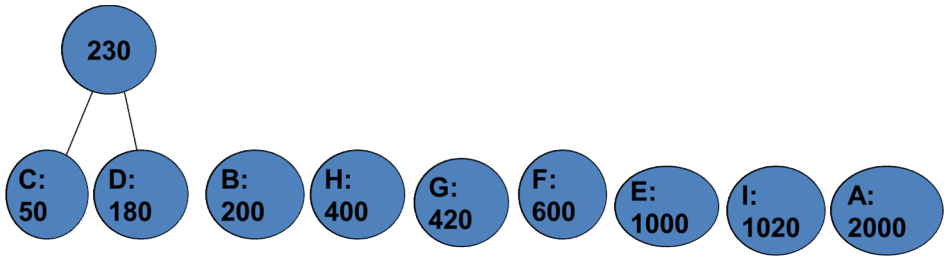
Cây Huffman

Thuật toán nén dữ liệu Huffman dựa vào cây nhị phân

Bước 1: Xây dựng rừng (gồm nhiều cây, mỗi cây chỉ có 1 nốt gồm ký tự và tần số xuất hiện của ký tự)

Cây Huffman

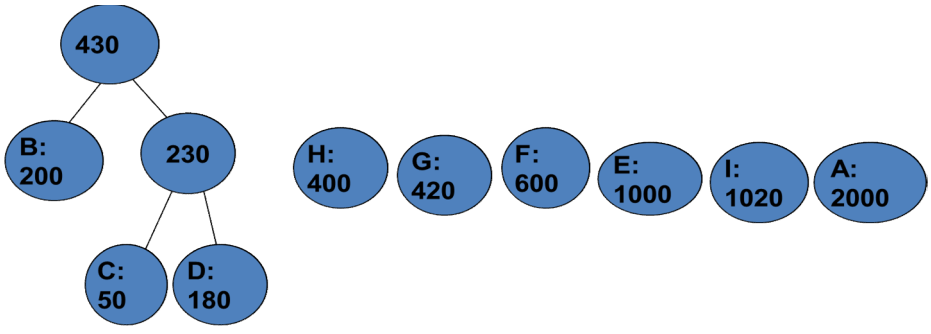
Gộp hai cây có trọng số nhỏ nhất thành 1 cây. Trọng số của cây mới bằng tổng trọng số của hai cây con (qui ước cây có trọng số nhỏ hơn nằm bên trái)



Cây Huffman

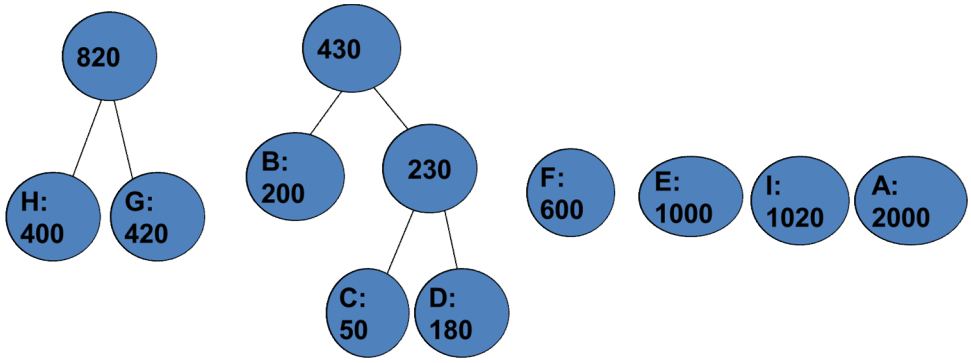
Gộp hai cây có trọng số nhỏ nhất thành 1 cây. Trọng số của cây mới bằng tổng trọng số của hai cây con (qui ước cây có trọng số nhỏ hơn nằm bên trái) **cho đến khi rừng thành 1 cây**

Lặp tiếp tục



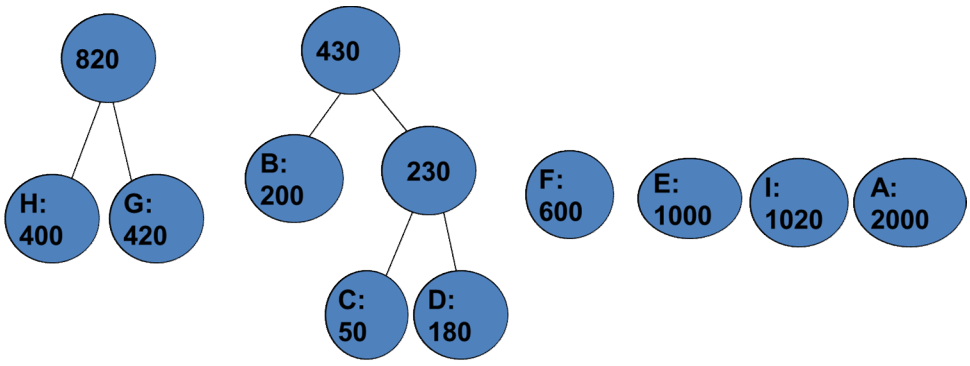
Cây Huffman

Lặp tiếp tục ….



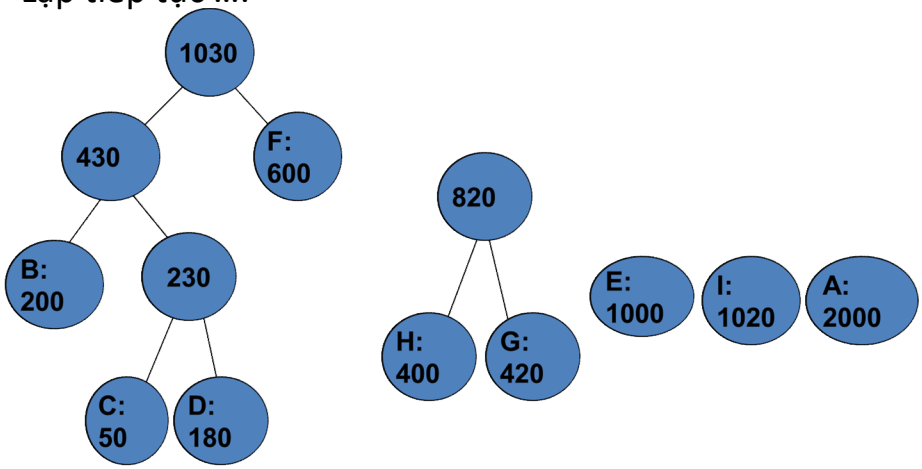
Cây Huffman

Tiếp theo sẽ nhóm cây nào???



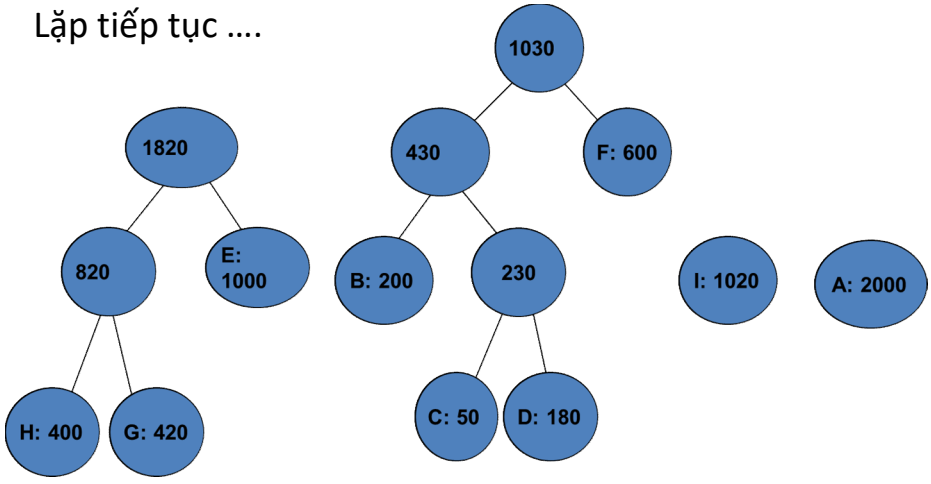
Cây Huffman

Lặp tiếp tục ….

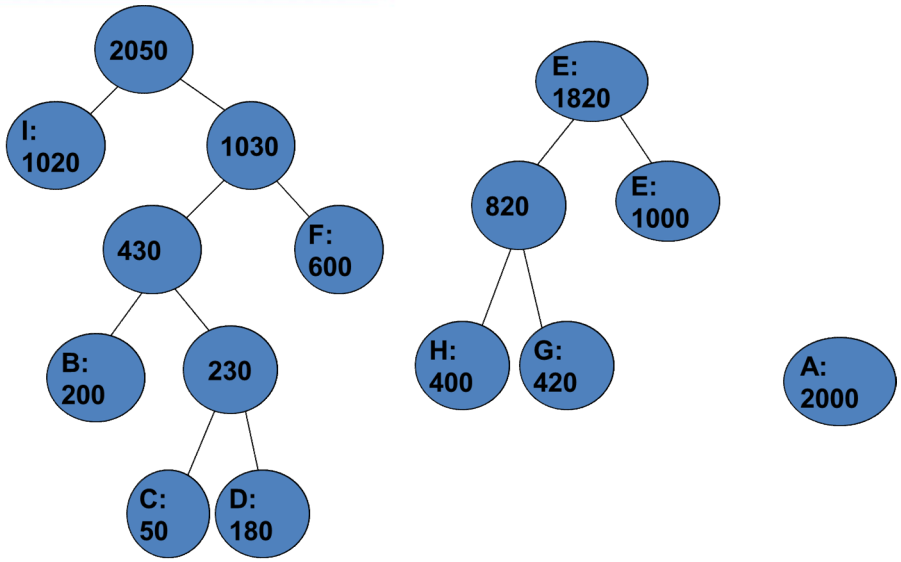


Cây Huffman

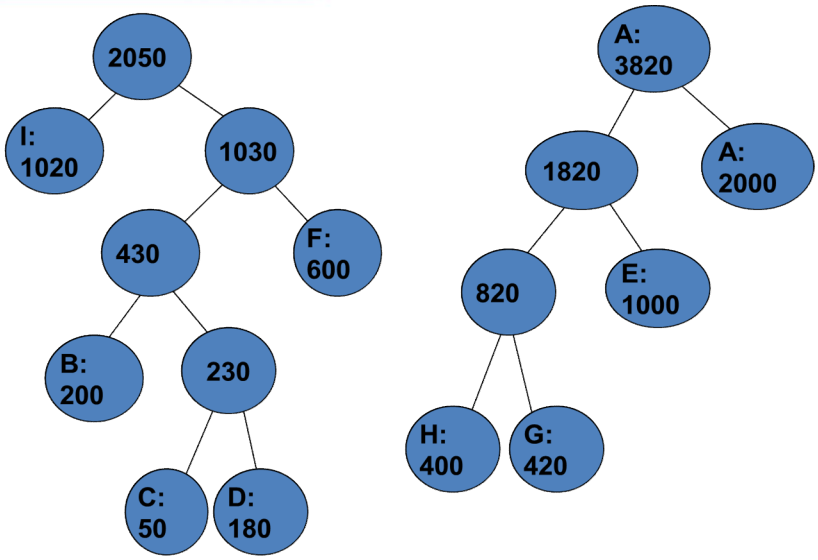
Lặp tiếp tục ….



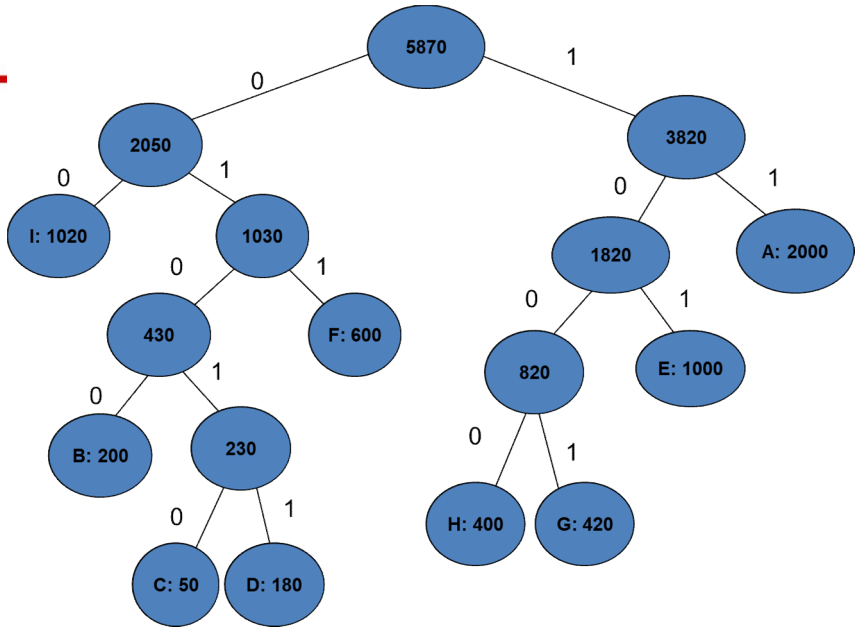
Cây Huffman

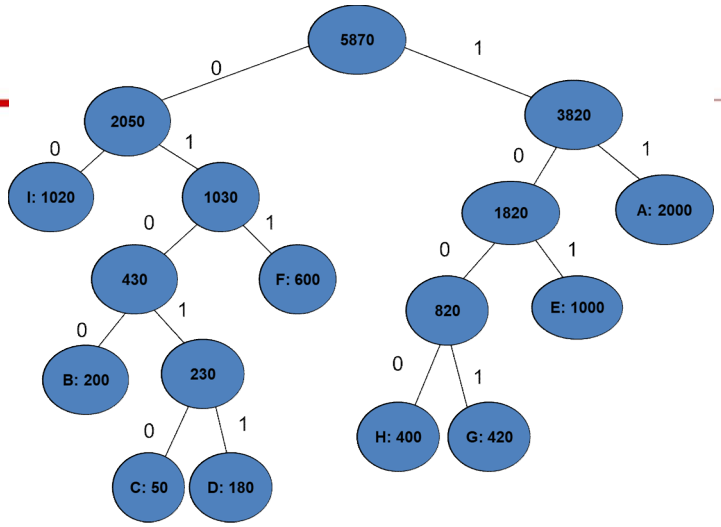


Cây Huffman



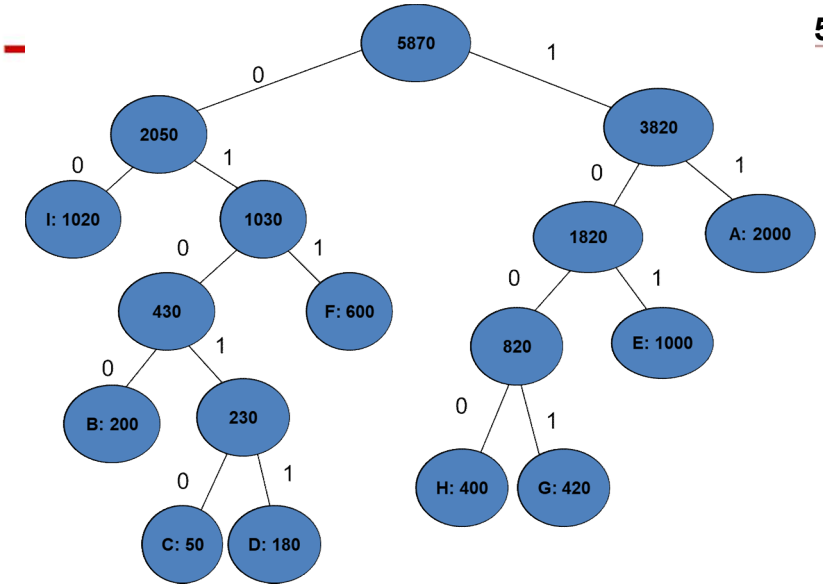
Cây Huffman





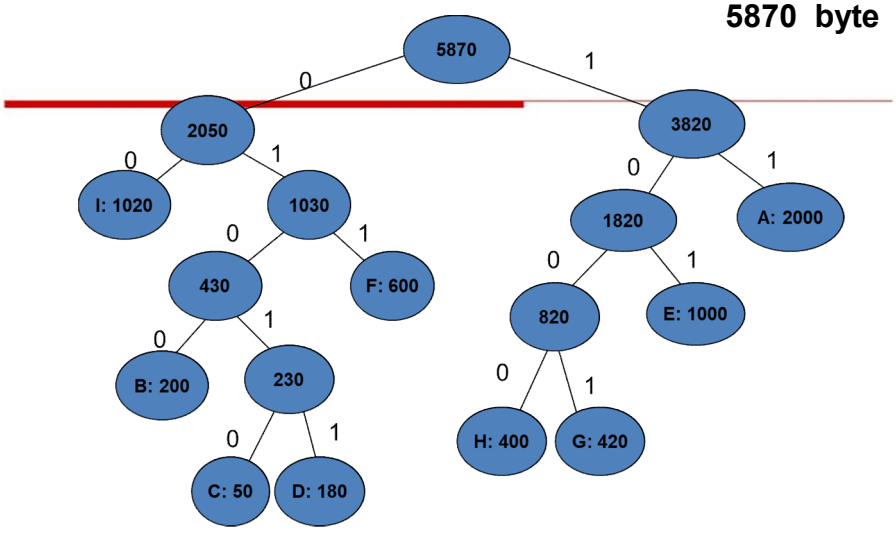
Thay mỗi ký tự bằng một nhóm các bit

A: 11 B: 0100 C: 01010 D: 01011 E: 101
F: 011 G: 1001 H: 1000 I: 00



5870 byte

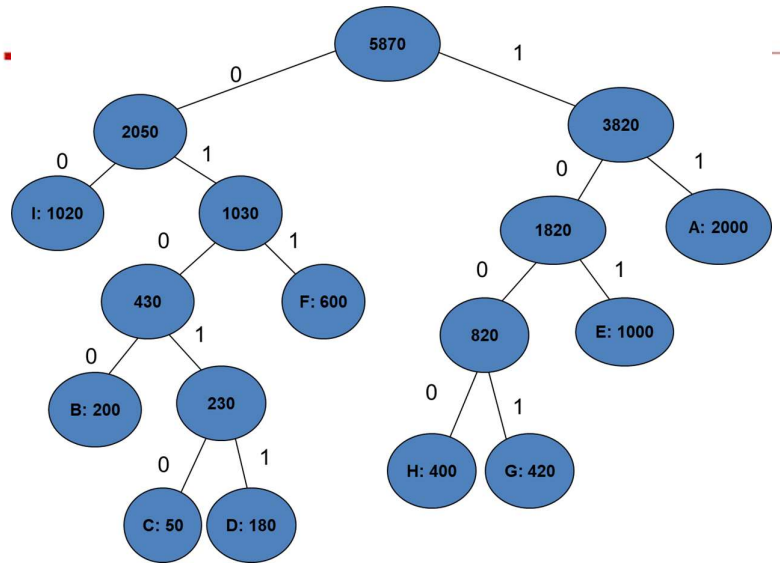
Số byte cần phải lưu là bao nhiêu???



5870 byte

=2000 * 2 + 200*4 + 50*5 + 180*5 + 1000*3 + 600*3 + 420*4 + 400*4 + 1020* 2 = 16070 bit = 2009 byte

5870 byte



Số byte tiết kiệm = (5870 – 2009) byte

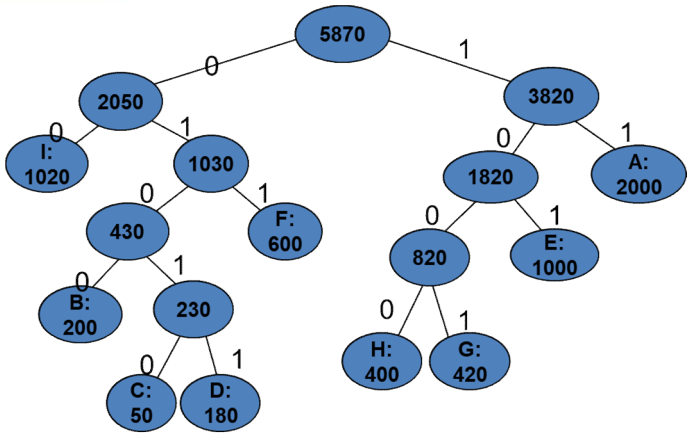
5870 byte

Cây Huffman

Giả sử có đoạn văn bản:

AIAFHAC DGE…

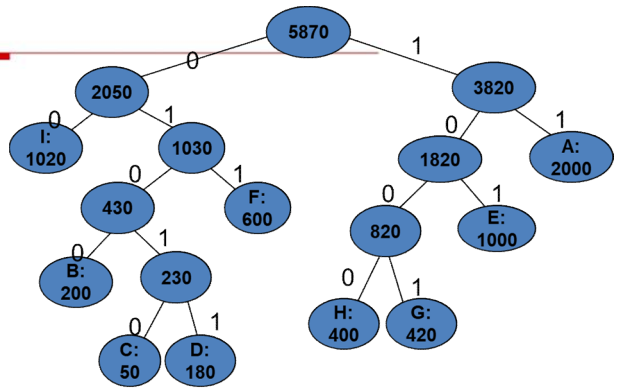
Khi mã hóa sẽ thành:



Cây Huffman

Giả sử có đoạn văn bản:

AIAFHAC DGE



Khi mã hóa sẽ thành:

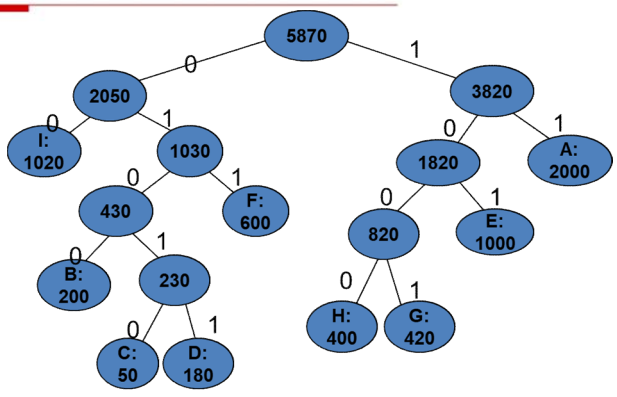
11001101110001101010101011001101

A: 11 B: 0100 C: 01010 D: 01011 E: 101
F: 011 G: 1001 H: 1000 I: 00

Cây Huffman

Giả sử có đoạn văn bản:

AIAFHACD GE



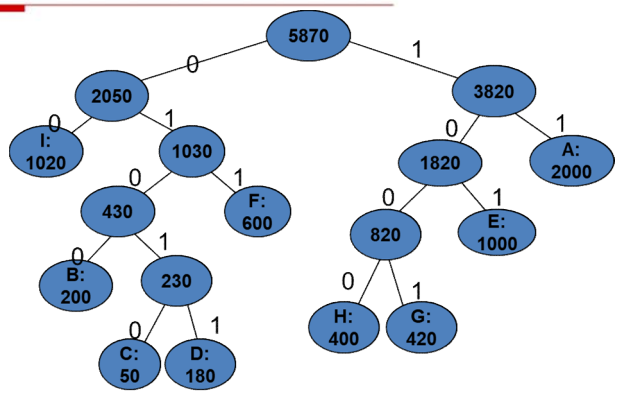
Khi mã hóa sẽ thành:
11 00 11 011 1000
11 01010 01011
1001 101

Khi ghi xuống file sẽ nhóm 8 bit lại thành 1 byte để ghi
1100110111000110**10100101**11001101

Giả sử có đoạn văn bản:

AIAFHAC DGE

Khi mã hóa sẽ thành:
11 00 11 011 1000
11 01010 01011
1001 101



Trong trường hợp thiếu các bit thì ghép cho đủ 1 byte để ghi xuống file (có thể là toàn các bit 1 hoặc toàn các bit 0)

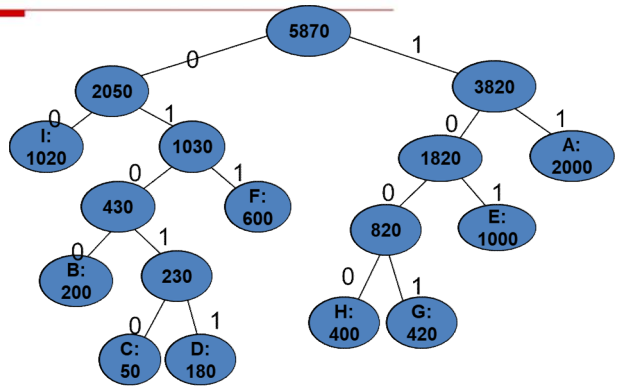
Cây Huffman

5870 byte

Giả sử có đoạn văn bản:

AIAFHAC DGE

Khi mã hóa sẽ thành:
11 00 11 011 1000
11 01010 01011
1001 101



Trong trường hợp thiếu các bit thì ghép cho đủ 1 byte để ghi xuống file (có thể là toàn các bit 1 hoặc toàn các bit 0)

Cây Huffman

Thông tin ghi ở file mã hóa gồm:

Các ký tự (byte) và tần số xuất hiện các ký tự (byte)
Các byte mã hóa theo thứ tự trong file

Khi giải mã

Dựa vào thông tin bảng tần suất, xây dựng cây
Thay các bit bằng các ký tự tương ứng

Bài tập về nhà

Bài tập

Cho file văn bản A.txt
Lập bảng tần suất của các ký tự trong file theo kiểu:
Ký tự : Số lần xuất hiện

Ví dụ file A.TXT

ABABA BABI DSOIWI DWHIDH OXAX
KLKAOJSAO IJDOI ADOIH AQ XAKJ
XIQ WU EU (*QE (*QUS A JS()EUQE Q)(
EUQ JD SJDJAO IJD OIAD

Bài tập về nhà

Lập bảng tần suất của các ký tự trong file A.txt

B1: Khởi tạo mảng gồm 256 phần tử và gán a[i] =0
B2: Mở file A.txt
B3: Lặp thao tác sau trong khi chưa hết file
- đọc 1 byte k
- a[k] ++;
B4: Cho i từ 0 đến 255
nếu a[i] > 0 thì in giá trị i và a[i]

```
import java.io.FileInputStream;
import java.io.InputStream;
public class HelloInputStream {
  public static void main(String[] args) {
    try {
        // Tạo một đối tượng InputStream theo class con của nó, đây là luồng đọc một file.
        InputStream is = new FileInputStream("data.txt");
        int i = -1;
       /*Đọc lần lượt các byte trong luồng, mỗi lần đọc ra 8 bit, chuyển nó thành số int, khi đọc ra giá trị -1 nghĩa là kết thúc luồng.*/

        while ((i = is.read()) != -1)  System.out.println(i + "  " + (char) i);
        is.close();
    } catch (Exception e) {  e.printStackTrace();    }
  }
}
```

Bài 1: AVLTREE.java
Viết chương trình đọc lần lượt các dữ liệu từ file AVLTREE.INP chèn vào cây AVL rồi viết kết quả duyệt cây theo mức ra file AVLTREE.OUT. Dữ liệu trong file AVLTREE.INP gồm các số nguyên cách nhau bởi dấu cách.

Bài 2: LIST.java
Viết chương trình đọc lần lượt các dữ liệu từ file LIST.INP vào danh sách theo thứ tự tăng dần, sau đó xóa các phần tử lớn thứ k_1, k_2, k_3 ra khỏi danh sách. CÁC phần tử được xóa được ghi ra file LIST.OUT. Dữ liệu trong file LIST.INP gồm dòng đầu là 3 số nguyên k1, k2, k3, các dòng sau đó là các dữ liệu được chèn vào danh sách.

AVLTREE.INP

```
5 9 11 7
6 13
```

AVLTREE.OUT

```
9 6 11 5
7  13
```

LIST.INP

```
5 5 3
5 1 6 23 64
4 23 9
```

LIST.OUT

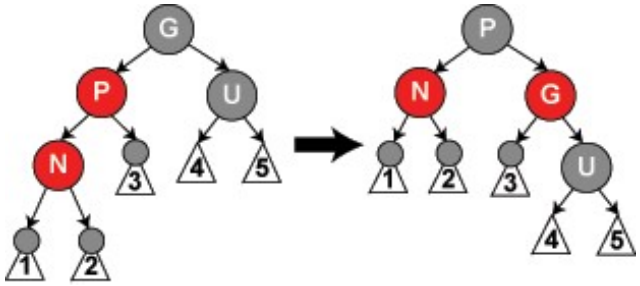
```
9 23 5
```

Tạo thư mục CMUSE252BIS_Hotensinhvien_4SOSINHVIEN chỉ chứa 2 bài trên, nén thành file zip CMUSE252BIS_Hotensinhvien_4SOSINHVIEN .zip
Gửi file nén qua mail dieuhb@gmail.com, tiêu đề mail **CMU-SE252BIS_Họ tên_4sốBD_TEST02**

Tài liệu đọc thêm về cây AVL

http://www2.hawaii.edu/~tvs/avl-tree/
https://www.geeksforgeeks.org/avl-tree-set-1-insertion/
https://www.geeksforgeeks.org/avl-tree-set-2-deletion/
https://www.tutorialspoint.com/cplusplus-program-to-implement-avl-tree

Cây đỏ đen



http://www.btechsmartclass.com/data_structures/red-black-trees.html

https://www.cs.auckland.ac.nz/software/AlgAnim/red_black.html

Link YouTube

https://www.youtube.com/watch?v=FNeL18KsWPc

https://www.youtube.com/watch?v=-9sHvAnLN_w