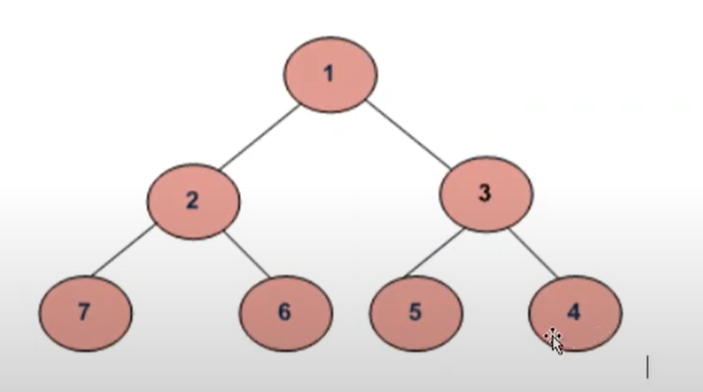
Cây nhị phân (Binary Indexed Tree)

1. Bài 1: Kiếm tra node lá:

Cho cây nhị phân, nhiệm vụ của bạn là kiểm tra xem tất cả các node lá của cây có cùng một mức hay không? Cây dưới là Yes



Input: + Dòng đầu là số lượng test case

+ Những dòng sau là bộ test. Dòng đầu tiên là N số cạnh của cây; dòng tiếp theo đưa vào N bộ ba u,v,x, trong đó u là node cha, v là node con, x là kí tự L, R (con bên trái hay bên phải của node cha)

+ 1<=T<=100, 1<=N<=1e3, 1<=u, v<=1e4

Output: Đưa ra kết quả từng dòng.

+ 2

+ 2

+ 1 2 R 1 3 L

=> 1

+ 4

+ 10 20 L 10 30 R 20 40 L 20 60 R

=> 0

1. Bài 2: Cây biểu thức

Cây biểu thức là một cây nhị phân trong đó mỗi node trung gian là một phép toán, mỗi node lá là một toán hạng. Ví dụ với biểu thức P = 3+((5+9)\*2) sẽ được biểu diễn như cây dưới đây.

Đối với cây biểu thức, duyệt theo thứ tự trước ta sẽ được biểu thức tiền tố, duyệt theo thứ ự sau ta sẽ được biểu thức hậu tố, duyệt theo thứ tự giữa ta được biểu thức trung tố. Chú ý, cây biểu thức luôn là cây nhị phân đầy (mỗi node trung gian đều có hai node con).

Cho biểu thức hậu tố P, hãy sử dụng cây biểu thức để đưa ra biểu thức trung tố tương ứng với biểu thức P.

* Input:
* Dòng đầu tiên đưa vào số lượng bộ test t
* Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test là một biểu thức hậu tố P
* 1<=T<=100, 1<=length(P)<=100
* Output: In ra kết quả trên từng dòng
* 2
* ab+ef\*g\*-

=> a+b-e\*f\*g

* wlrb+-\*
* => w\*l-r+b

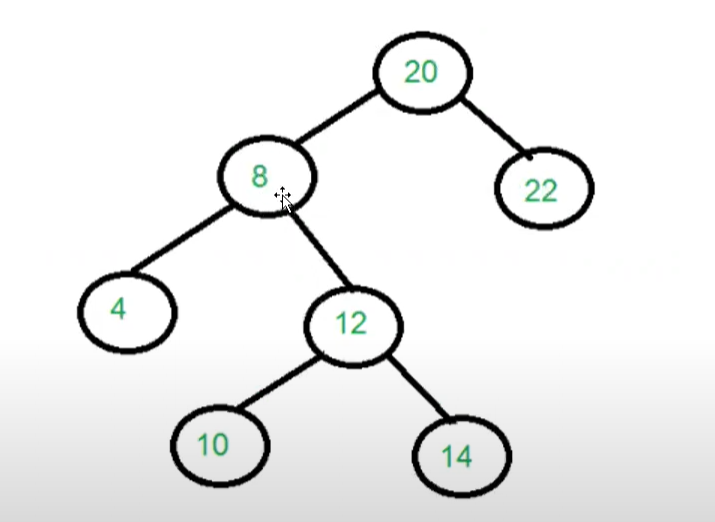
1. Bài 3: Duyệt cây 1

Cho phép duyệt cây nhị phân Inorder và PreOrder, hãy đua ra kết quả phép duyệt cây PostOrder của cây nhị phân. Ví dụ với cây nhị phân có các phép duyệt cây nhị phân của cây dưới đây

* Inorder: 4 2 5 1 2 6
* PreOrder: 1 2 4 5 3 6
* PostOrder: 4 5 2 6 3 1
* Input:
* Dòng đầu số lượng Testcase
* Những dòng sau là bộ test, gồm 3 dòng. Dòng đầu N node, N số theo phép duyệt Inorder, N số theo phép duyệt PreOrder cách nhau bởi các khoảng trống
* 1<=T<=100; 1<=N<=1000, 1<=giá trị node <=10000
* Output: In ra kết quả

1. Bài 4: Duyệt cây theo mức (Level-Order):

Cho cây nhị phân, nhiệm vụ của bạn là duyệt cây theo Level-Order. Phép duyệt này là phép thăm node theo từng mức của cây, Ví dụ với cây dưới đây sẽ cho ta kết quả của phép duyệt Level-Order: 20 8 22 4 12 10 14



* Input: Dòng đầu tiên đưa vào số lượng Testcase
* Những dòng tiếp là các bộ test, gồm 2 dòng. Dòng đầu tiên là số N là số lượng cạnh của cây, tiếp theo là N bộ ba (u,v,x)
* 1<=T<=100, 1<=N<=1000, 1<=u,v<=10000
* Output: In ra kết quả Level-Order
* 2
* 2
* 1 2 R 1 3 L
* 1 3 2
* 4
* 10 20 L 10 30 R 20 40 L 20 60 R
* 10 20 30 40 60

1. Bài 5: Spiral Order

In ra cây được duyệt theo Spiral Order (từ trái qua phải rồi ngược lại)

* Input:
* 2
* 2
* 1 2 R 1 3 L
* 1 3 2
* 4
* 10 20 L 10 30 R 20 40 L 20 60 R
* 10 20 30 60 40

1. Bài 6: Cây nhị phân hoàn hảo

Cho cây nhị phân, nhiệm vụ củ bạn là kiểm tra xem cây nhị phân có phải là một cây hoàn hảo hay không (perfect tree)? Một cây nhị phân được gọi là hoàn hảo nếu tất cả các node trung gian của nó đều có hai node con và tất cả các node lá đều có cùng một mức

* Input:
* Số lượng Test
* Bộ test gồm 2 dòng. Dòng đầu tiên N cạnh, dòng tiếp theo bộ ba (u,v,x)
* 1<=T<=100, 1<=N<=1000, 1<=u,v<=10000
* Output: Yes or No
* 3
* 6
* 10 20 L 10 30 R 20 40 L 20 50 R 30 60 L 30 70 R
* YES
* 2
* 18 15 L 18 30 R
* YES
* 5
* 1 2 L 2 4 R 1 3 R 3 5 L 3 6 R
* NO

1. Bài 7: Cây nhị phân đầy đủ

Cho cây nihj phân, kiểm tra xem cây nhị phân có phải là đầy đủ hay không (Full binary tree) Một cây nhị phân được gọi là đầy đủ nếu tất cả các node trung gian của nó đều có hai node con

* Input:
* Số lượng test
* Mỗi bộ test gồm 2 dòng. N số lượng cạnh và bộ ba (u,v,x)
* 1<=T<=100, 1<=N<=1000, 1<=u,v<=10000
* Output: Yes or No
* 2 : t
* 4
* 1 2 L 1 3 R 2 4 L 2 5 R
* YES
* 3
* 1 2 L 1 3 R 2 4 L
* NO

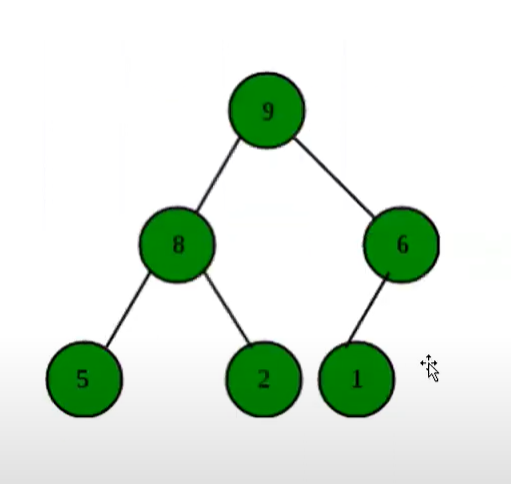
1. Bài 8: Cây nhị phân bằng nhau

Cho hai cây nhị phân, nhiệm vụ của bạn là kiểm tra xem cây nhị phân có giống nhau hay không?

* Input:
* Số lượng bộ test
* Mỗi bộ test gồm 3 dòng. N số lượng cạnh của cây, N bộ ba (u,v,x)
* 1<=T<=100, 1<=N<=1000, 1<=u,v<=10000
* Output: Yes or No
* 2
* 2
* 1 2 L 1 3 R
* 2
* 1 2 L 1 3 R
* Yes
* 2
* 1 2 L 1 3 R
* 2
* 1 3 L 1 2 R
* No

1. Bài 9: Tổng nốt lá bên phải

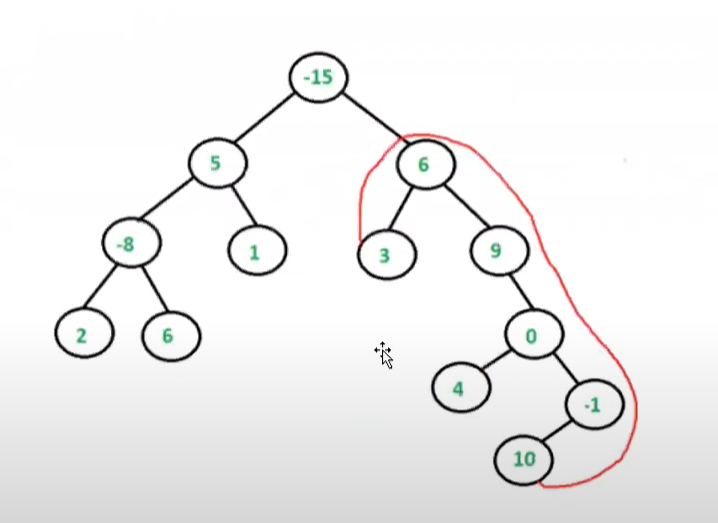
Cho cây nhị phân, nhiệm vụ của bạn là tính tổng của tất cả các node lá bên phải trên cây. Ví dụ với cây dưới dây ta có kết quả bằng 2



* Input:
* 2
* 2
* 1 2 L 1 3 R
* 3
* 5
* 10 20 L 10 30 R 20 40 L 20 60 R 30 90 L
* 60

1. Bài 10: Tổng lớn nhất

Cho cây nhị phân có giá trị mỗi node là một số. Nhiệm vụ của bạn là tìm tổng lớn nhất từ một node lá này sang một node lá khác? Ví dụ với cây dưới đây ta có tổng lớn nhất là 27



* Input:
* Số lượng bộ test
* Mỗi bộ test gồm 2 dòng. N số lượng cạnh của cây, N bộ ba (u,v,x)
* 1<=T<=100, 1<=N<=1000, 1<=u,v<=10000
* Output: Đưa ra kết quả mỗi test
* 1
* 12
* -15 5 L -15 6 R 5 -8 L 5 1 R -8 2 L -8 -3 R 6 3 L 6 9 R 9 0 R 0 4 L 0 -1 R -1 10 L
* 27

1. Bài 11: Độ cao của cây

Cho một cây có N node, gốc tại 1. Độ cao của một node u được tính bằng khoảng cách từ u tới node gốc. Độ cao của node gốc bằng 0. Nhiệm vụ của bạn là hãy tìm node xa node gốc nhất và in ra độ cao của node này.

* Input:
* Số lượng bộ test
* Mỗi test bắt đầu bằng 1 số nguyên N (1<=N<=100.000)
* N-1 dòng tiếp theo, mỗi dòng gồm 2 số nguyên u,v cho biết node u nối với node v.
* Output: In ra kết quả của đề bài
* 2
* 5
* 1 2
* 1 3
* 2 4
* 2 5
* 2
* 4
* 1 2
* 2 3
* 3 4
* 3

1. Bài 12: Đường đi tới node lá

Cho một cây có N node, gốc tại 1. Với mỗi Node lá, bạn hãy in ra đường đi từ Node gốc tới nó.

* Input:
* Số lượng test case
* Số nguyên N (1<=N<=1000)
* N-1 dòng gồm hai số u,v (cha-con)
* Output: K dòng (Số lượng node lá). Đường đi từ node gốc tới node lá. Node lá nào có nhãn nhỏ hơn thì in ra trước
* 2
* 5
* 1 2
* 1 3
* 2 4
* 2 5
* 1 3
* 1 2 4
* 1 2 5
* 4
* 1 2
* 2 3
* 3 4
* 1 2 3 4

1. Bài 13: Duyệt cây 2

Cho hai mảng InOrder và levelOrder, xây cây nhị phân và đưa ra thứ tự PostOrder. Ví dụ:

Inorder: 4 8 10 12 14 20 22

levelOrder: 20 8 22 4 12 10 14

* Input:
* Số lượng test case
* Bộ test gồm 3 dòng: N số lượng node, N phần tử InOrder và N phần tử levelOrder. ( 1<=T<=100, 1<=N<=1000, 1<=val<= 1e4)
* Output: Đưa ra thứ tự postOrder
* 2
* 3
* 1 0 2
* 0 1 2
* 1 2 0
* 7
* 3 1 4 0 5 2 6
* 0 1 2 3 4 5 6
* 3 4 1 5 6 2 0

1. Bài 14: Duyệt cây BST 1

Cho mảng a[], gồm n node biểu diễn theo thứ tự PreOrder của BST, đưa ra phép duyệt sau của BST.

* Input:
* Số lượng bộ test
* Gồm 2 dòng, số lượng node N, và các phần tử a[].
* ( 1<=T<=100, 1<=N<=1000, 1<=val<= 1e4)
* Output: đưa ra thứ tự duyệt sau
* 2
* 5
* 40 30 35 80 100
* 35 30 100 80 40
* 8
* 40 30 32 35 80 90 100 120
* 35 32 30 120 100 90 80 40

1. Bài 15: Kiểm tra cây BST

Cho mảng a[], n phần tử. Đưa ra 1 nếu nó được duyệt theo inOrder của cây BST, ngược lại in ra 0

* Input:
* Số lượng test
* Gồm 2 dòng, N và số lượng phần tử (1<=T<=100,1<=N<=1000, 1<=val<=1e4)
* Output: 1 or 0
* 3
* 5
* 10 20 30 40 50
* 1
* 6
* 90 80 100 70 40 30
* 0
* 3
* 1 1 2
* 0

1. Bài 16: Node trung gian của cây BST

Cho dãy gồm N số theo thứ tự PreOrder của BST. Đưa ra số node trung gian

* Input:
* Số lượng test
* N và N phần tử (1<=T<=100, 1<=N<=1000, 1<=val<=1e4)
* Output: số lượng node trung gian
* 2
* 6
* 10 5 1 7 40 50
* 3
* 11
* 30 20 15 25 23 28 40 35 33 38 45
* 5

1. Bài 17: Cây AVL 1

Xây dựng cây nhị phân tìm kiếm cân bằng từ dãy a[]. Đưa ra node gốc của cây.

* Input:
* Số lượng bộ test
* Gồm 2 dòng, N và số lượng phần tử ( T<=100, N<=1e6)
* Output: Đưa ra node gốc
* 2
* 11
* 40 28 45 38 33 15 25 20 23 35 30
* 30
* 10
* 1 2 3 4 5 6 7 8 9 10
* 5

1. Bài 18: Cây AVL 2

Đưa ra thứ tự duyệt trước PreOrder của cây AVL

* Input:
* Số lượng test
* N và N phần tử
* Output: Thứ tự preOrder
* 2
* 11
* 40 28 45 38 33 15 25 20 23 35 30
* 30 23 15 20 25 28 38 33 35 40 45
* 10
* 1 2 3 4 5 6 7 8 9 10
* 5 2 1 3 4 8 6 7 9 10

1. Bài 19: Cây AVL 3

Đưa ra thứ tự duyệt sau PostOrder của cây AVL

* Input:
* Số lượng test
* N và N phần tử
* Output: Thứ tự postOrder
* 2
* 11
* 40 28 45 38 33 15 25 20 23 35 30
* 20 15 28 25 23 35 33 45 40 38 30
* 10
* 1 2 3 4 5 6 7 8 9 10
* 1 4 3 2 7 6 10 9 8 5

1. Bài 20: Đếm số lượng node lá

Xây dựng cây AVL và đếm số node lá

* Input:
* Số lượng test
* N và N phần tử
* Output: Số lượng node lá
* 2
* 11
* 40 28 45 38 33 15 25 20 23 35 30
* 4
* 10
* 1 2 3 4 5 6 7 8 9 10
* 4

40 28 45 38 33 15 25 20 23 35 30

2

5

40 30 35 80 100

8

40 30 32 35 80 90 100 120