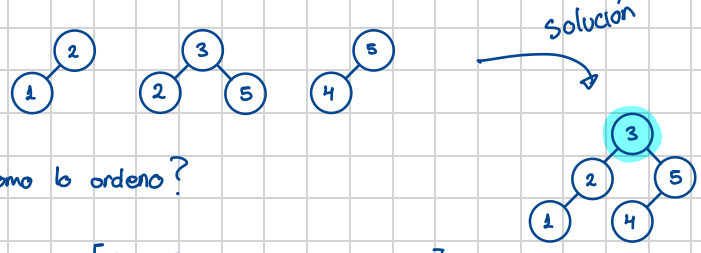
1932

Input: list[BST1, ...] , len(BSTn) ≤ 3

output: BST.root ->



¿Como lo ordeno?

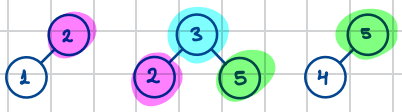
trees = [[2,1], [3,2,5], [5,4]]

tree -> sublista de trees

ANALIZANDO

- Cada tree es un BST donde tree[0] es su root
- Si unimos todos los tree solo existira un UNICO ROOT o FINAL ROOT
  - ↳ Por lo tanto:
    todo tree[0] (root de la sublista)

ejem:



debe poder 'reemplazar' a una hoja de otro arbol, con excepción del final_root

SOLUCIÓN

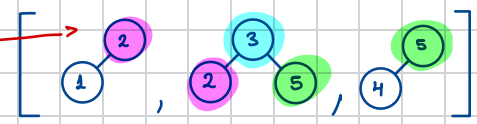
Ya que sabemos que cada root se debe poder insertar en otro arbol la solución seria Coger un root cualquiera y buscar dicho valor en todas las hojas de los otros arboles.

¡COSTOSO!

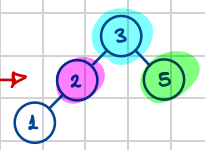
Algoritmo.

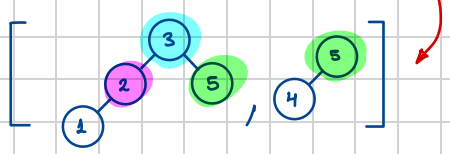
```
for tree in trees        // iterar por toda la lista:
    root = tree[0]       // Ya los pasamos de sublistas a arboles.
    node = buscar root->value  en las hojas de los demas arboles.  // O(n)
    node = root
    delete tree si node exist
si al terminar el for len(trees) > 1 , return = []
else:
    trees[0].root
```

Selecciona al (2)



busca el 2 en las otras hojas [2,5], [4]





$O(n^2)$

¡Carisimo!

Como se realizaran busquedas sucesivas, deberiamos usar una estructura de datos que beneficie dicha busqueda.

¿HASH? ¿AVL? ¿BST?

¿Que guardaria?

direcciones de las hojas existentes para reemplazarlas en O(1) despues de ubicarlas, si guardamos solo su valor, sabriamos que existe una hoja con dicho valor, sin embargo buscarla donde esta seria problematico.