Tree Isomorphism

郭至軒（KuoE0）

KuoE0.tw@gmail.com
KuoE0.ch

Attribution-ShareAlike 3.0 Unported (CC BY-SA 3.0)

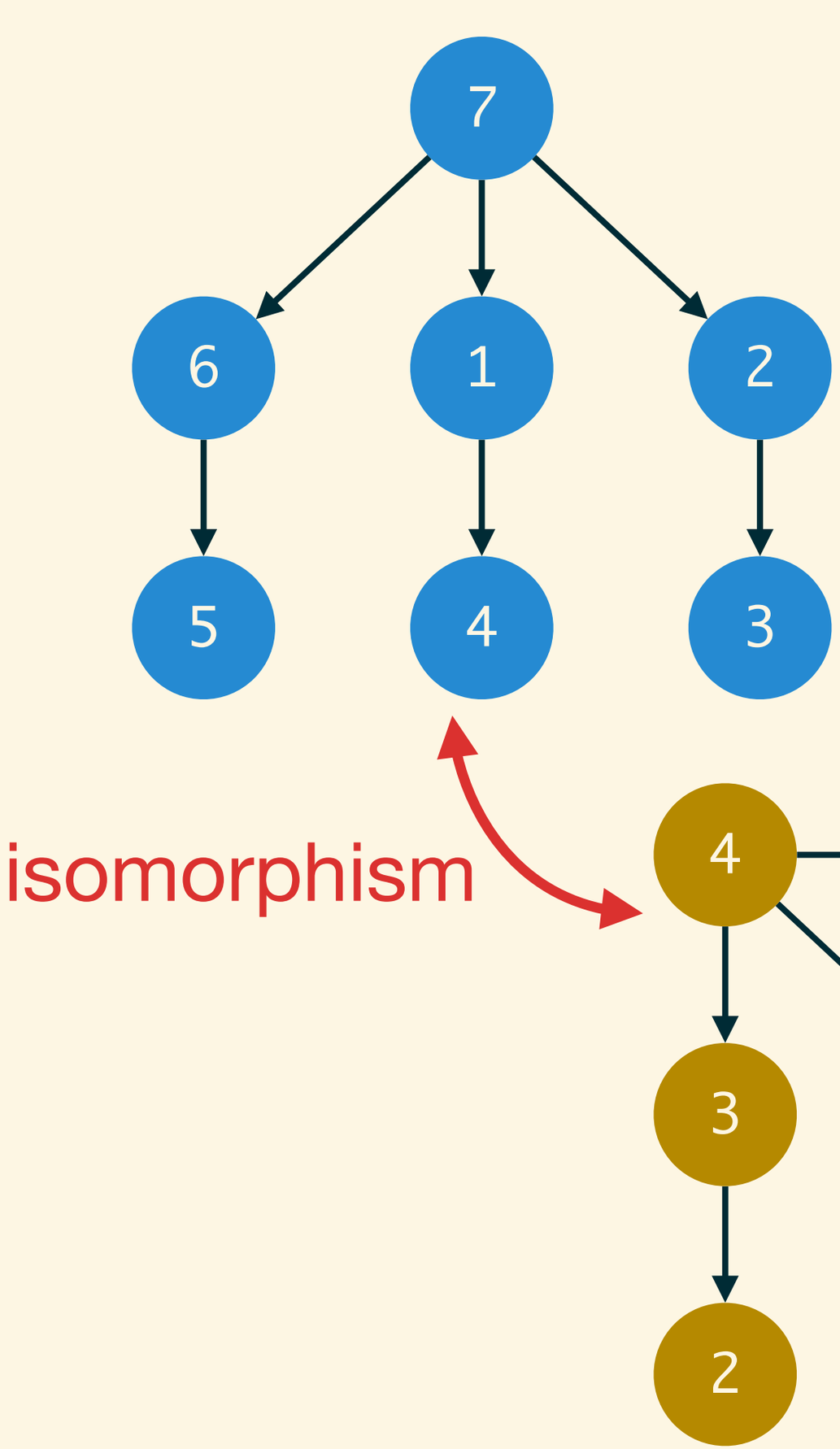
http://creativecommons.org/licenses/by-sa/3.0/

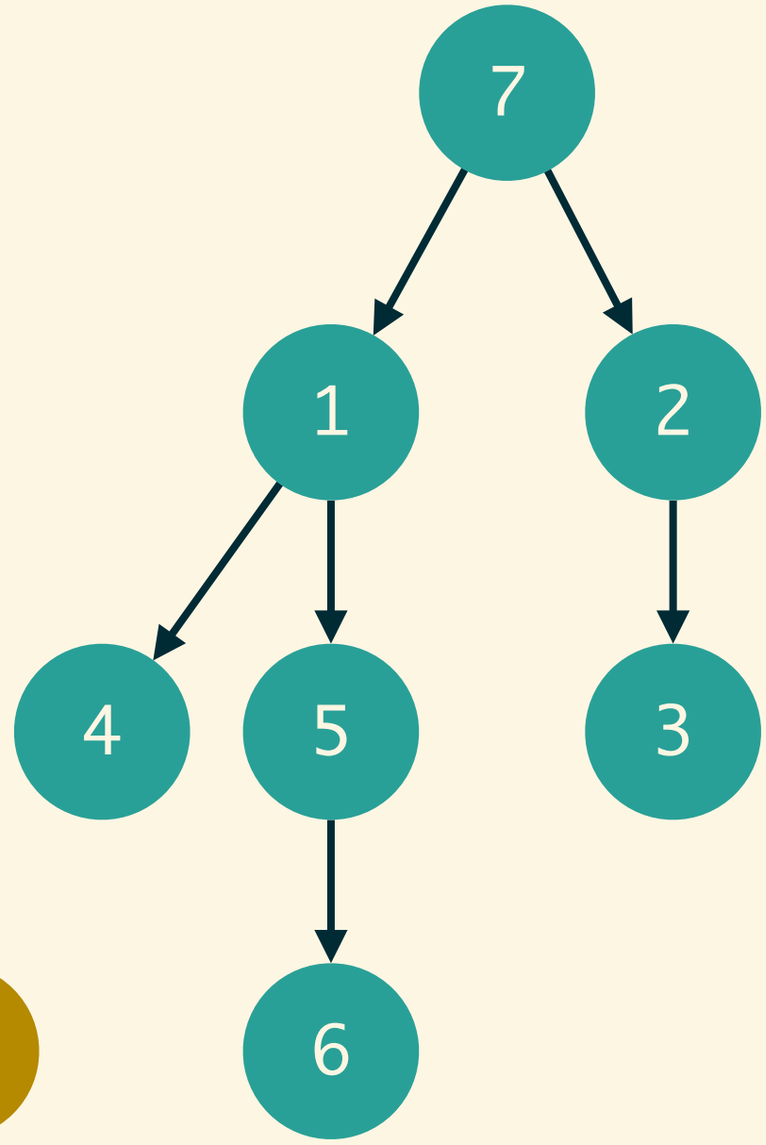
Latest update: May 1, 2013

Isomorphism

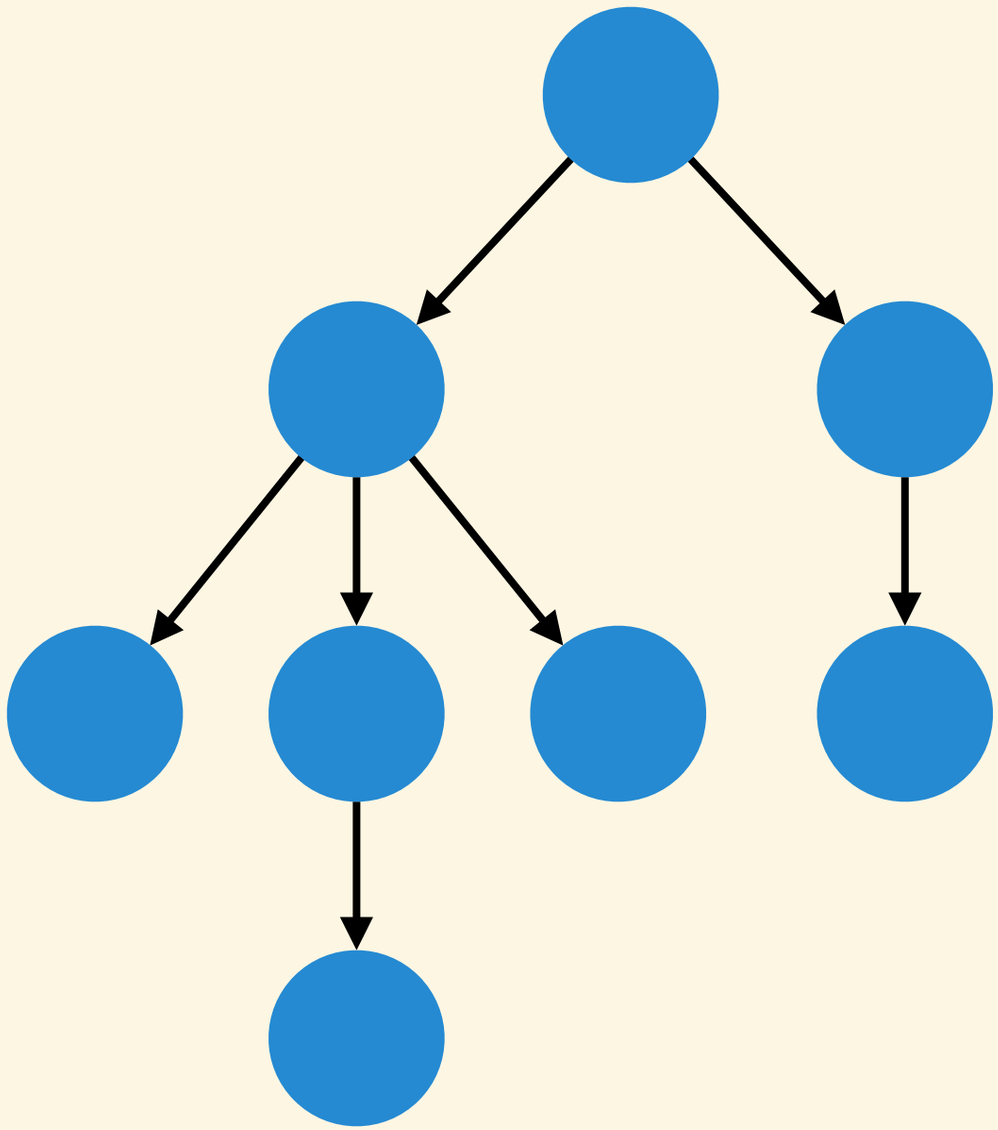
The structures of two trees are equal.

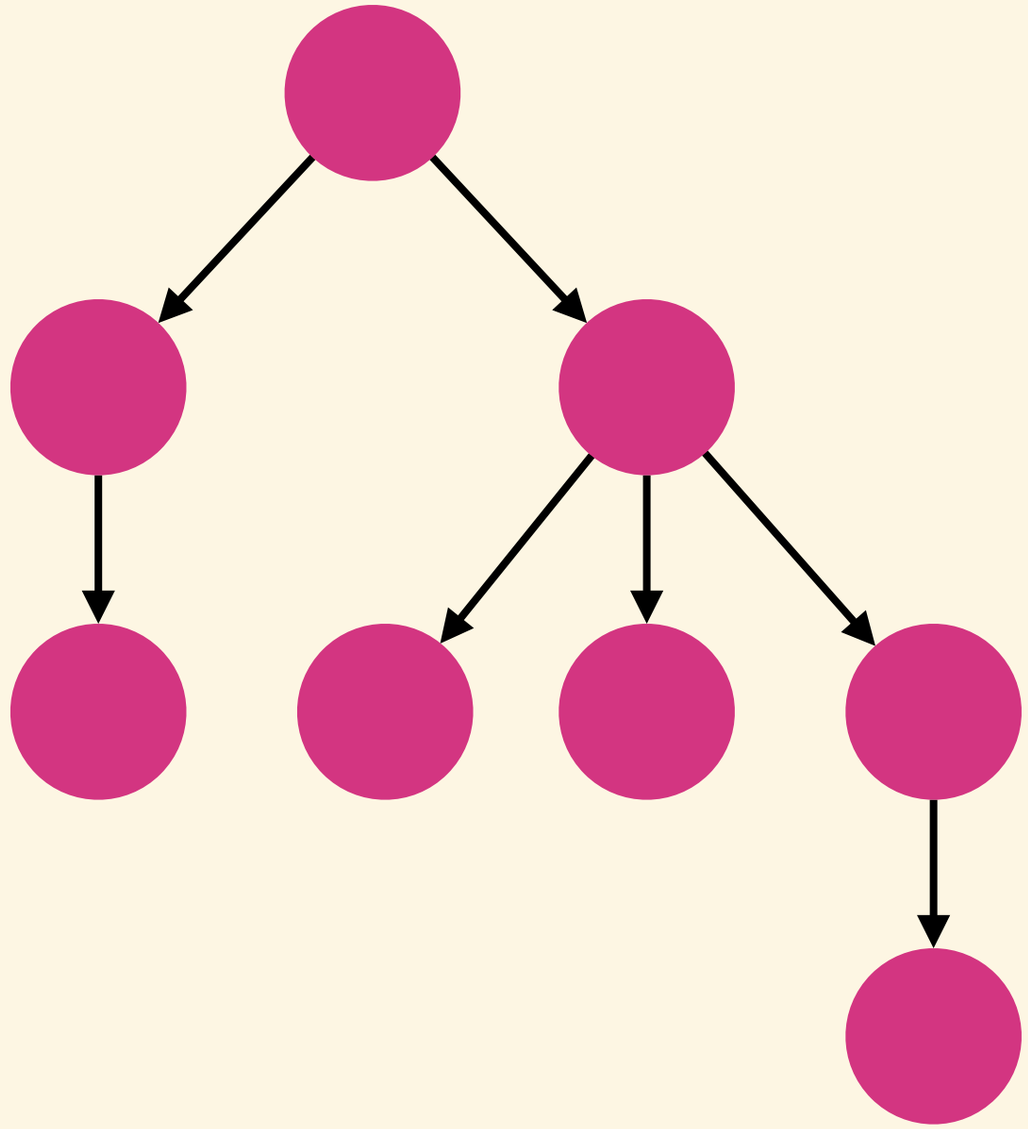
so abstract…

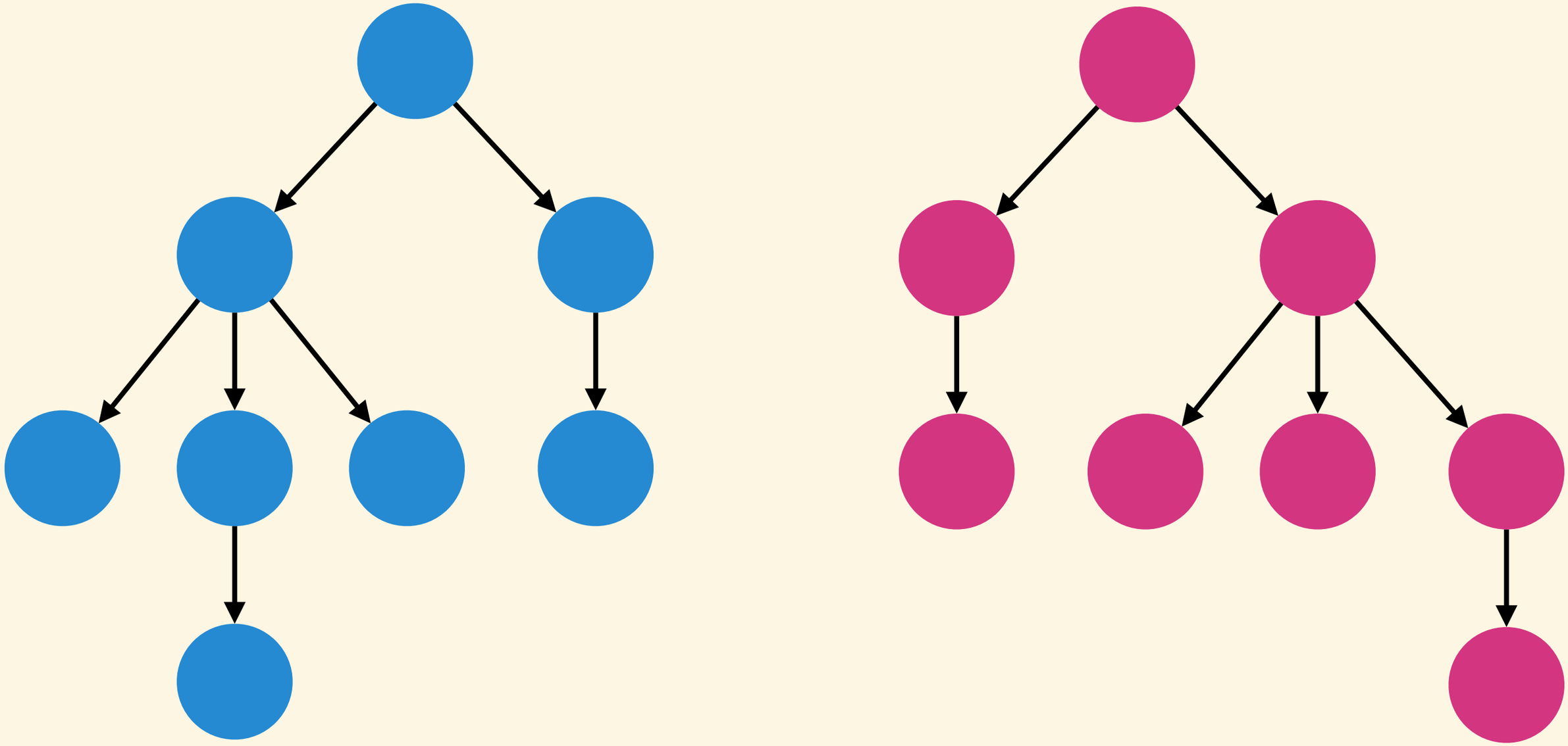

7
6
1
2
5
4
3
isomorphism
4
5
3
1
2
7
6


7
1
2
4
5
3
6

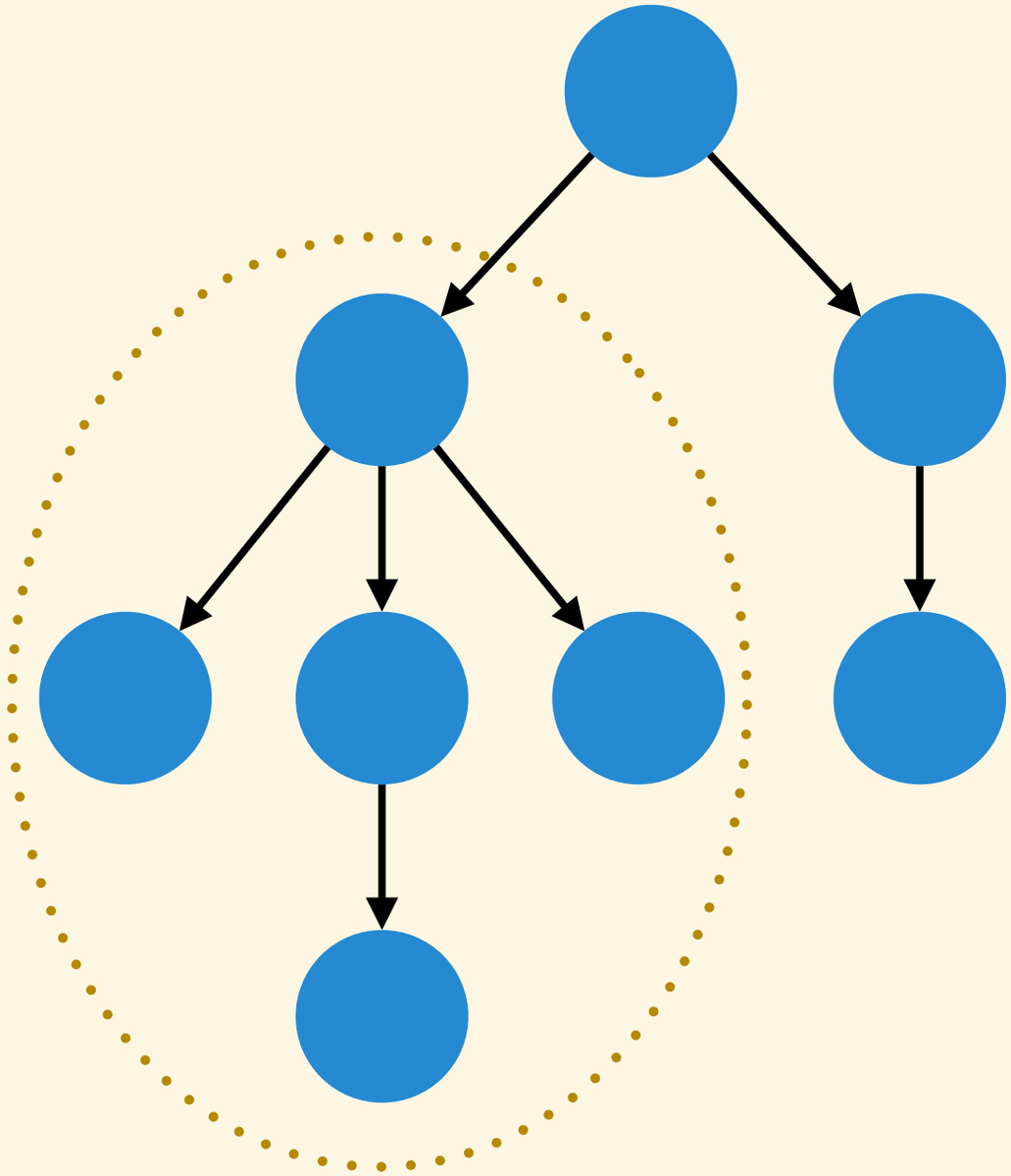
How to judge two rooted tree are isomorphic?

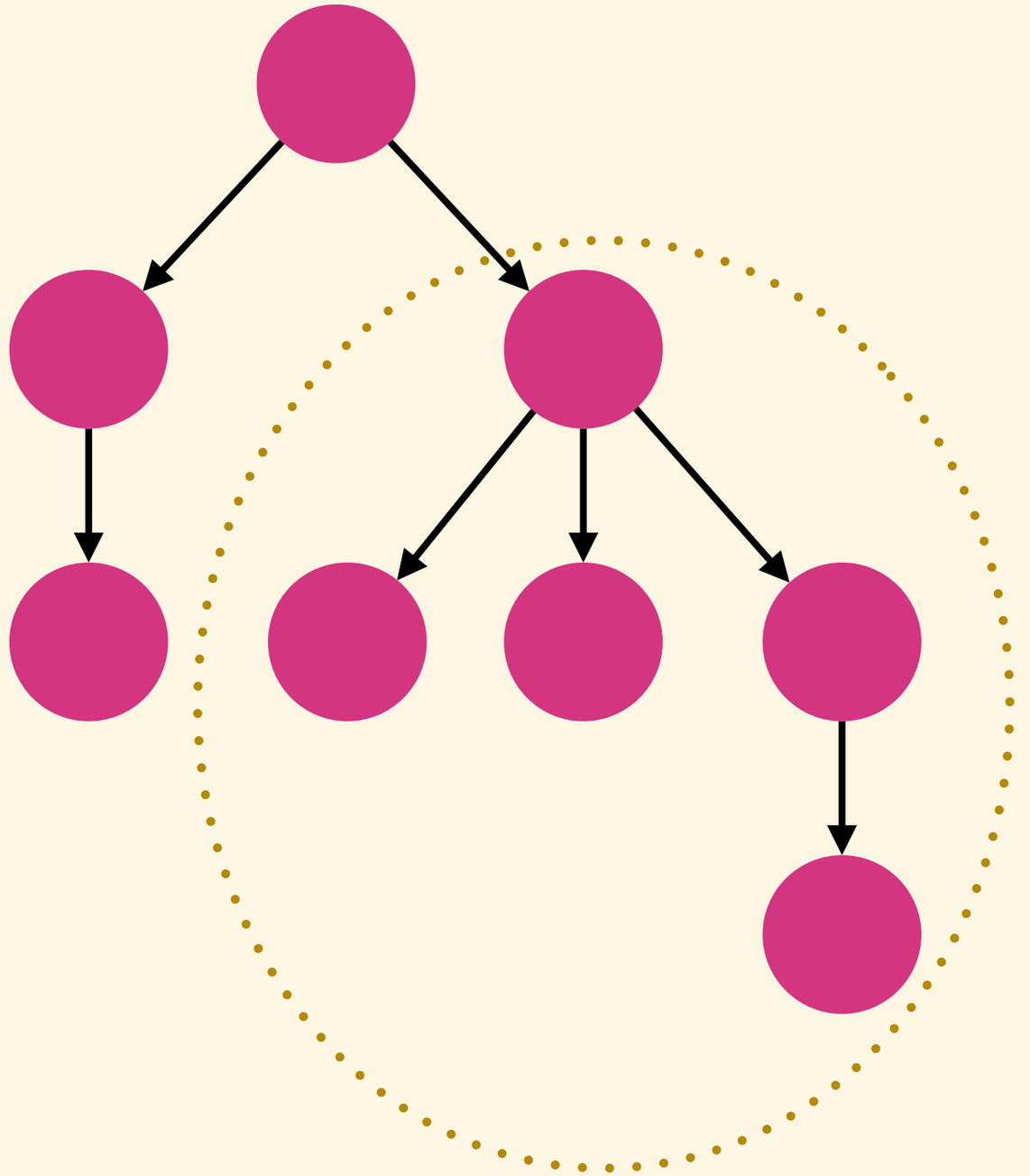




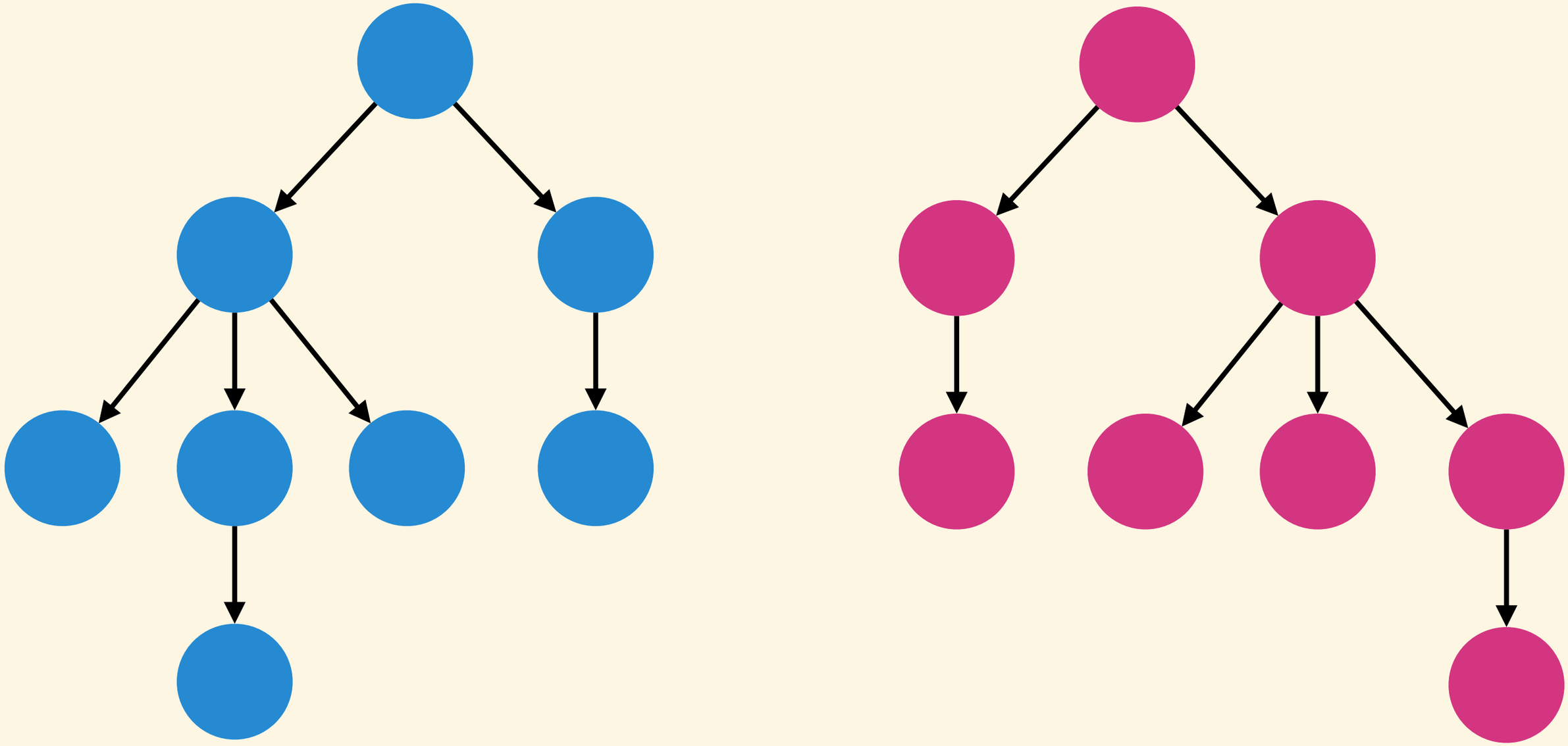


If the trees are isomorphic, all their sub-trees are also isomorphic.

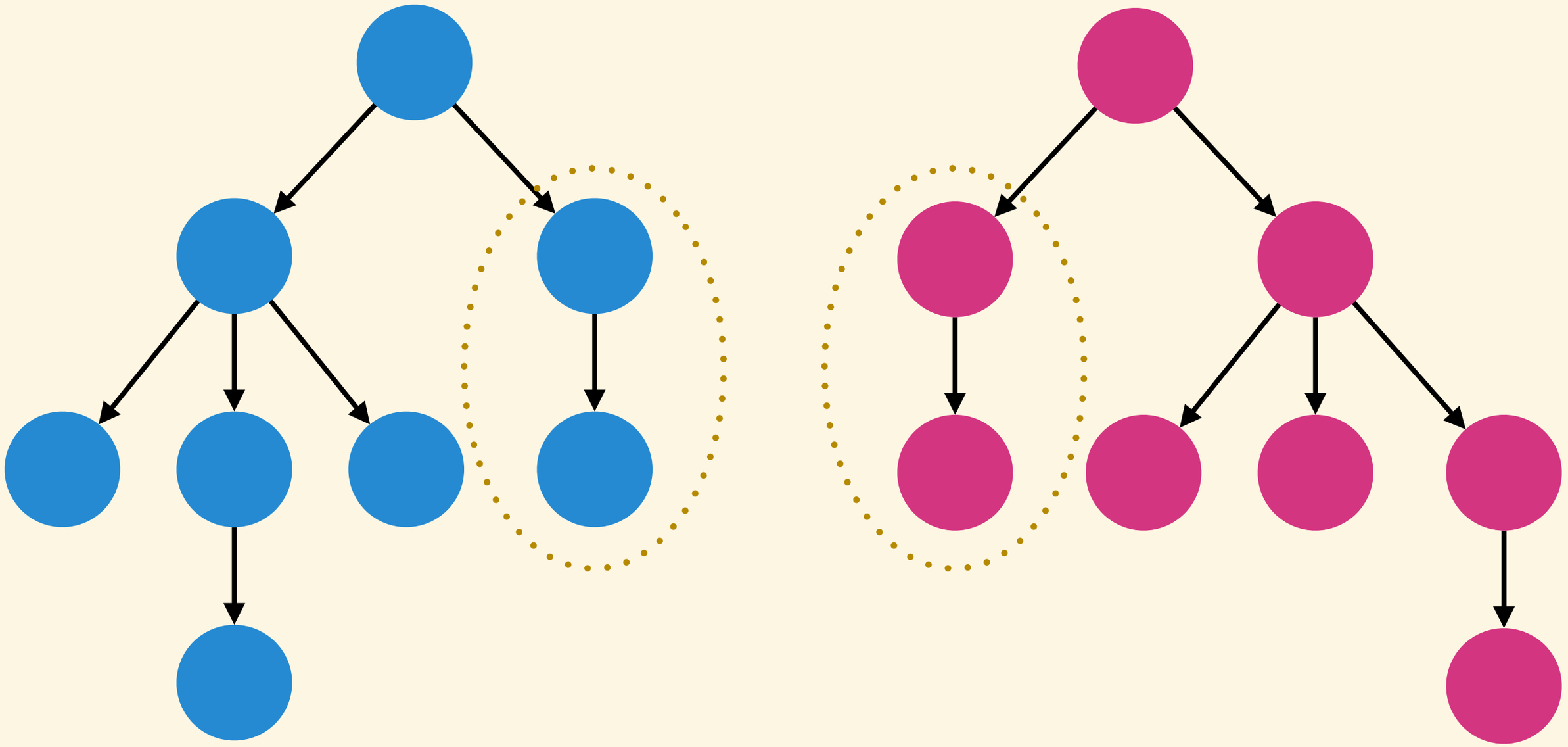




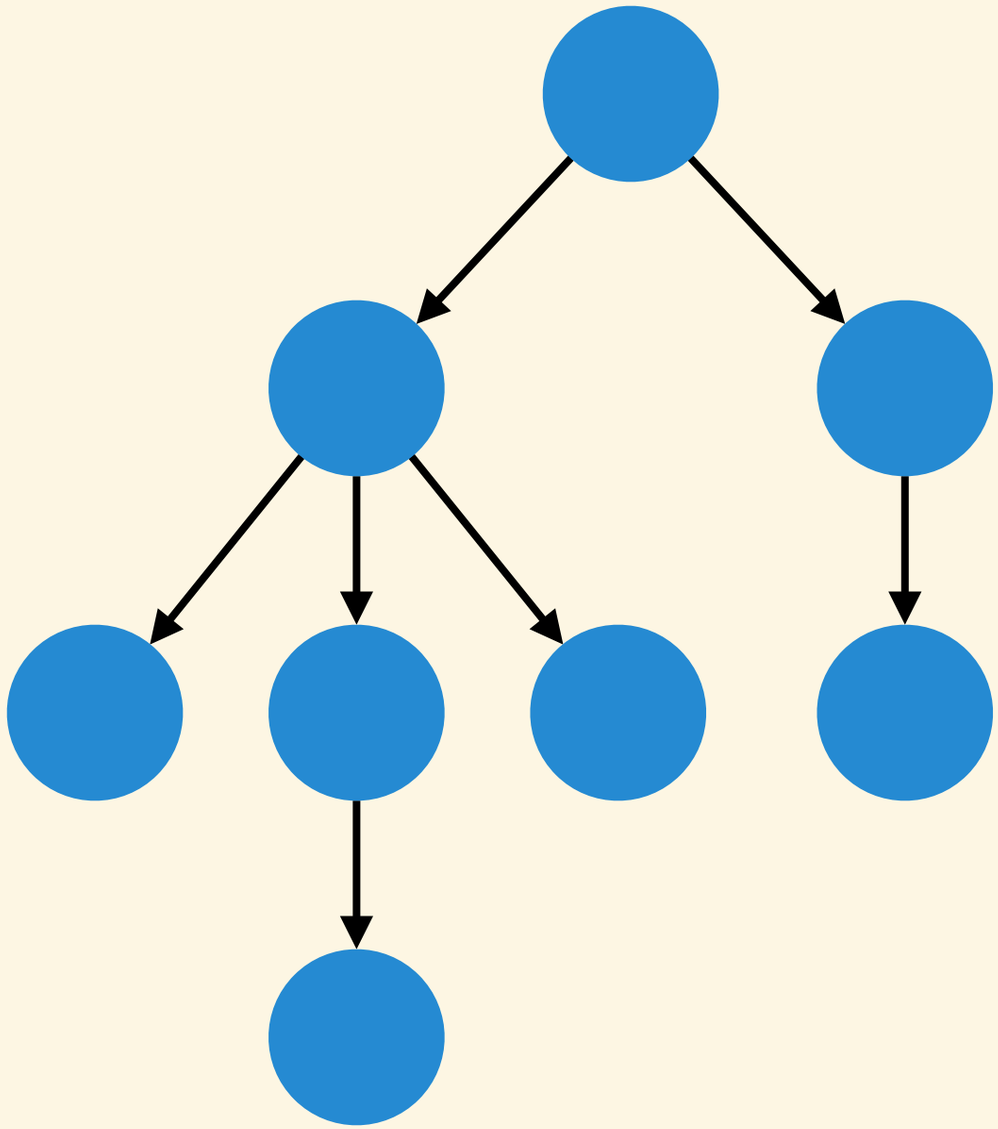
If the trees are isomorphic, all their sub-trees are also isomorphic.



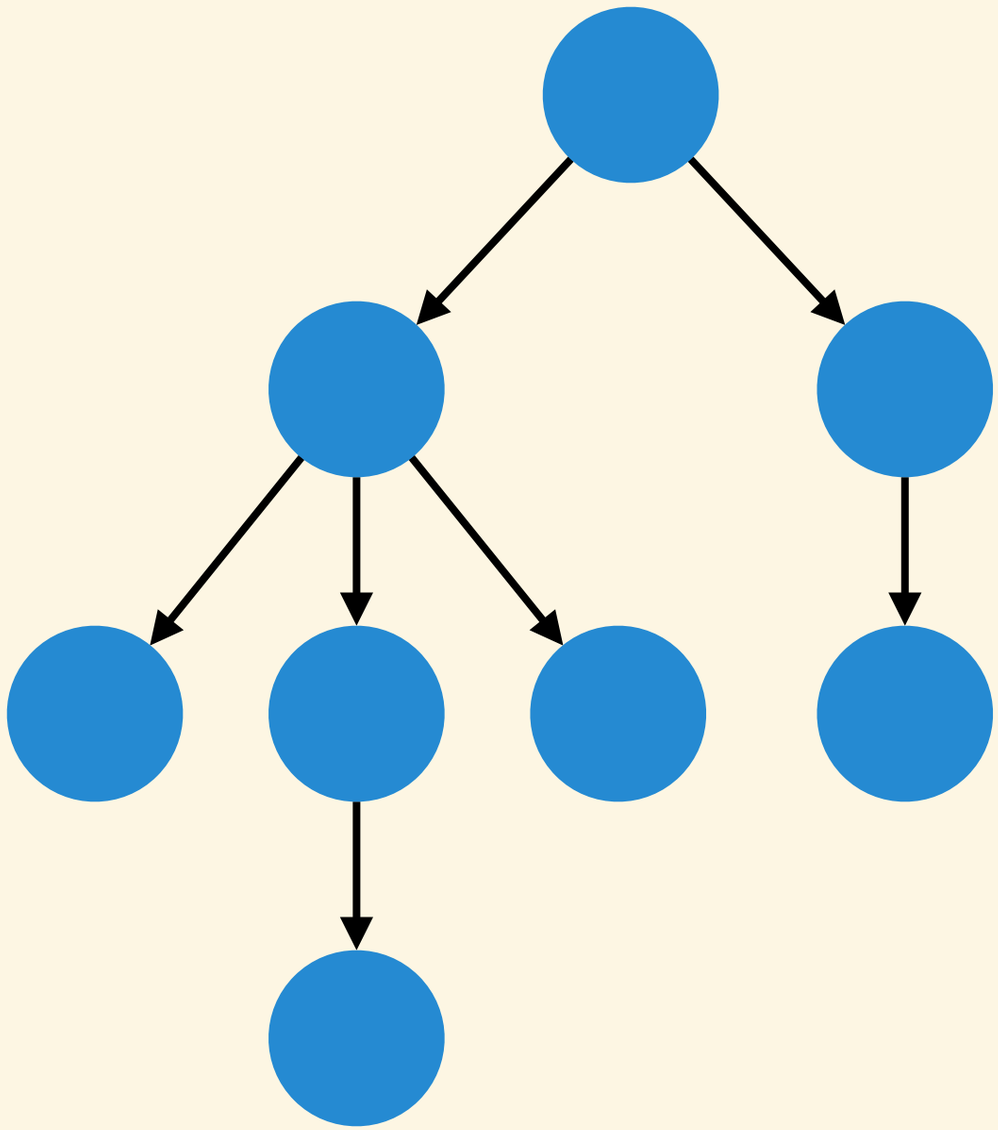
If the trees are isomorphic, all their sub-trees are also isomorphic.



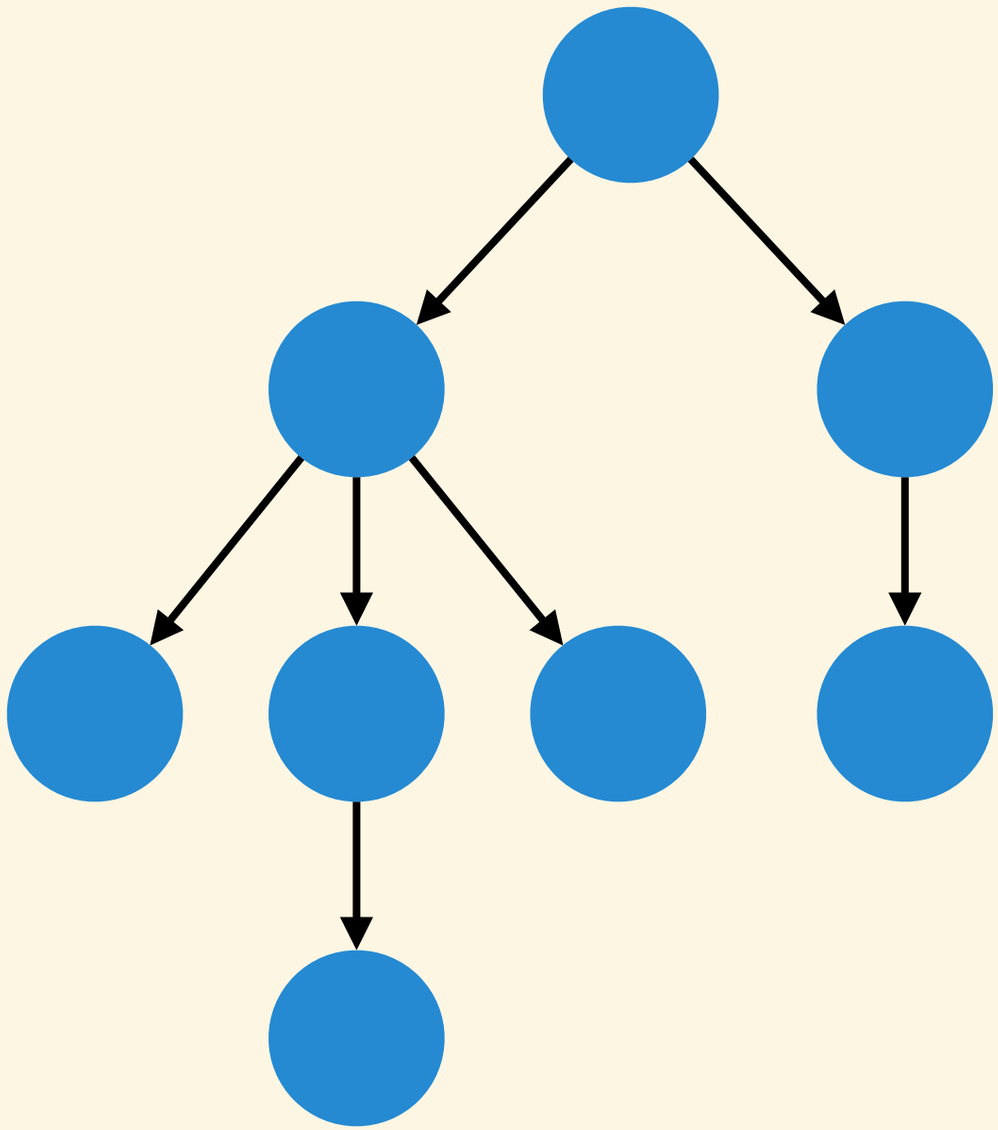
If the trees are isomorphic, all their sub-trees are also isomorphic.



Hash the tree

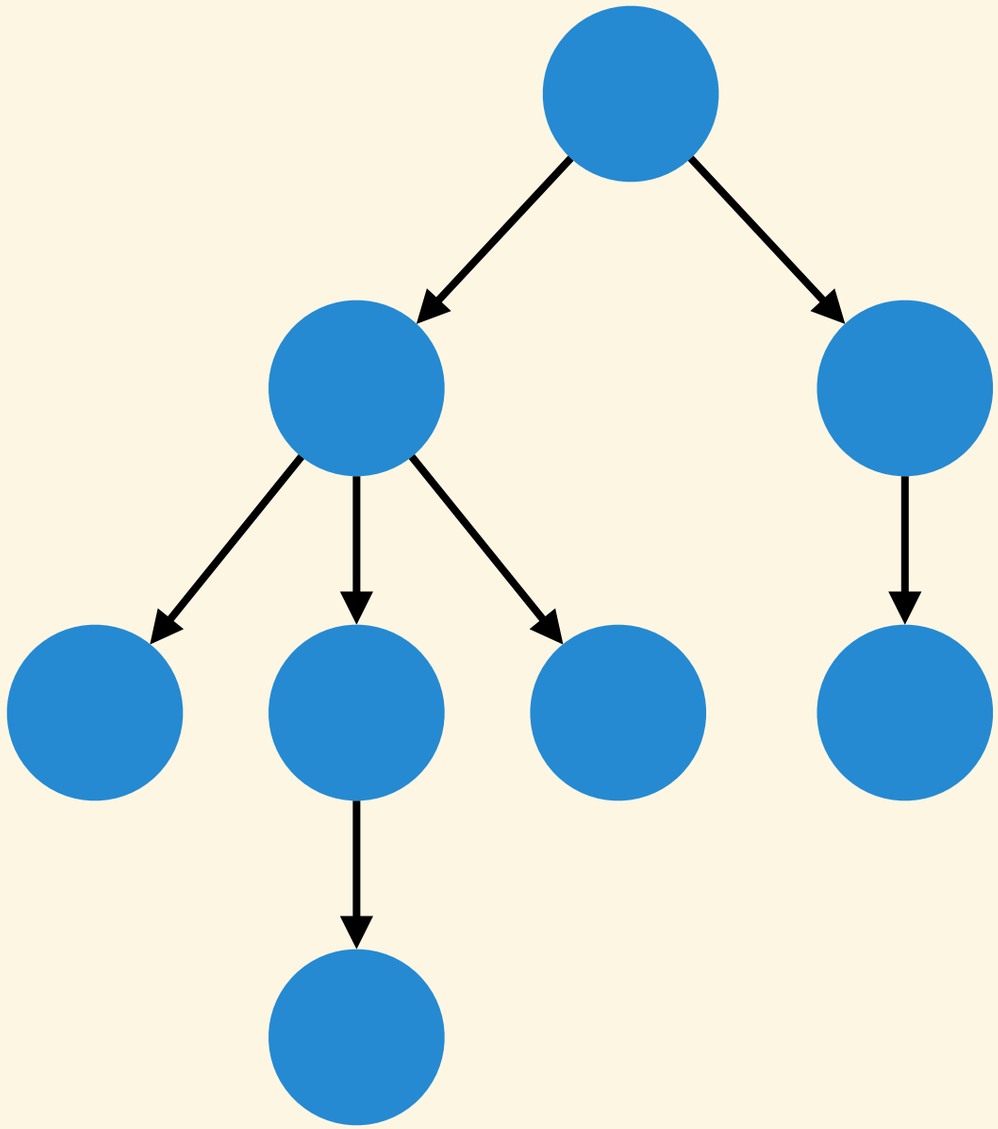


Hash all sub-tree



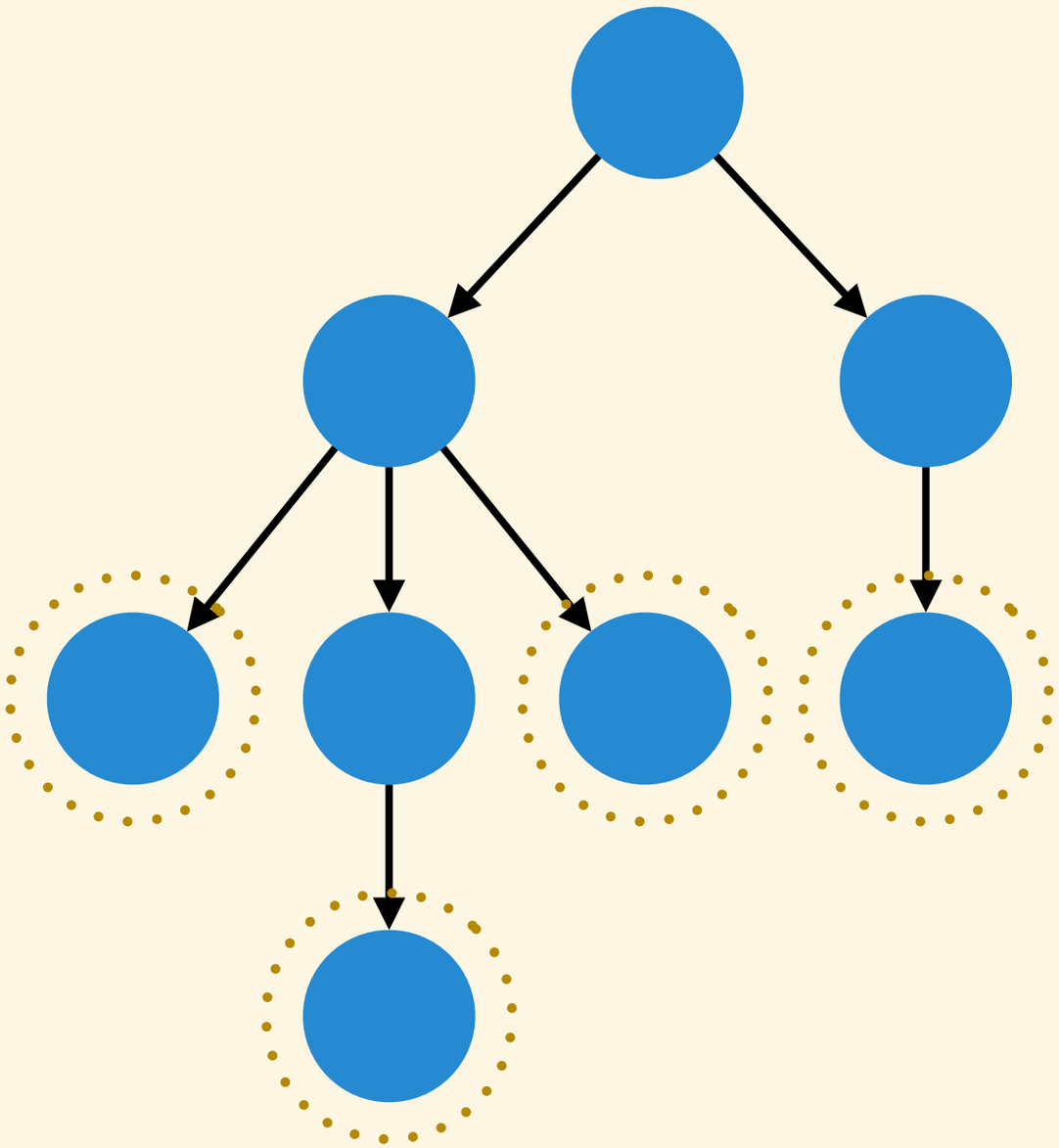
Hash all sub-tree

recursively



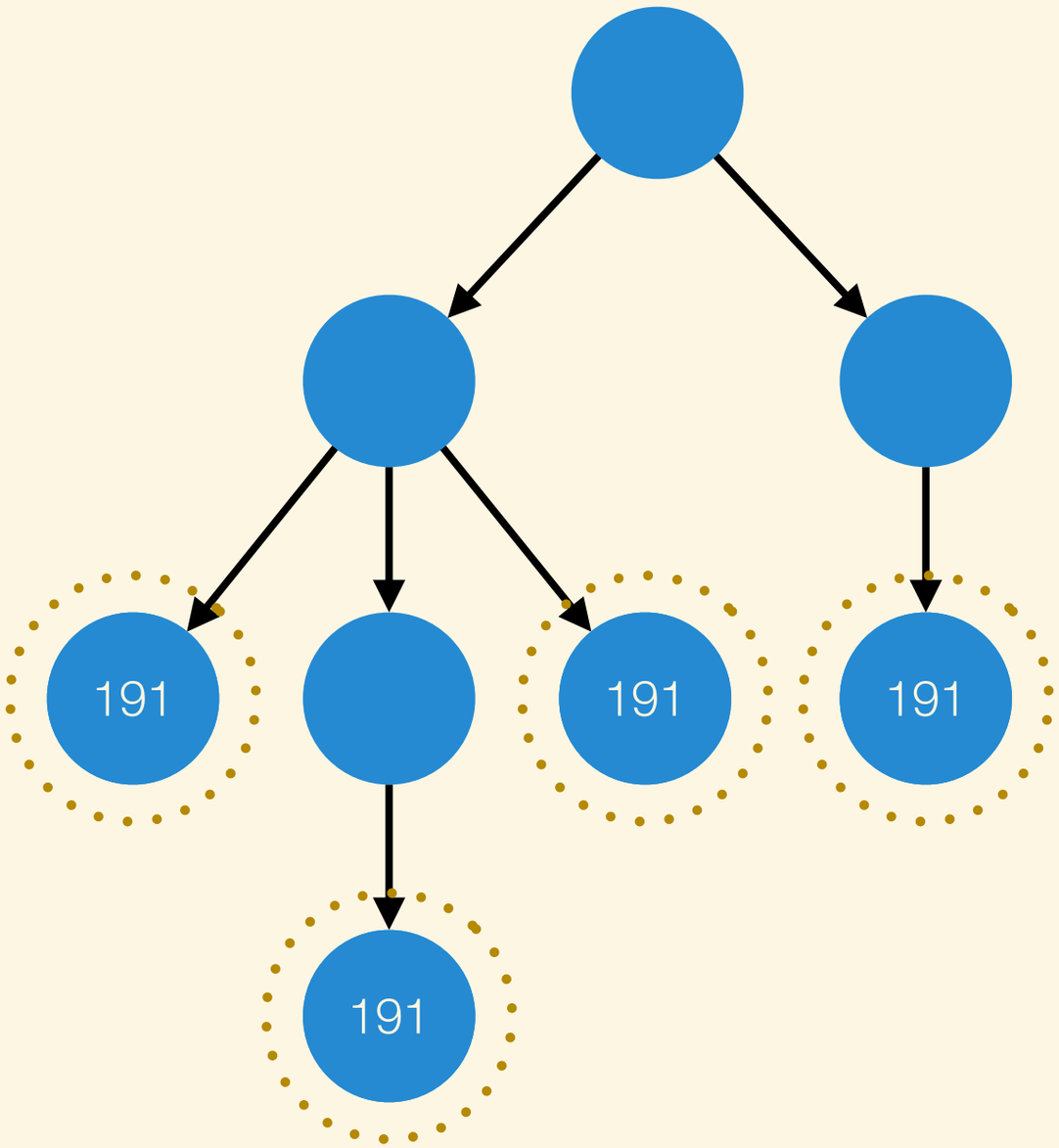
single vertex

initial value = 191



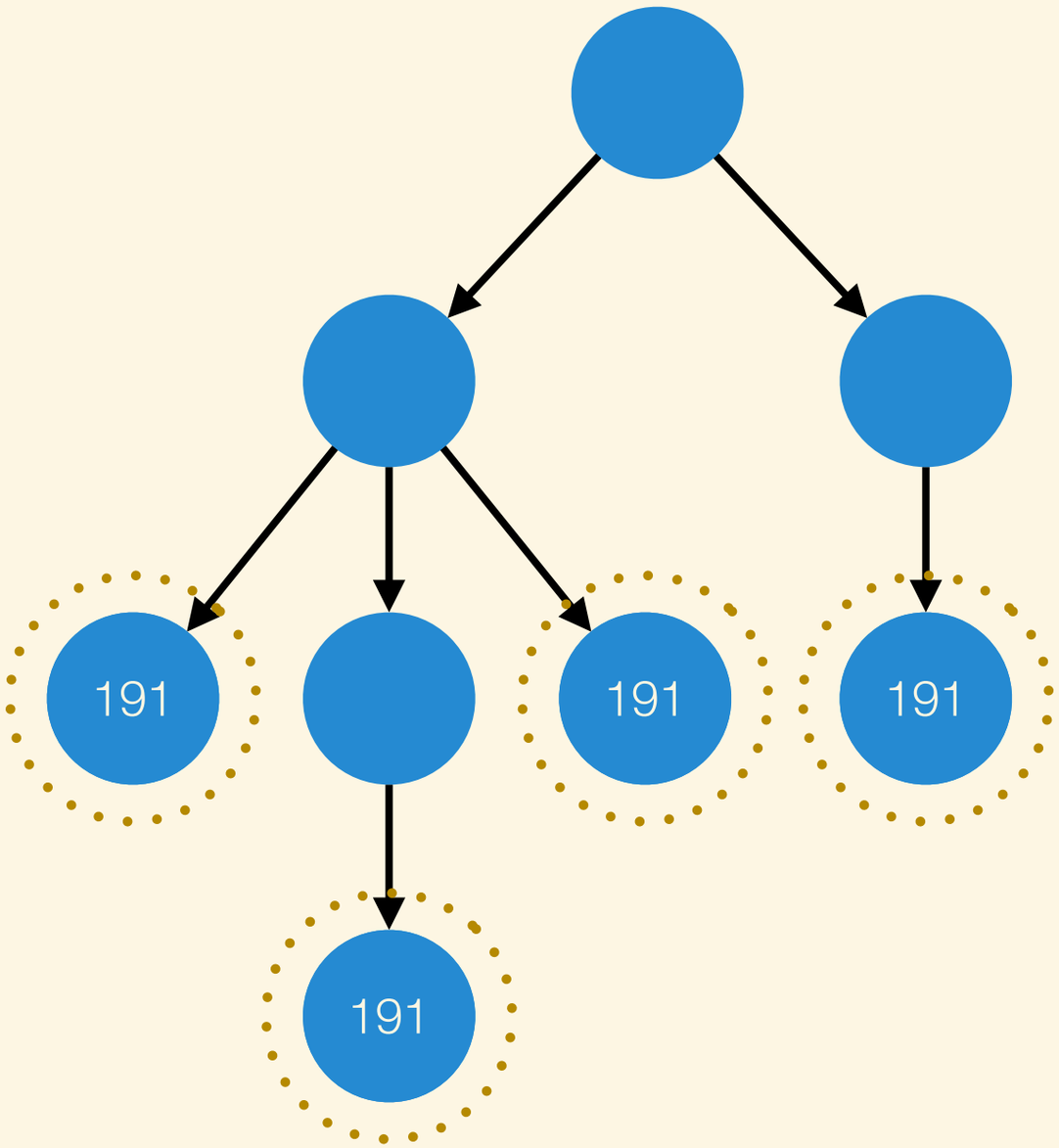
single vertex

initial value = 191



single vertex

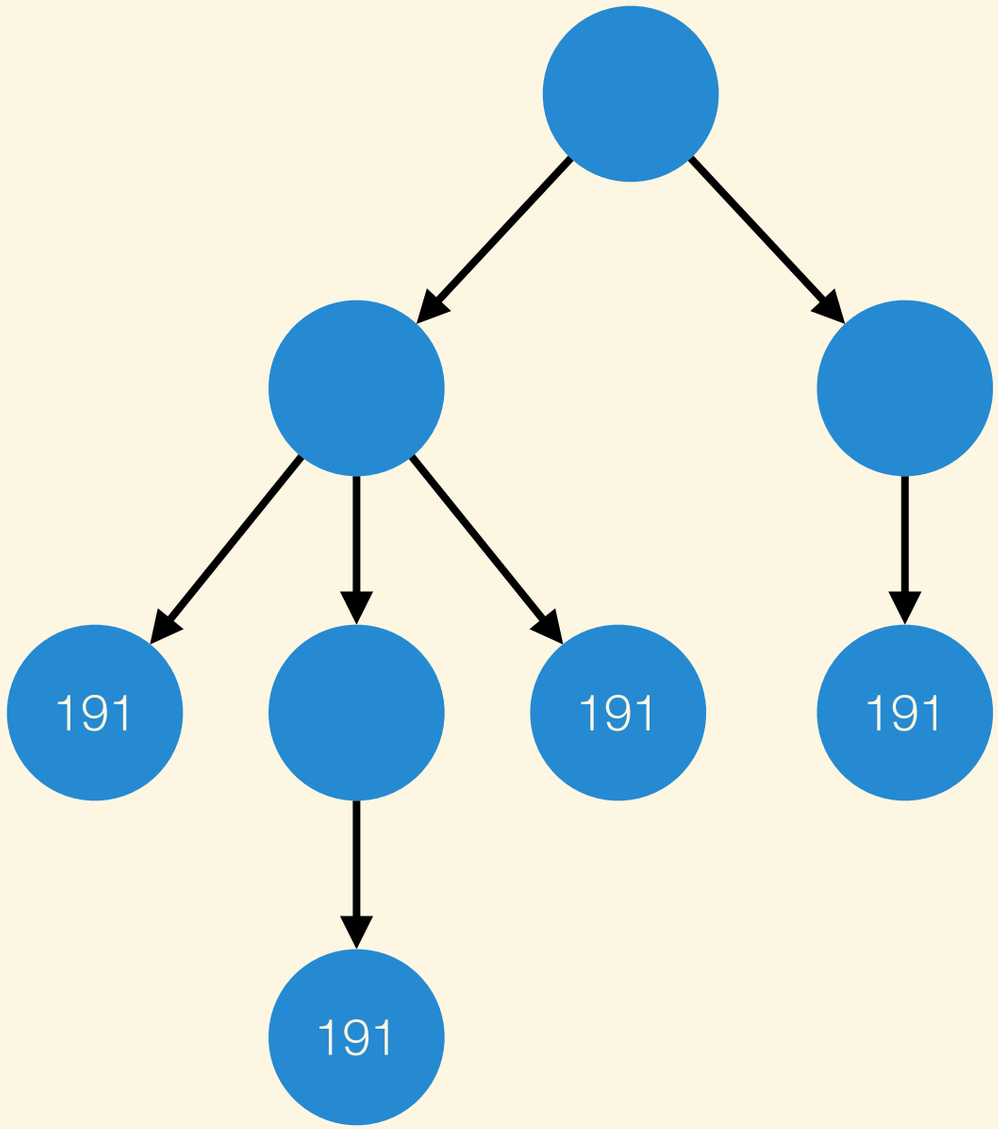
initial value = 191



define by yourself

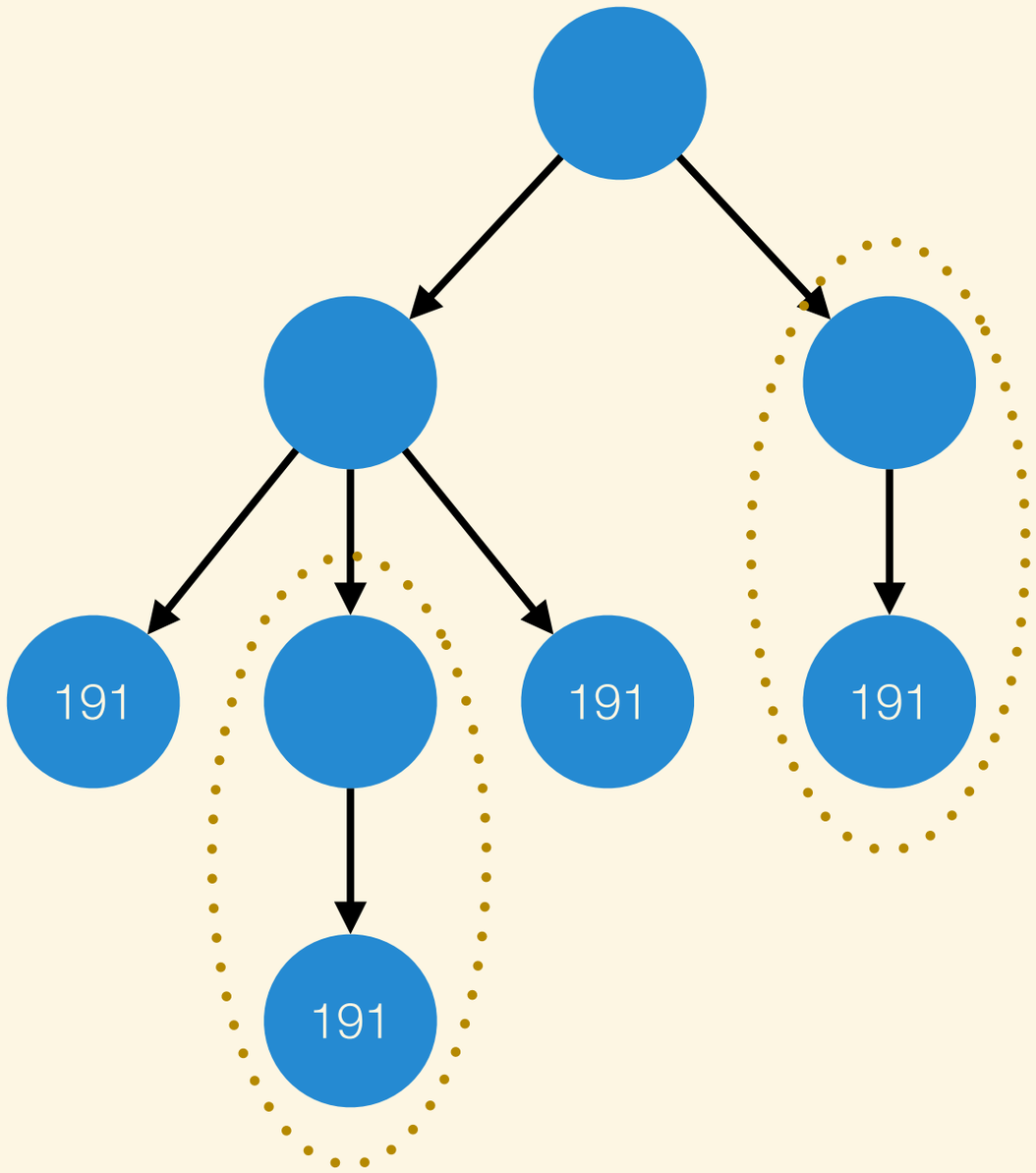
single vertex

initial value = 191



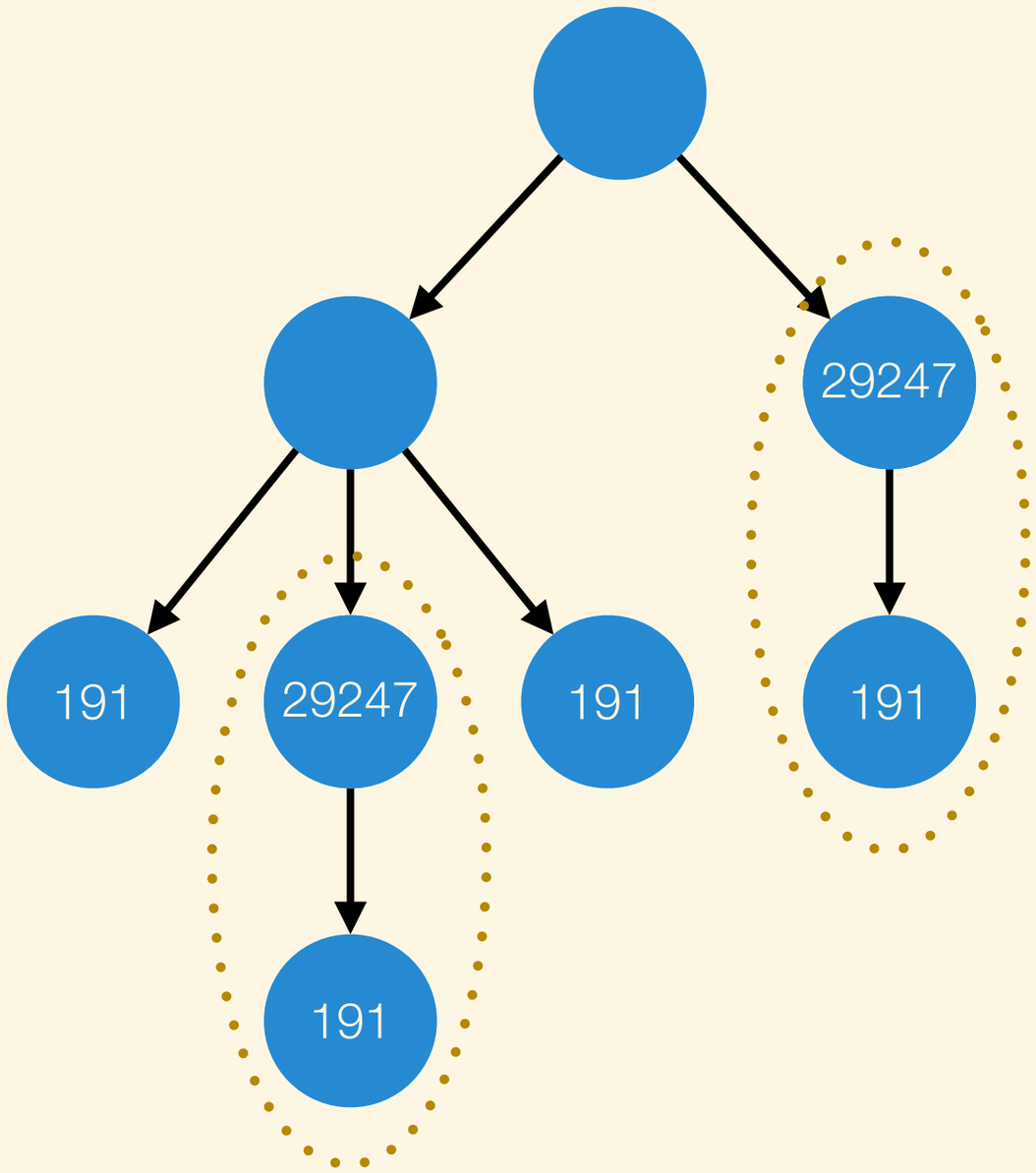
two-level sub-tree

child = (191)



two-level sub-tree

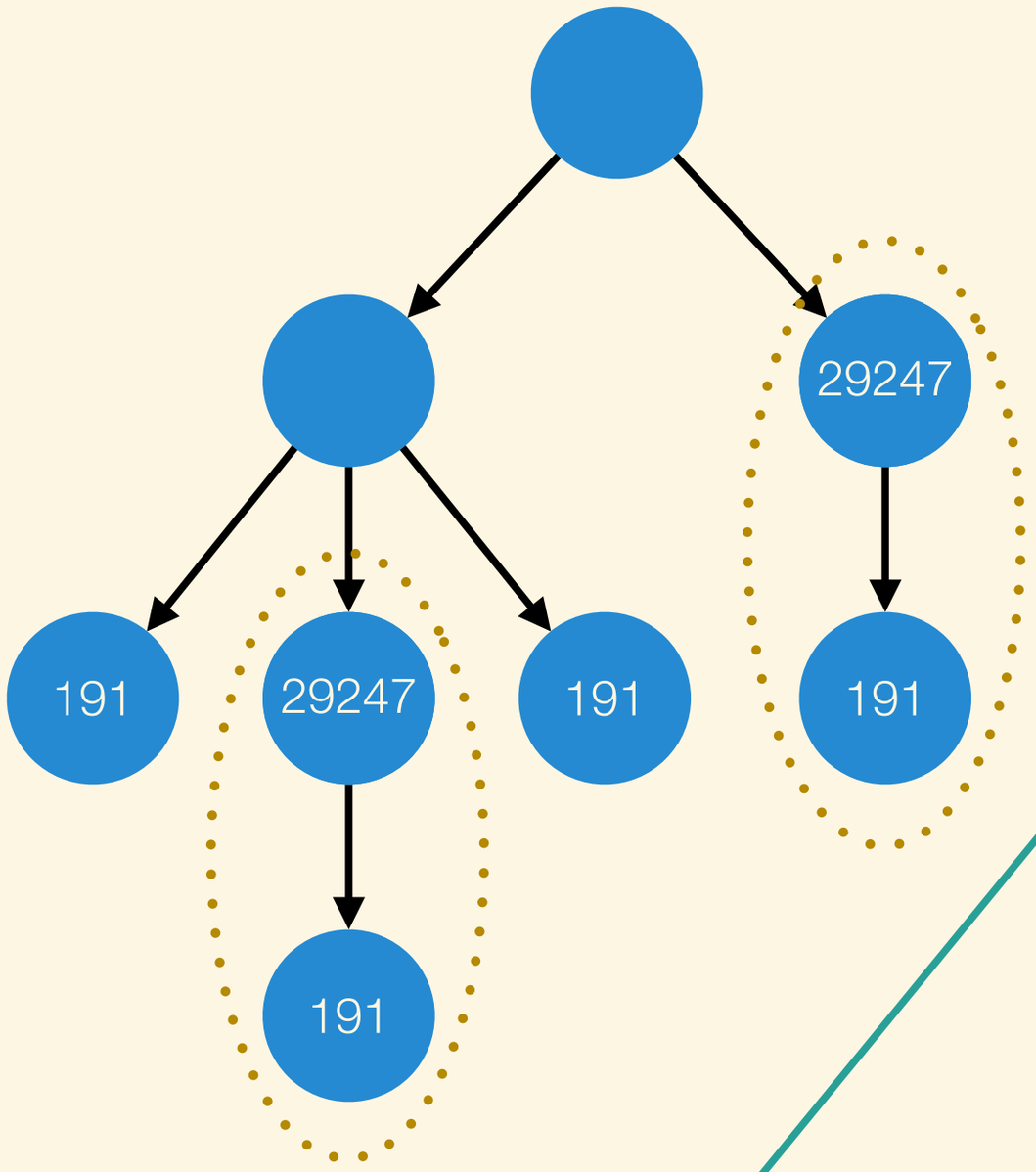
child = (191)



two-level sub-tree

child = (191)

(191 × 701 xor 191) mod 34943 = 29247

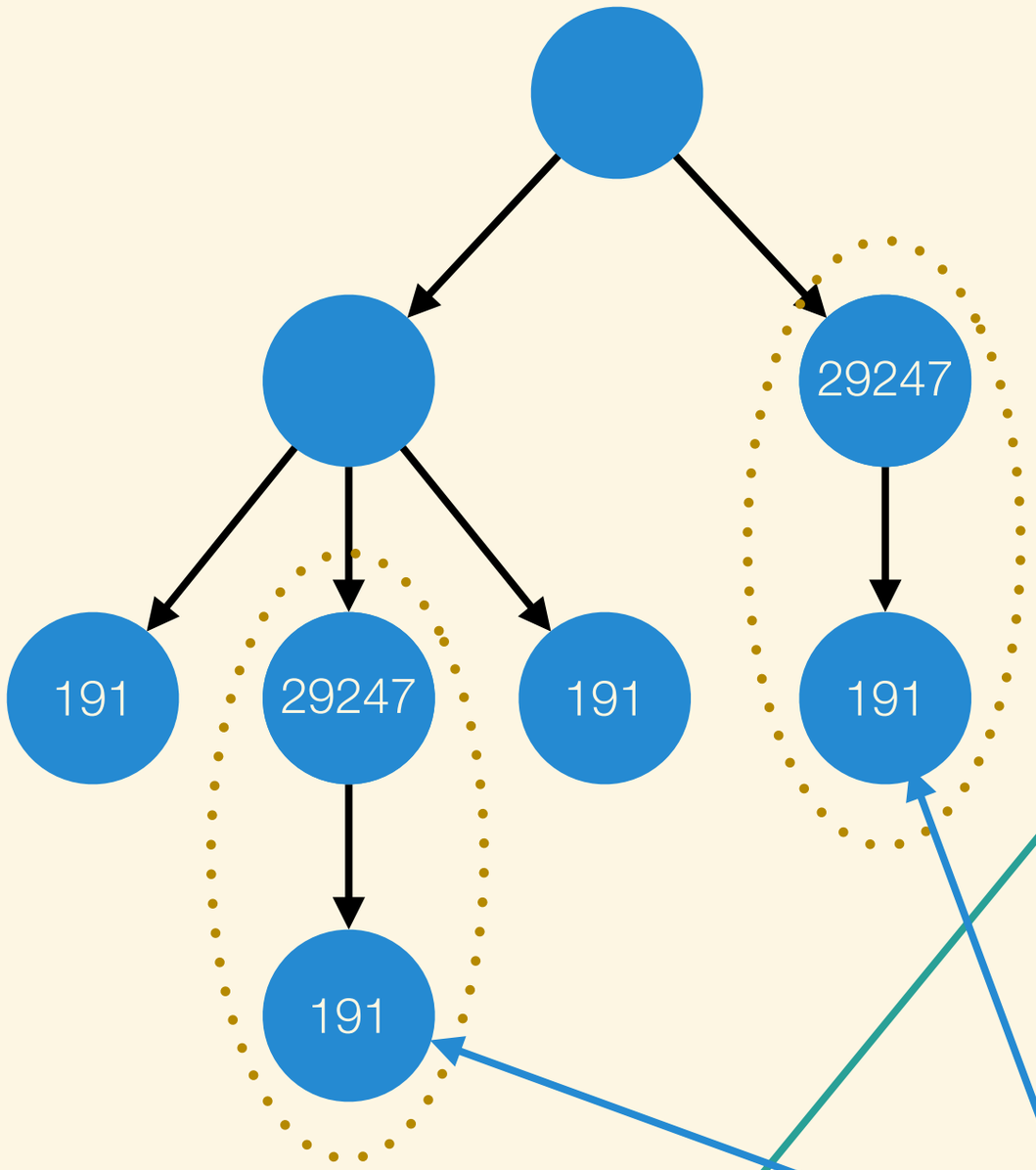


define by yourself

two-level sub-tree

child = (191)

(191 × 701 xor 191) mod 34943 = 29247

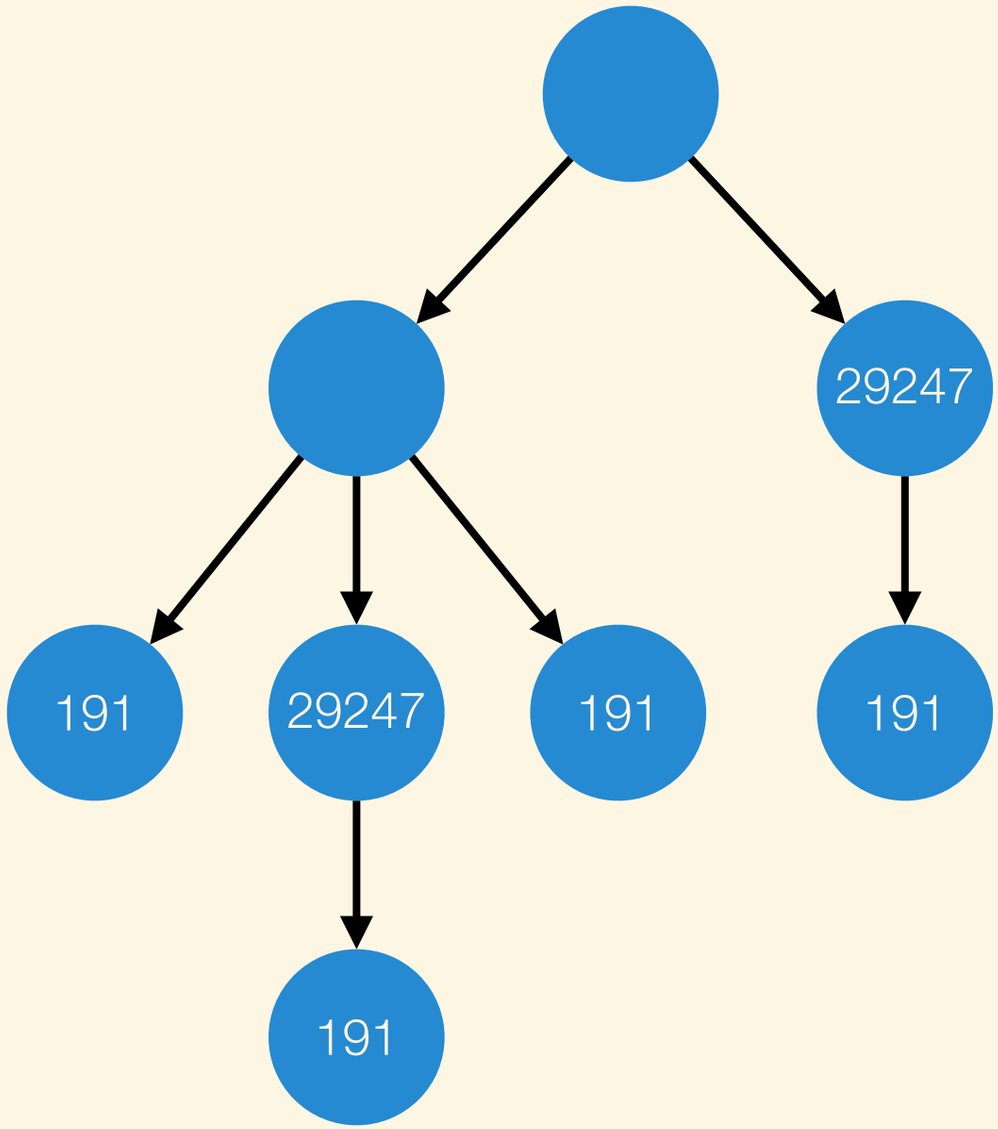


define by yourself

two-level sub-tree

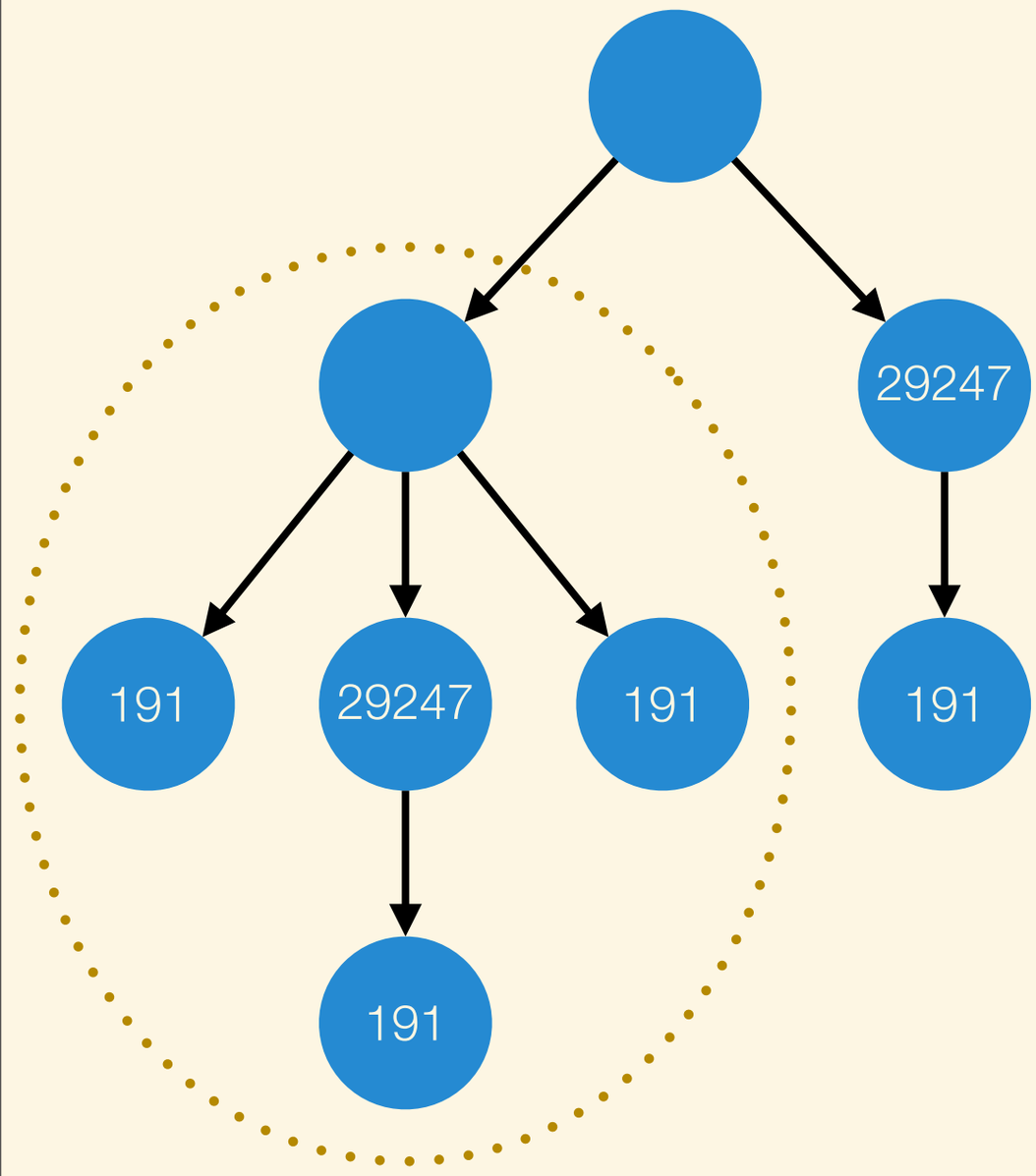
child = (191)

(191 × 701 xor 191) mod 34943 = 29247



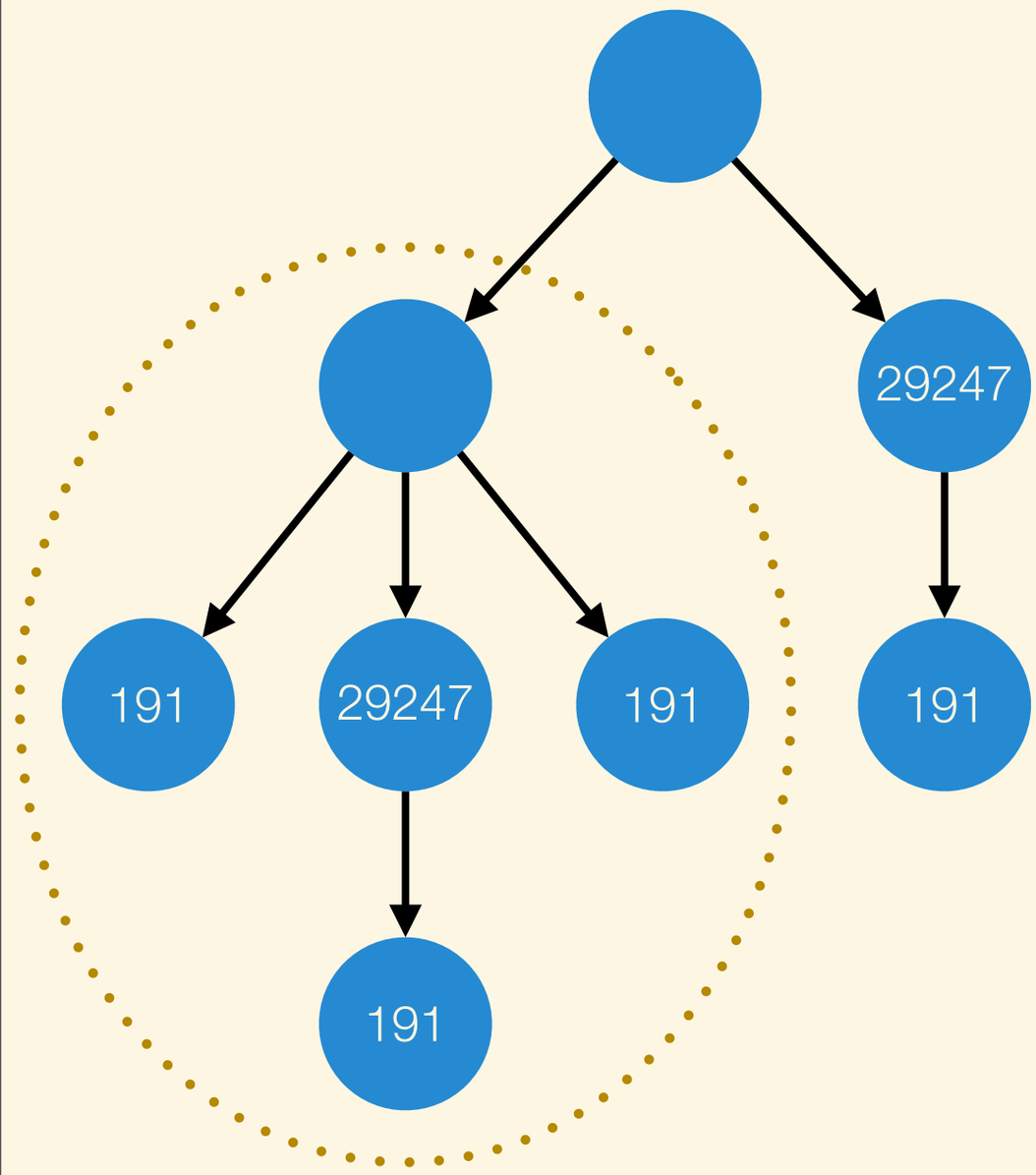
three-level sub-tree

child = (191,29247,191)



three-level sub-tree

child = (191,29247,191)

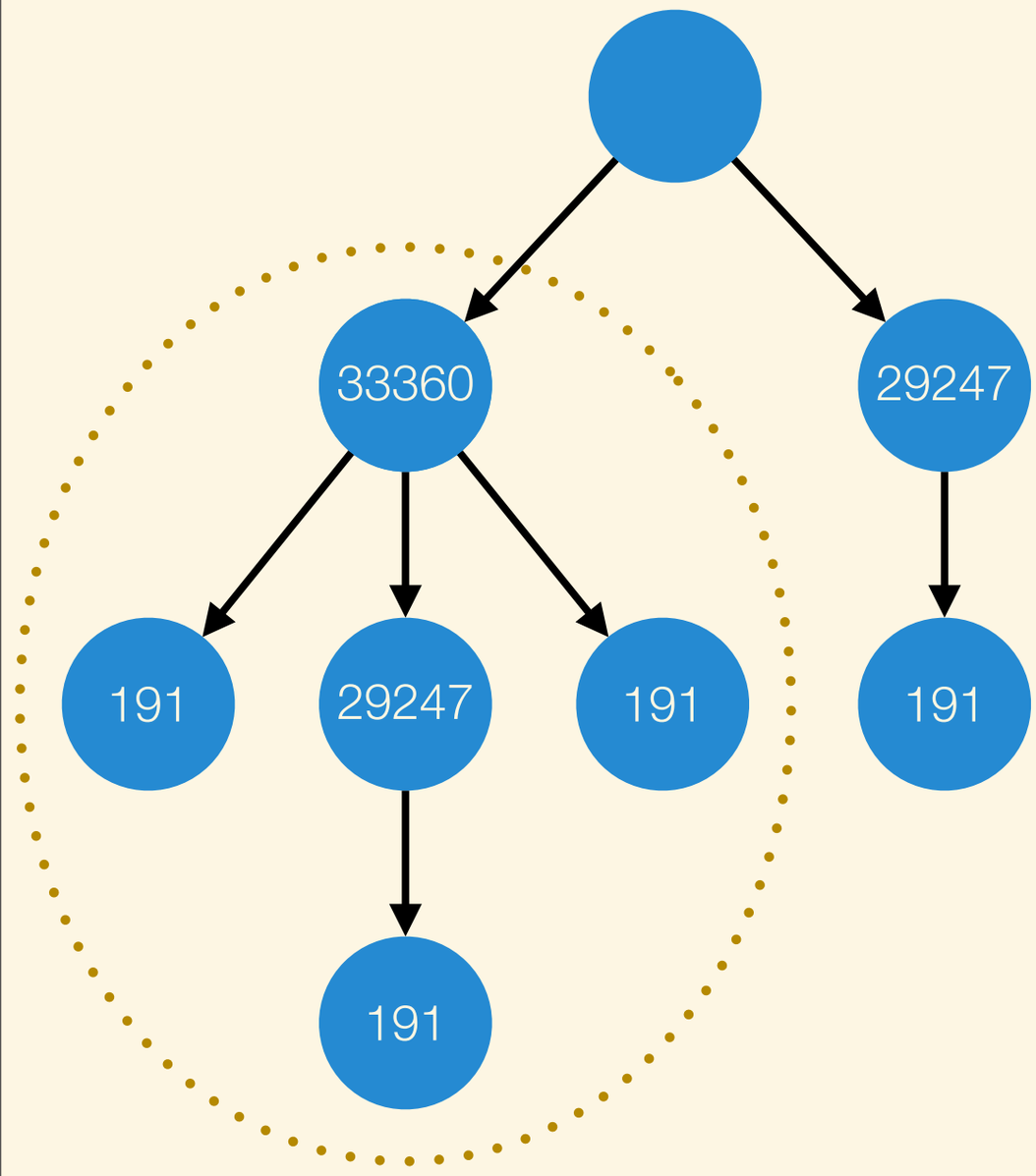


three-level sub-tree

child = (191,29247,191)

↓ sort

child = (191,191,29247)



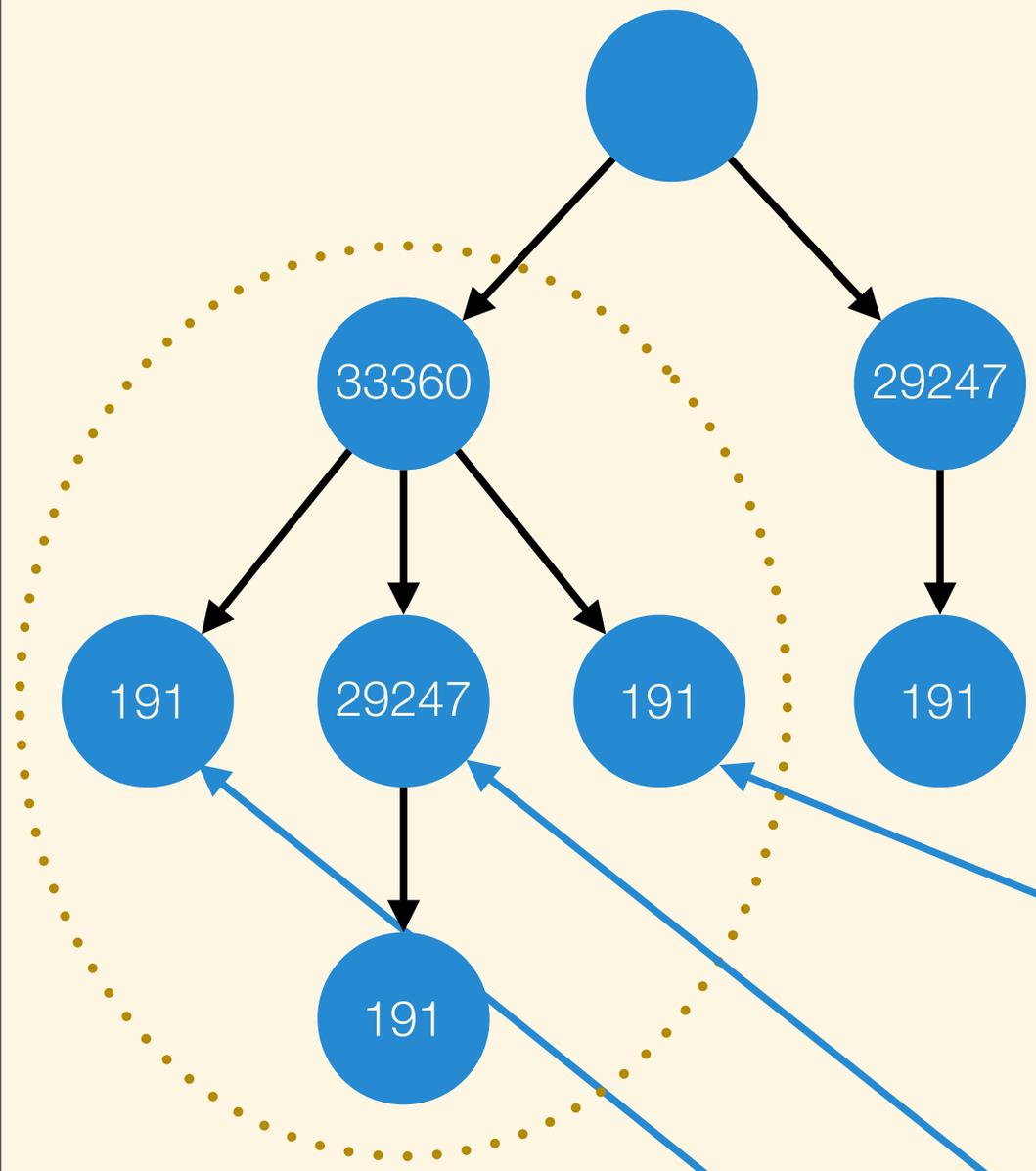
three-level sub-tree

child = (191,29247,191)

↓ sort

child = (191,191,29247)

(((((191 × 701 xor 191) mod 34943) × 701 xor 191) mod 34943) × 701 xor 29247) mod 34943 = 33360



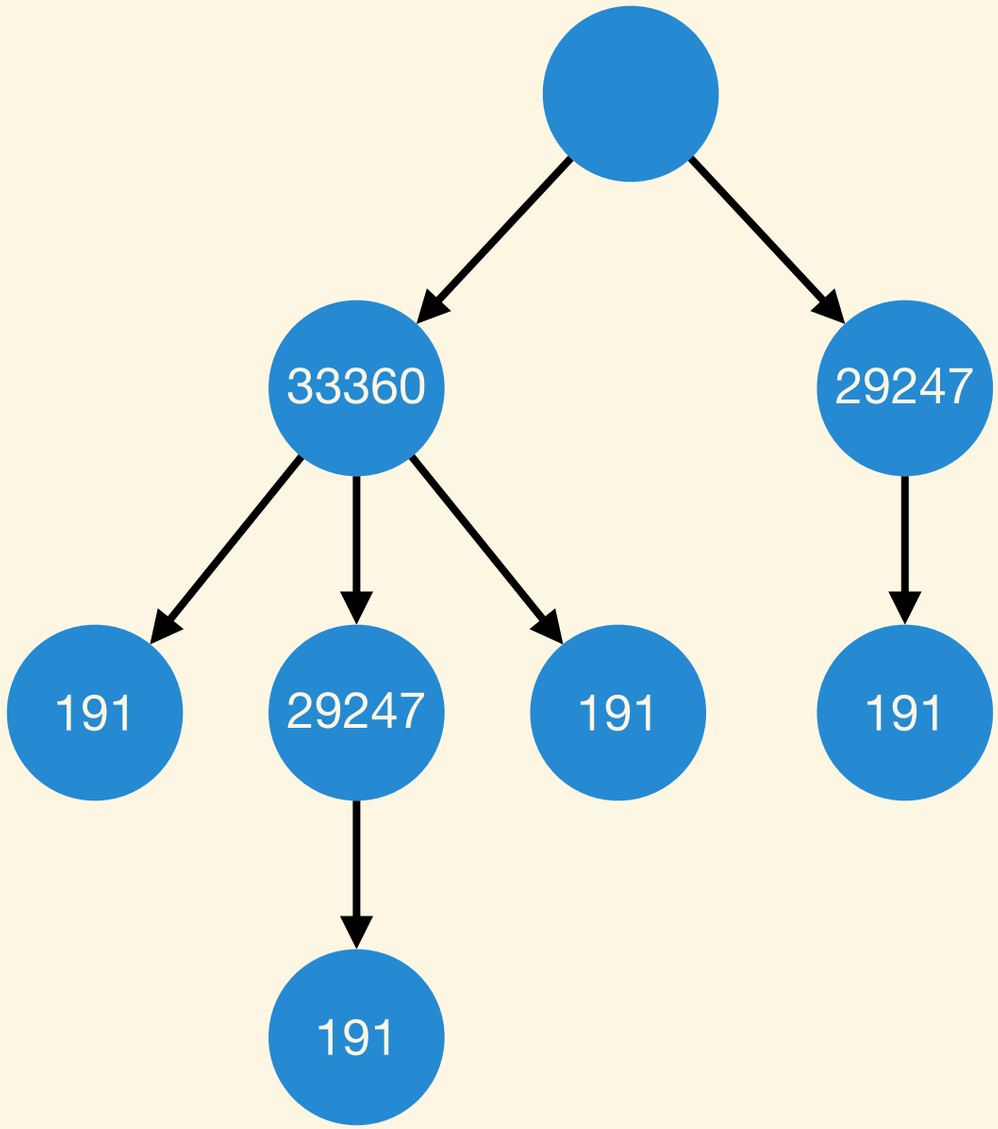
three-level sub-tree

child = (191,29247,191)

↓ sort

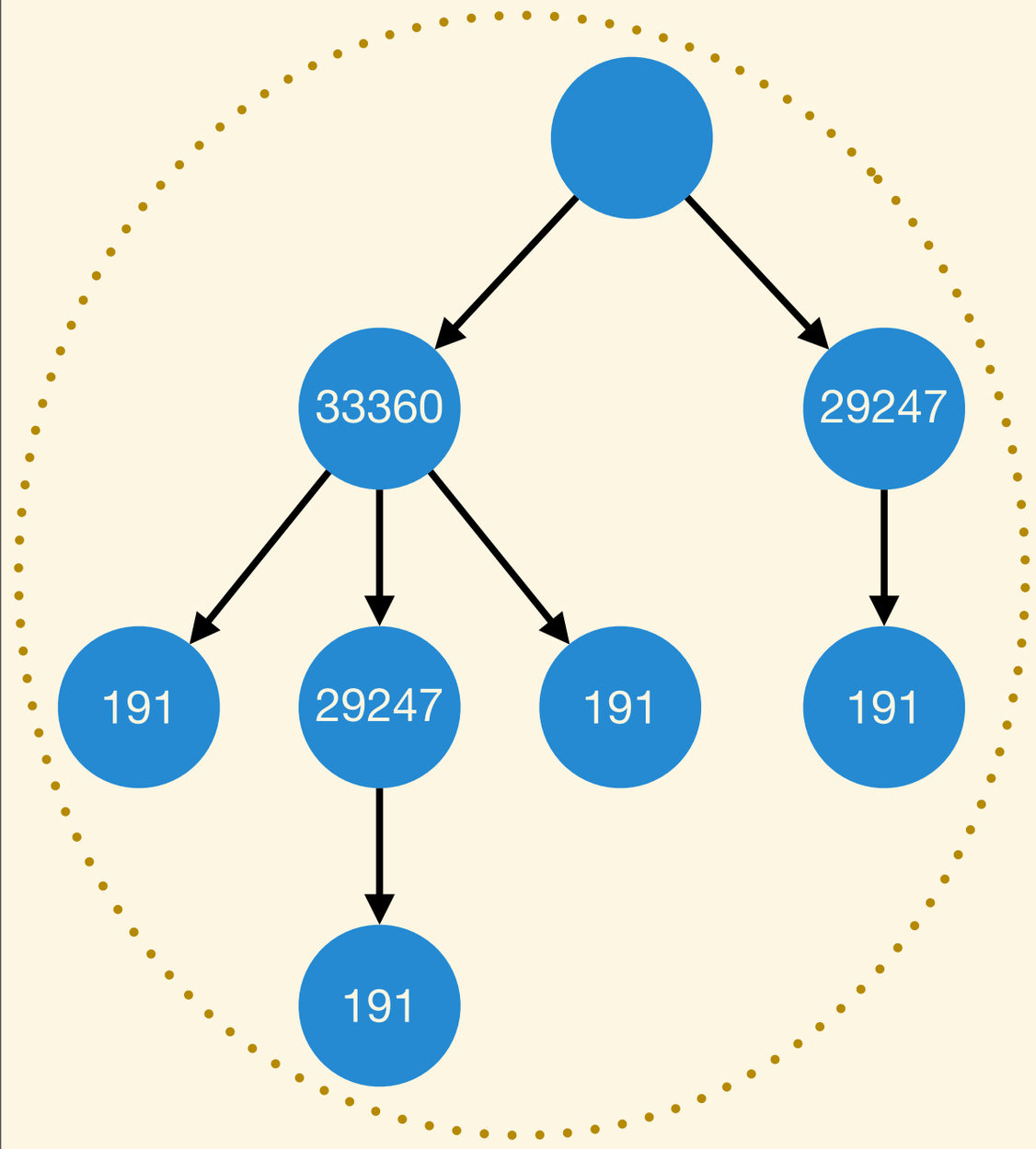
child = (191,191,29247)

((((191 × 701 xor 191) mod 34943) × 701 xor 191) mod 34943) × 701 xor 29247) mod 34943 = 33360



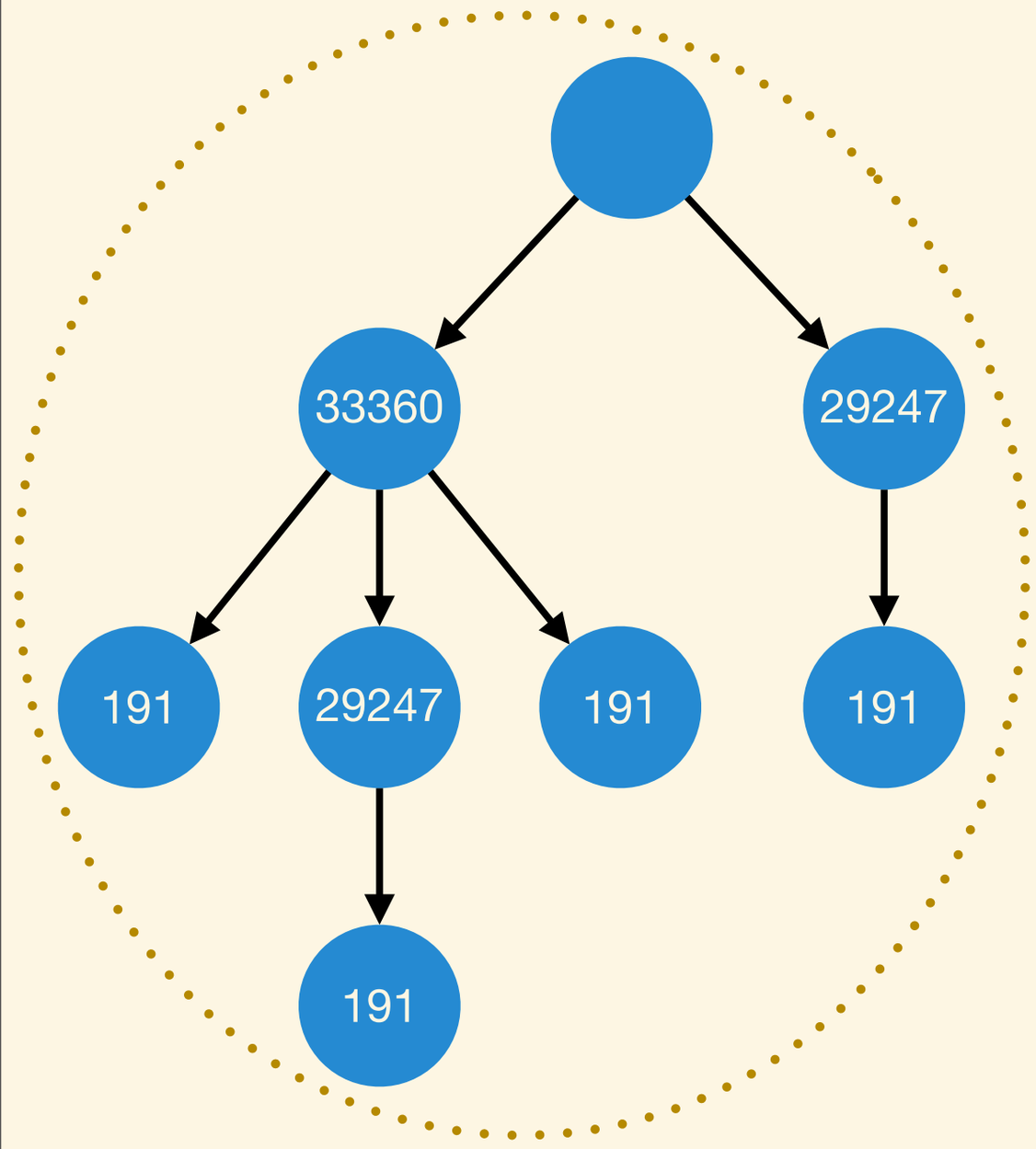
the total tree

child = (33360,29247)



the total tree

child = (33360,29247)

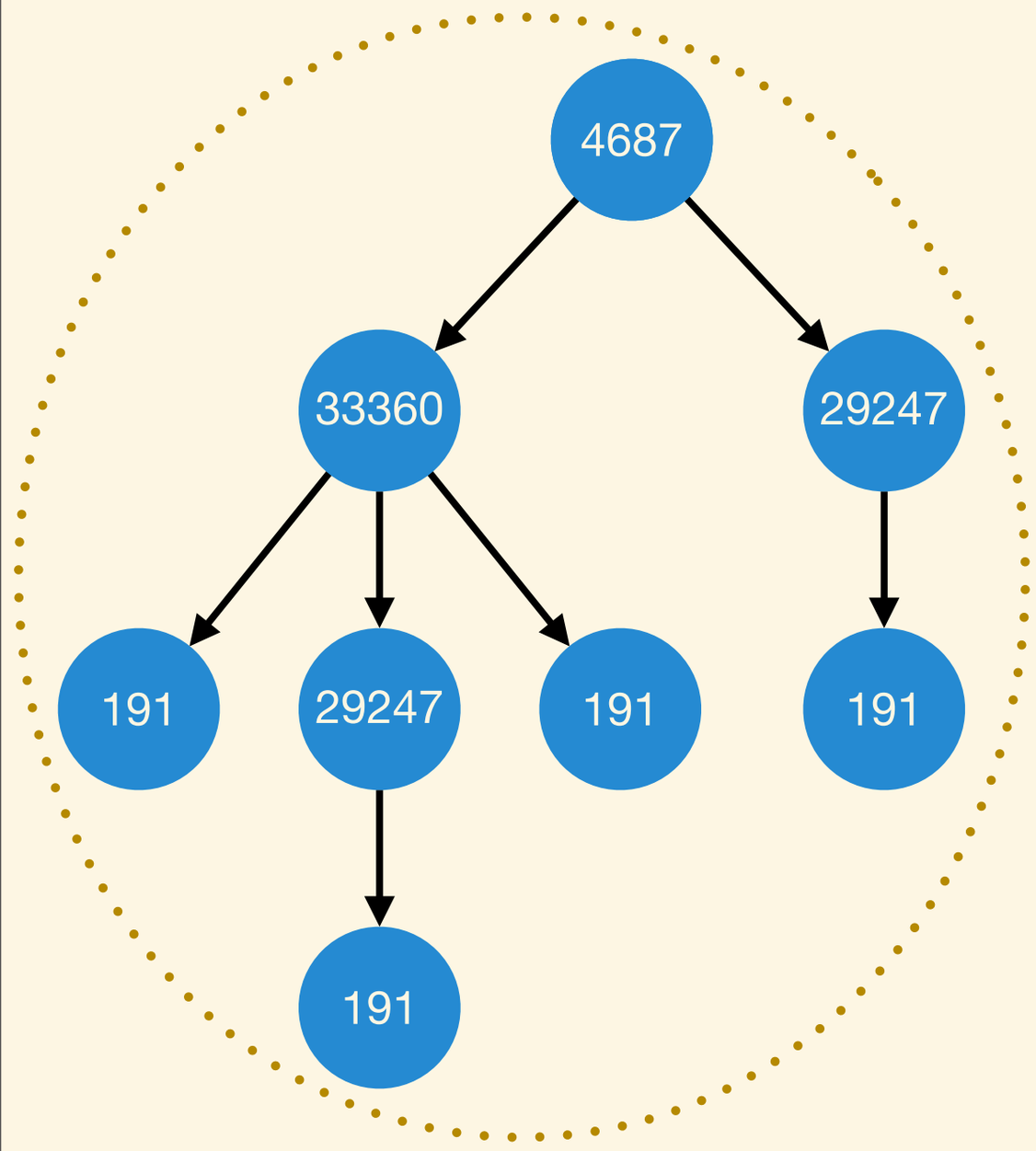


the total tree

child = (33360,29247)

↓ sort

child = (29247,33360)



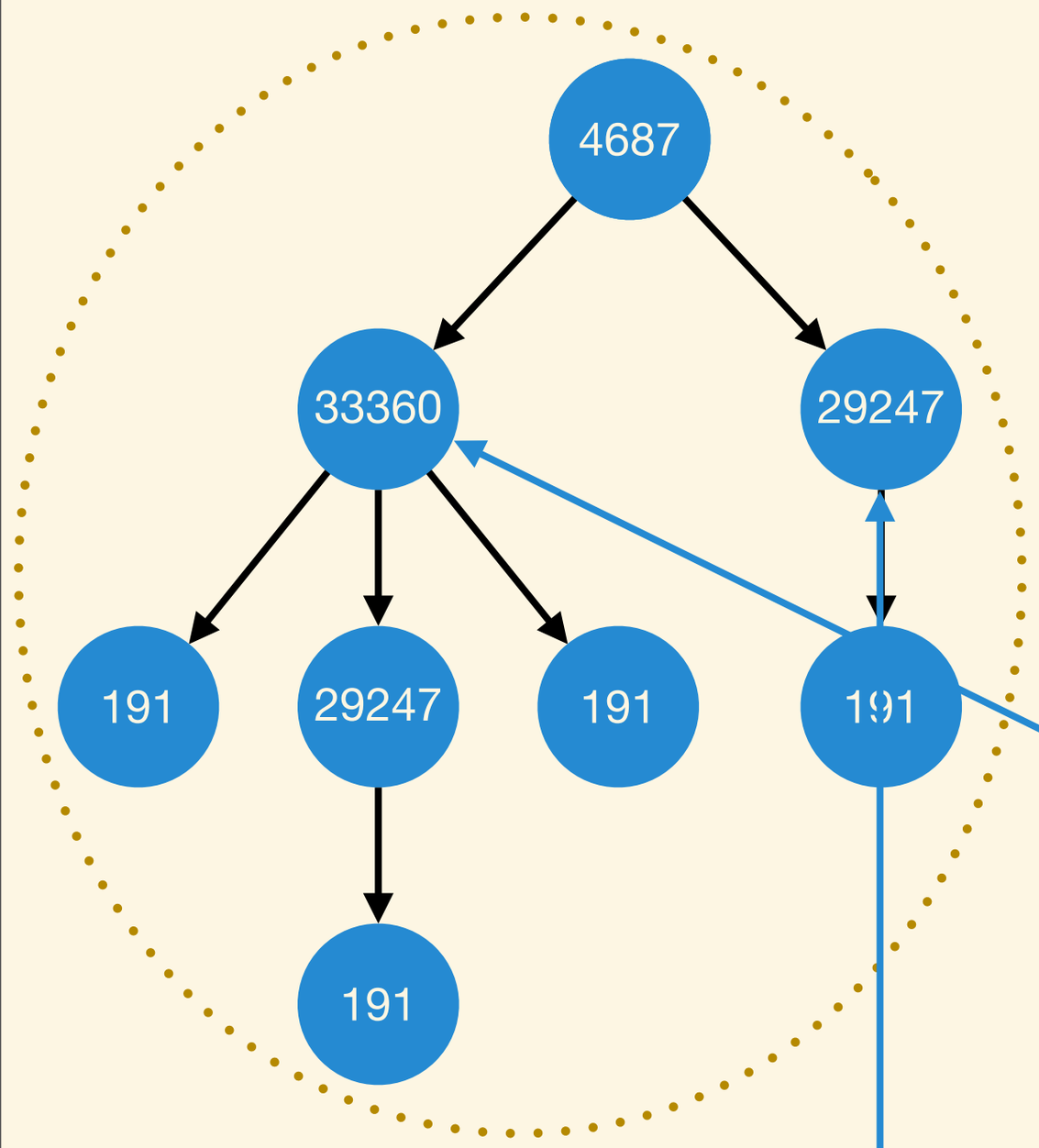
the total tree

child = (33360,29247)

↓ sort

child = (29247,33360)

(((191 × 701 xor 29247) mod 34943) × 701 xor 33360)
mod 34943 = 4687



the total tree

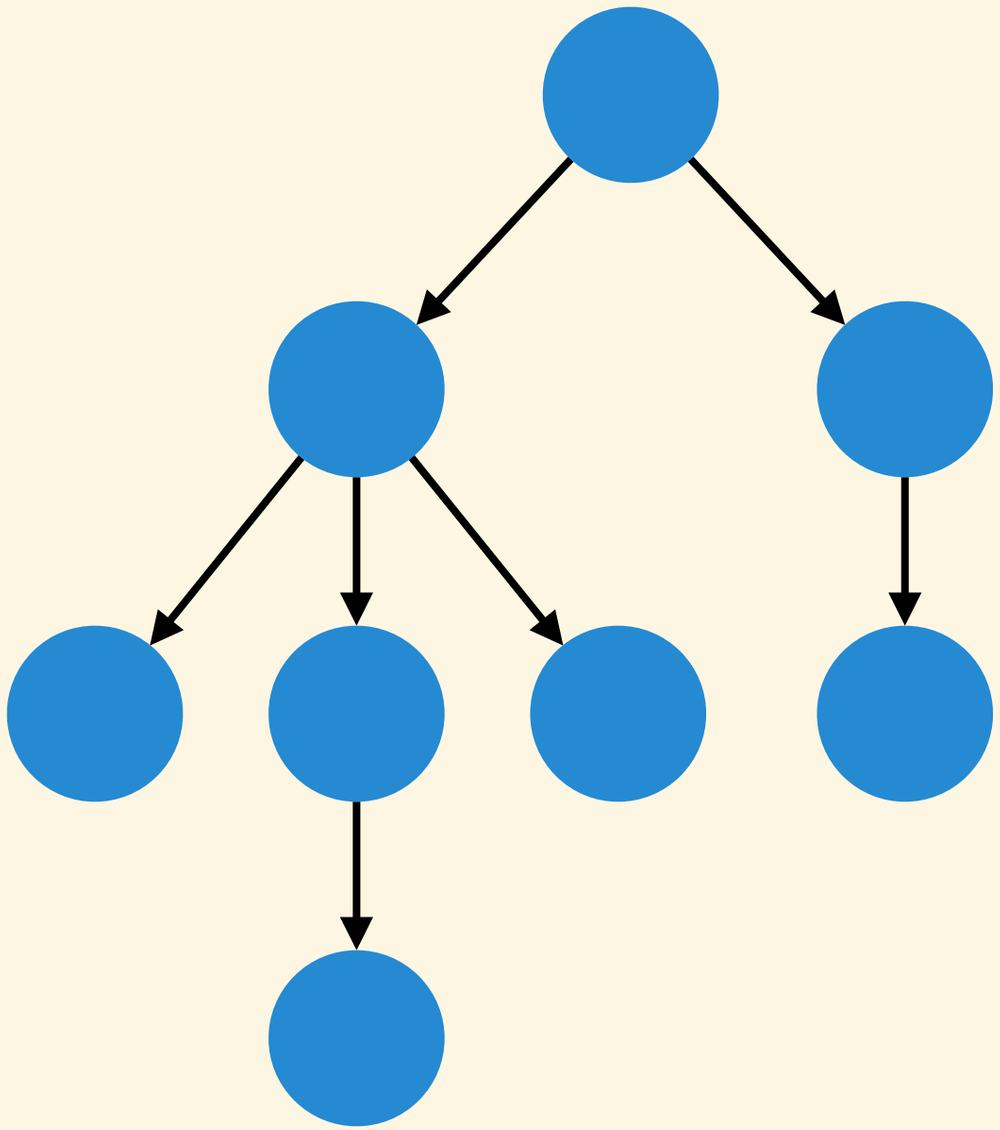
child = (33360,29247)

↓ sort

child = (29247,33360)

(((191 × 701 xor 29247) mod 34943) × 701 xor 33360)

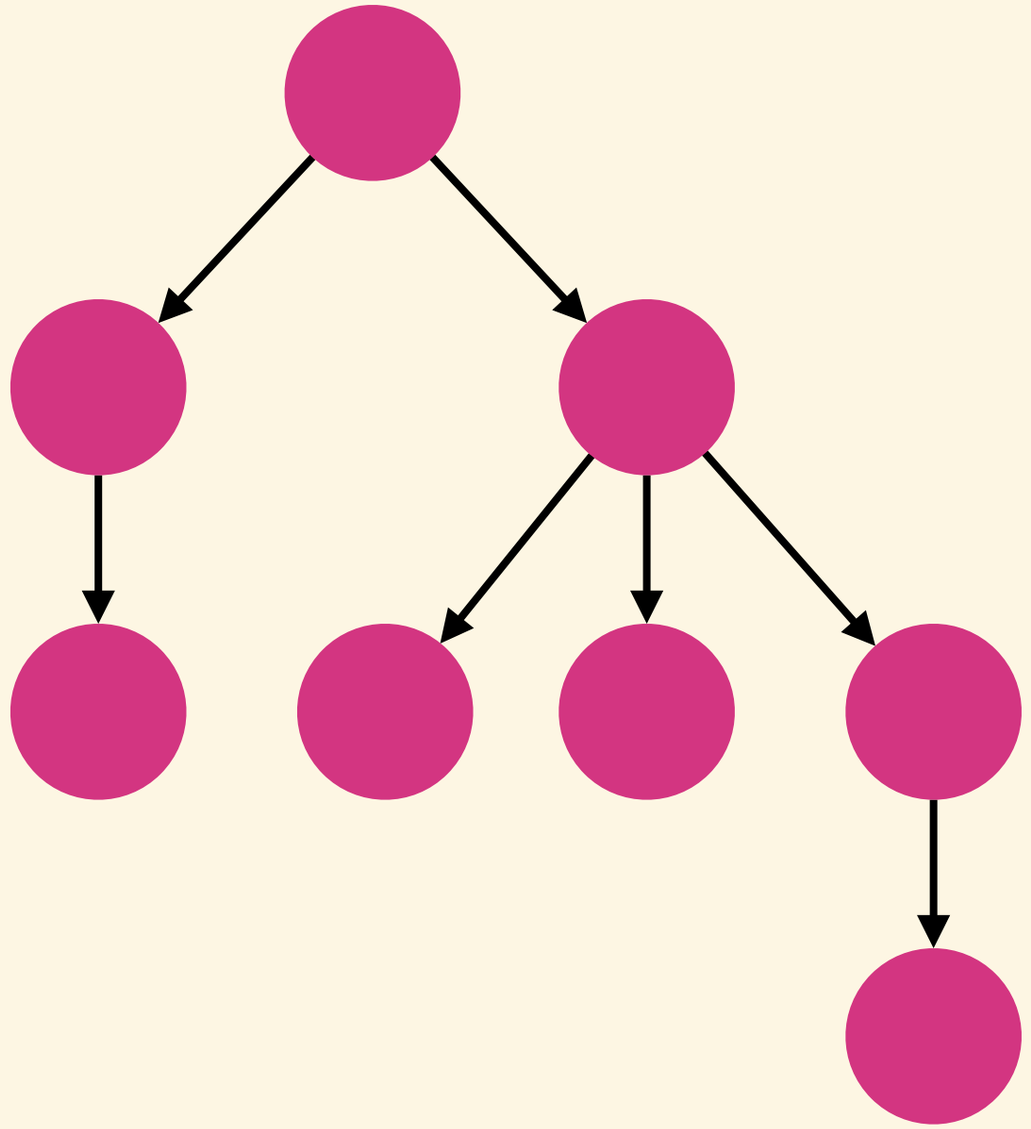
mod 34943 = 4687



hash value of the tree is 4687

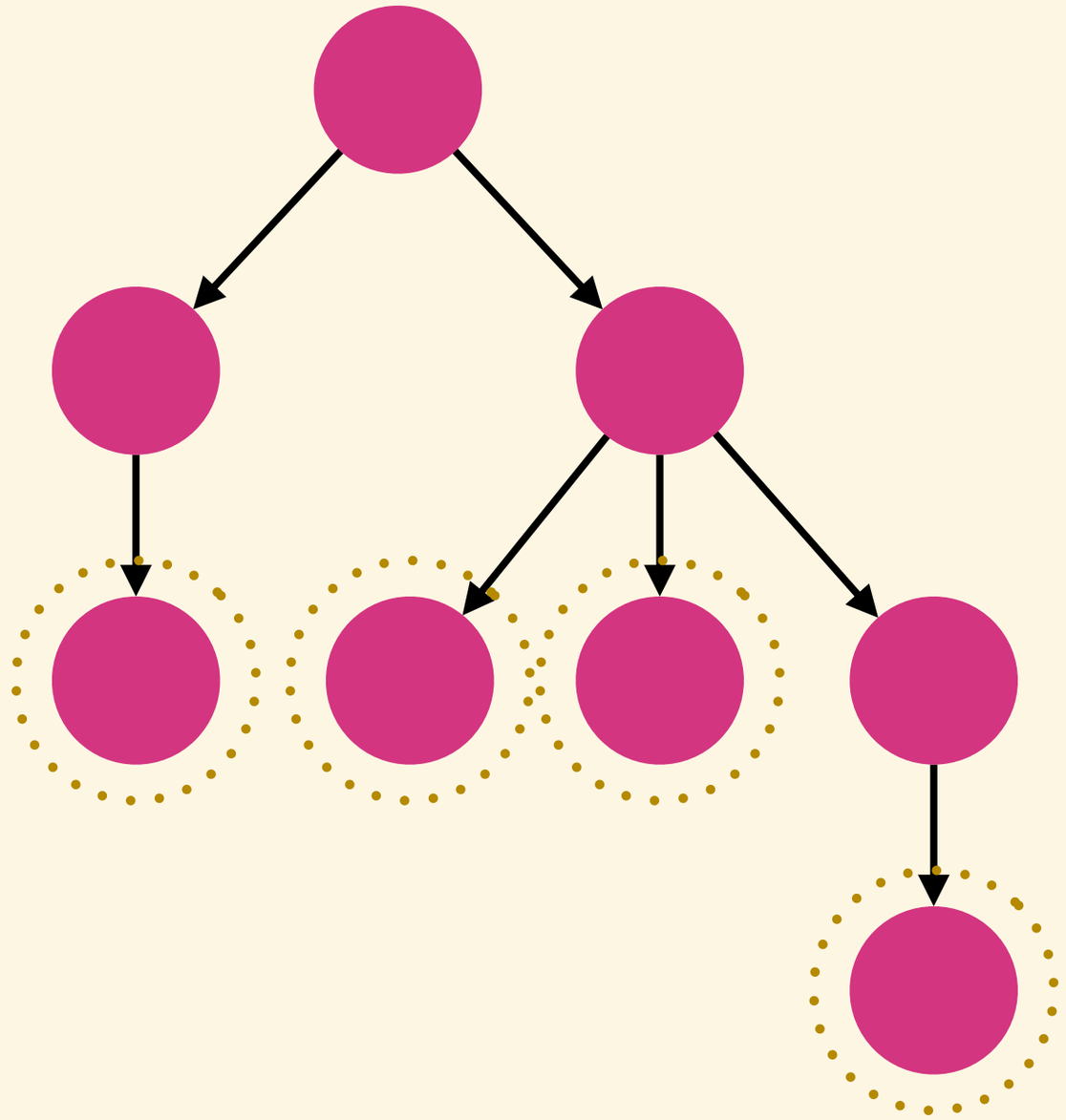
single vertex

initial value = 191



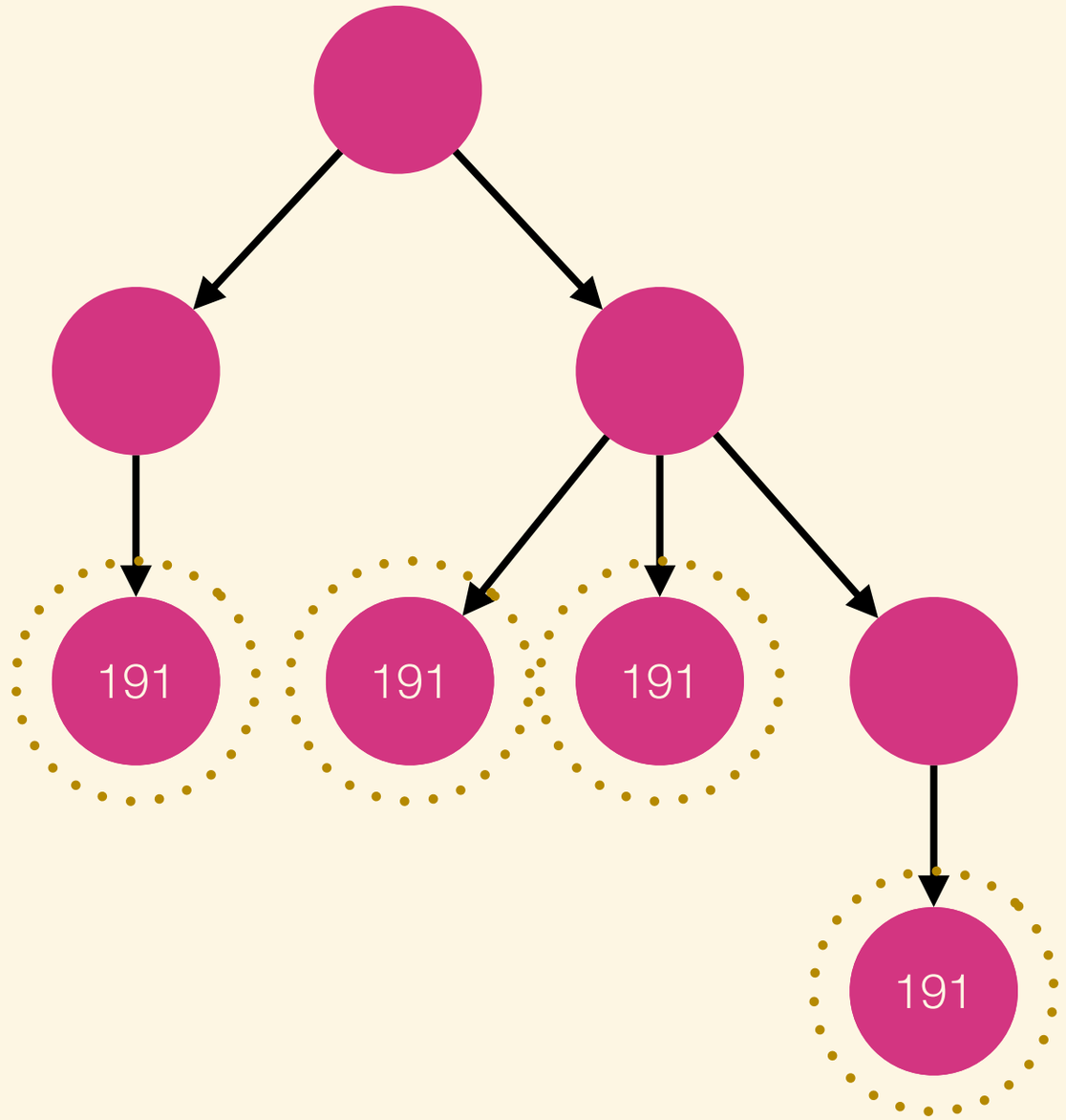
single vertex

initial value = 191



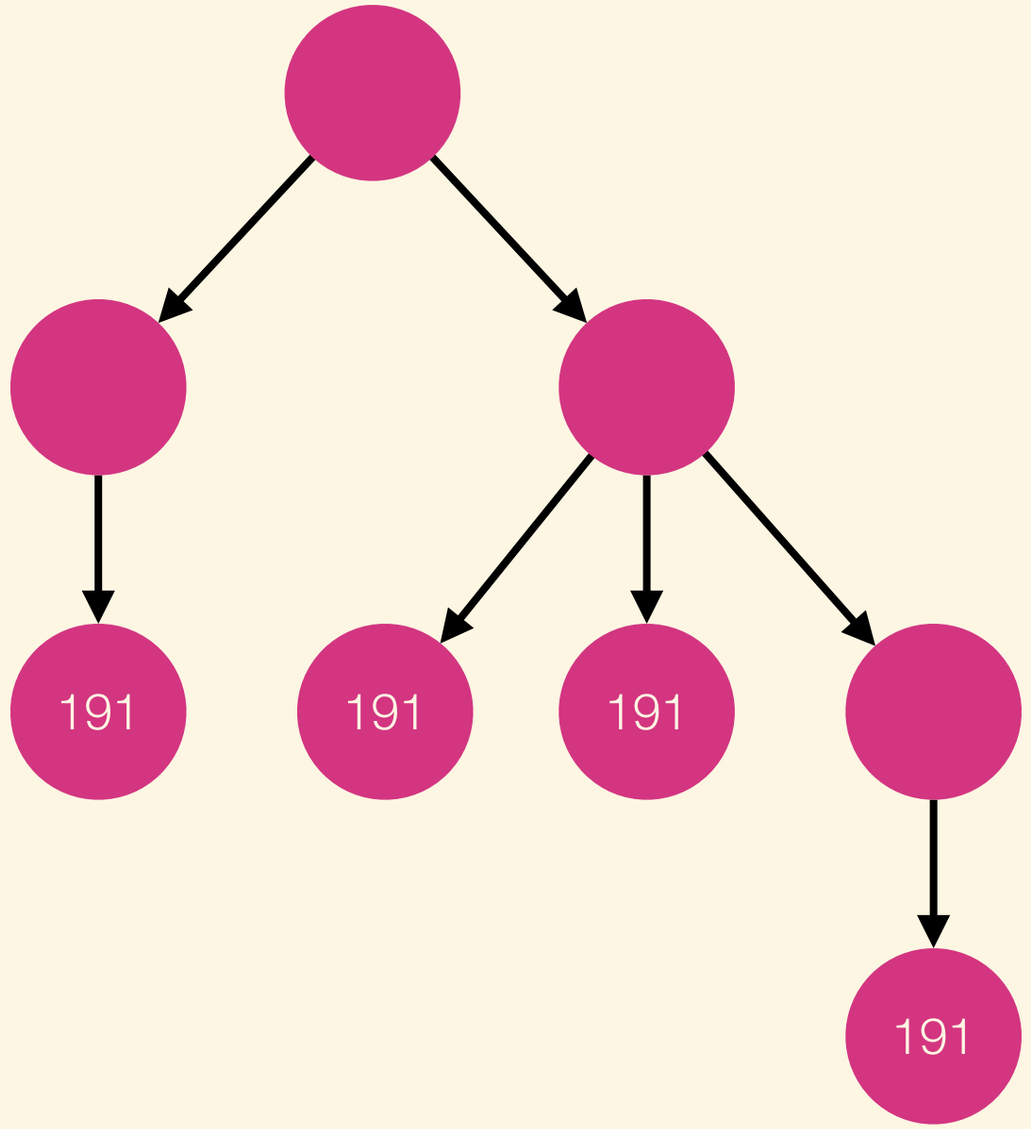
single vertex

initial value = 191



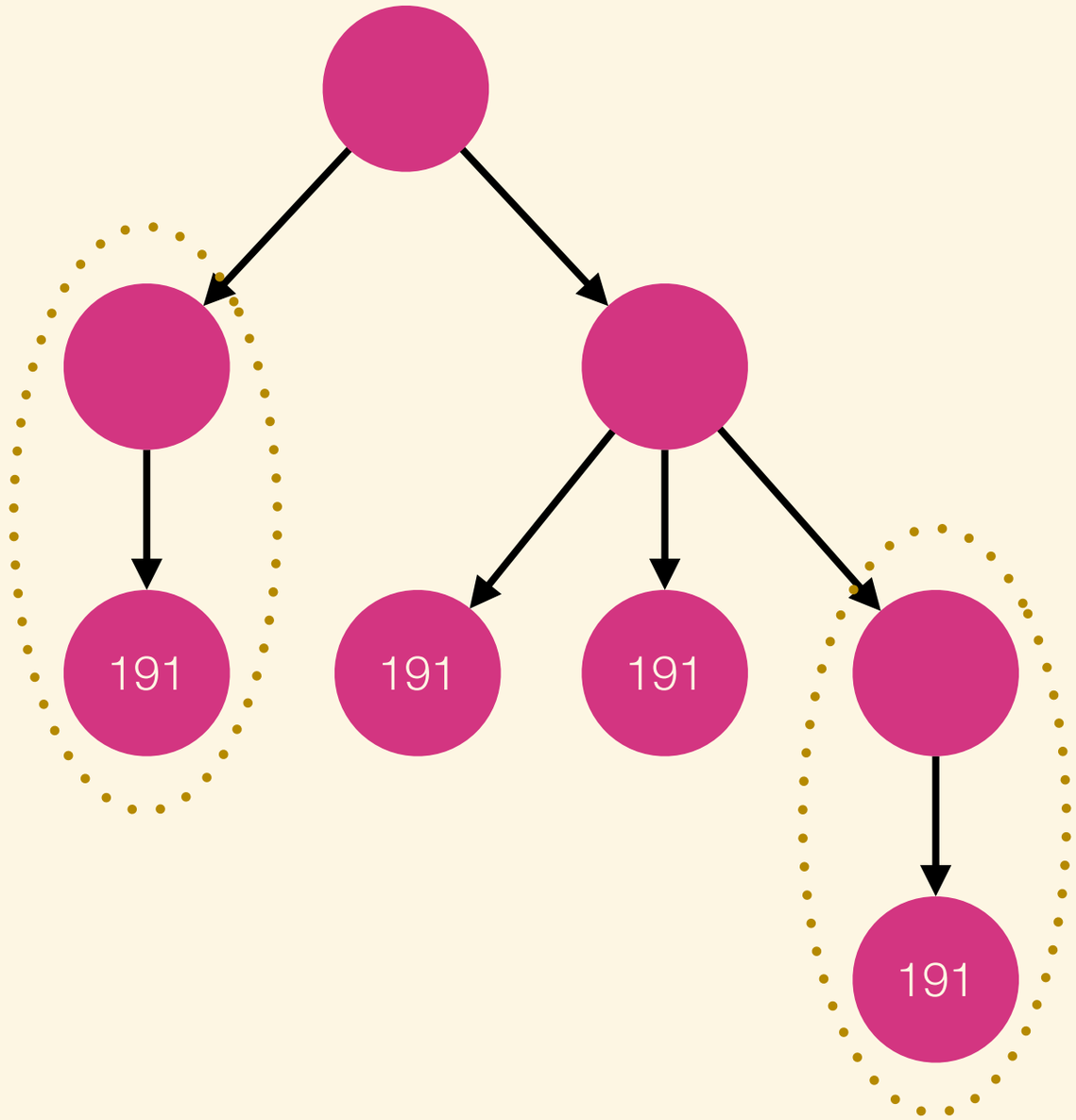
two-level sub-tree

child = (191)



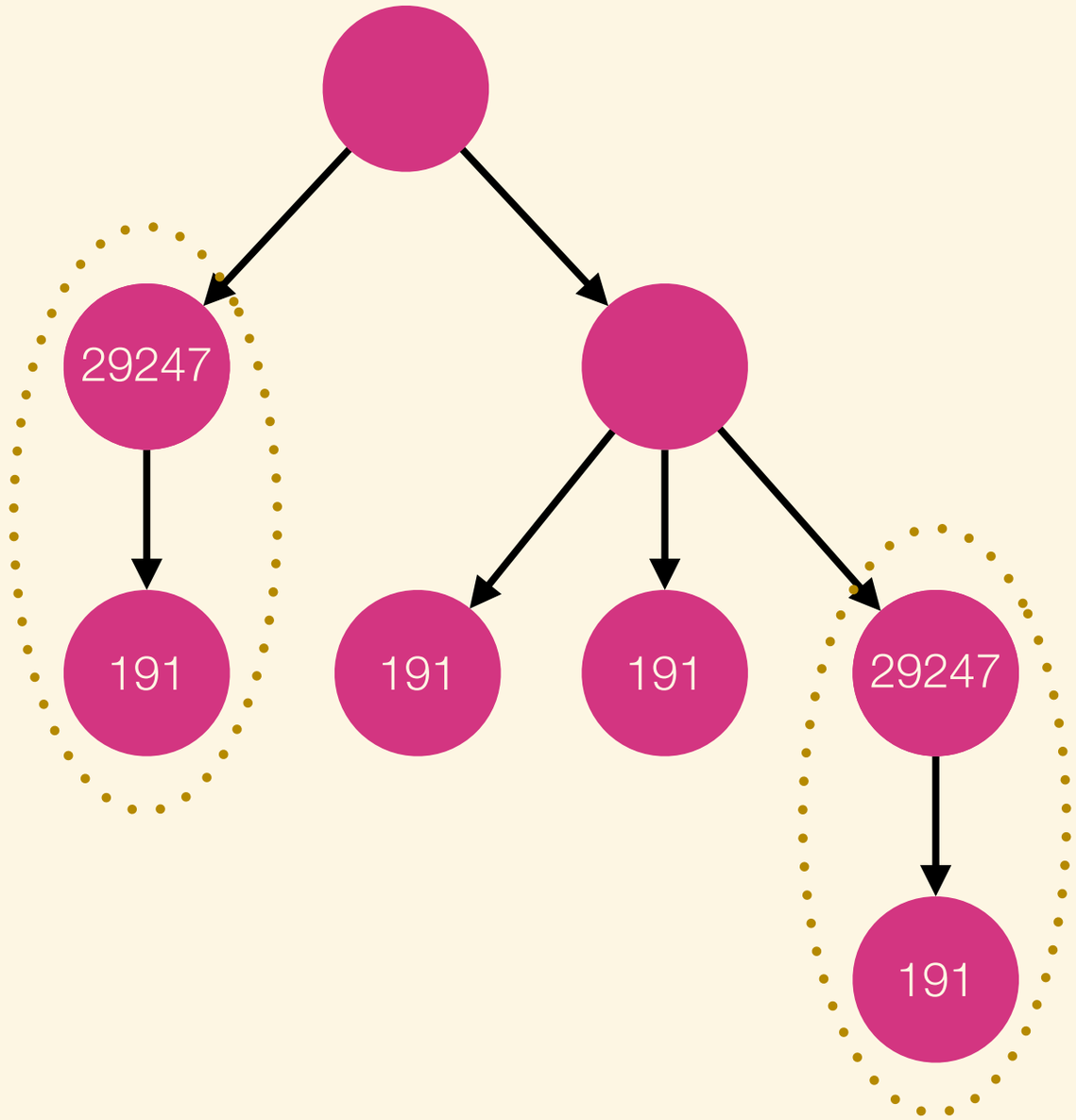
two-level sub-tree

child = (191)



two-level sub-tree

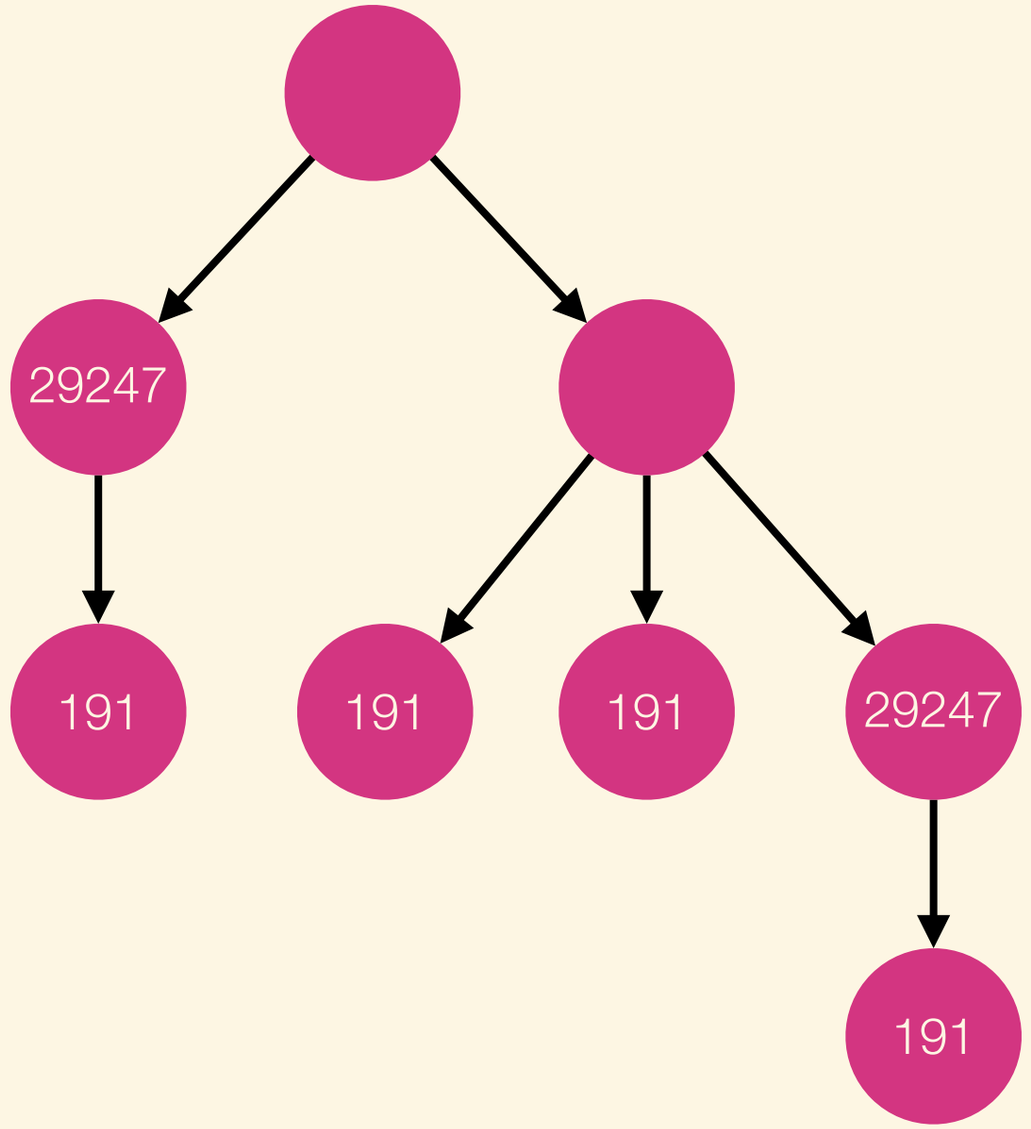
child = (191)



(191 × 701 xor 191) mod 34943 = 29247

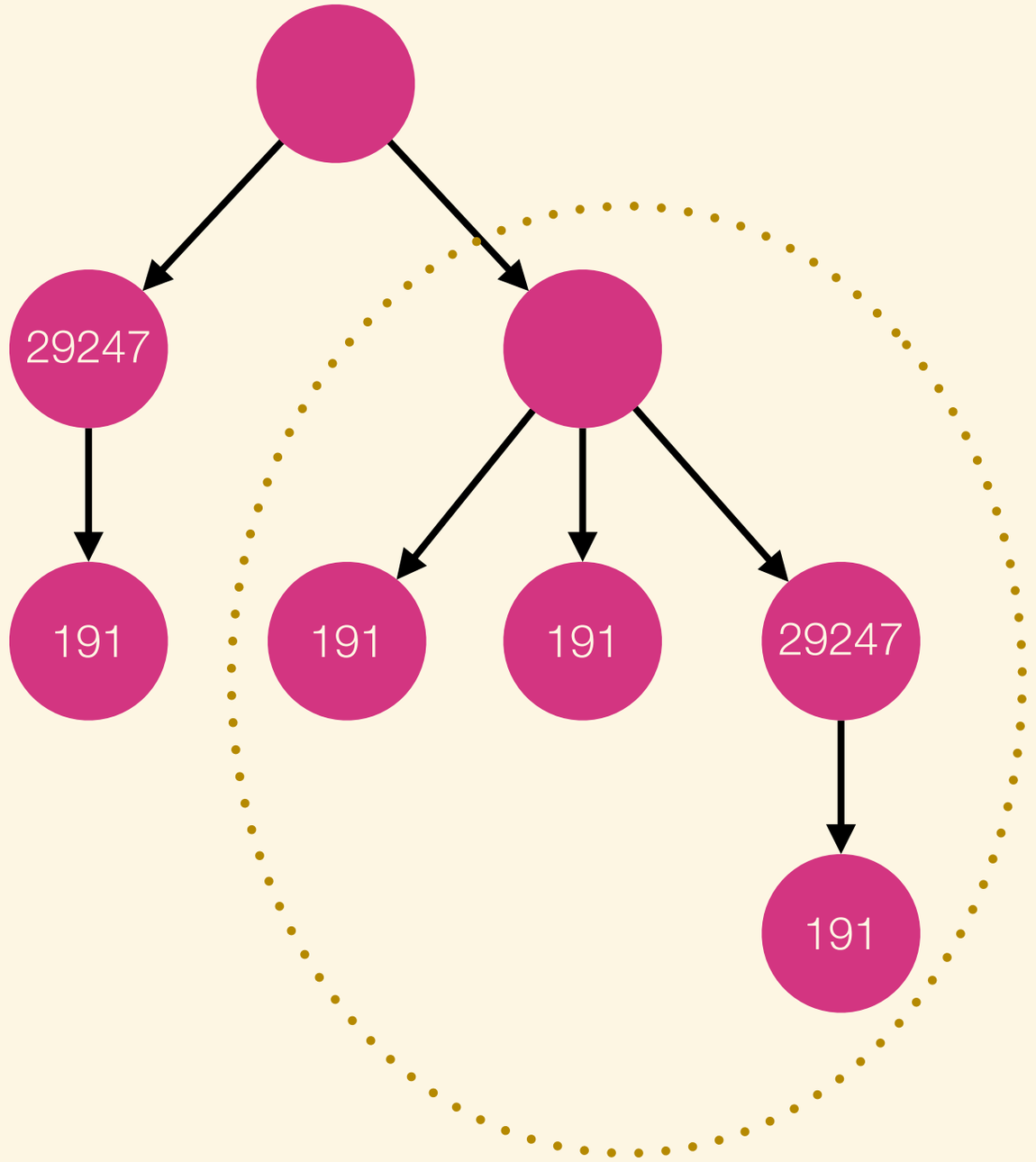
three-level sub-tree

child = (191,191,29247)



three-level sub-tree

child = (191,191,29247)

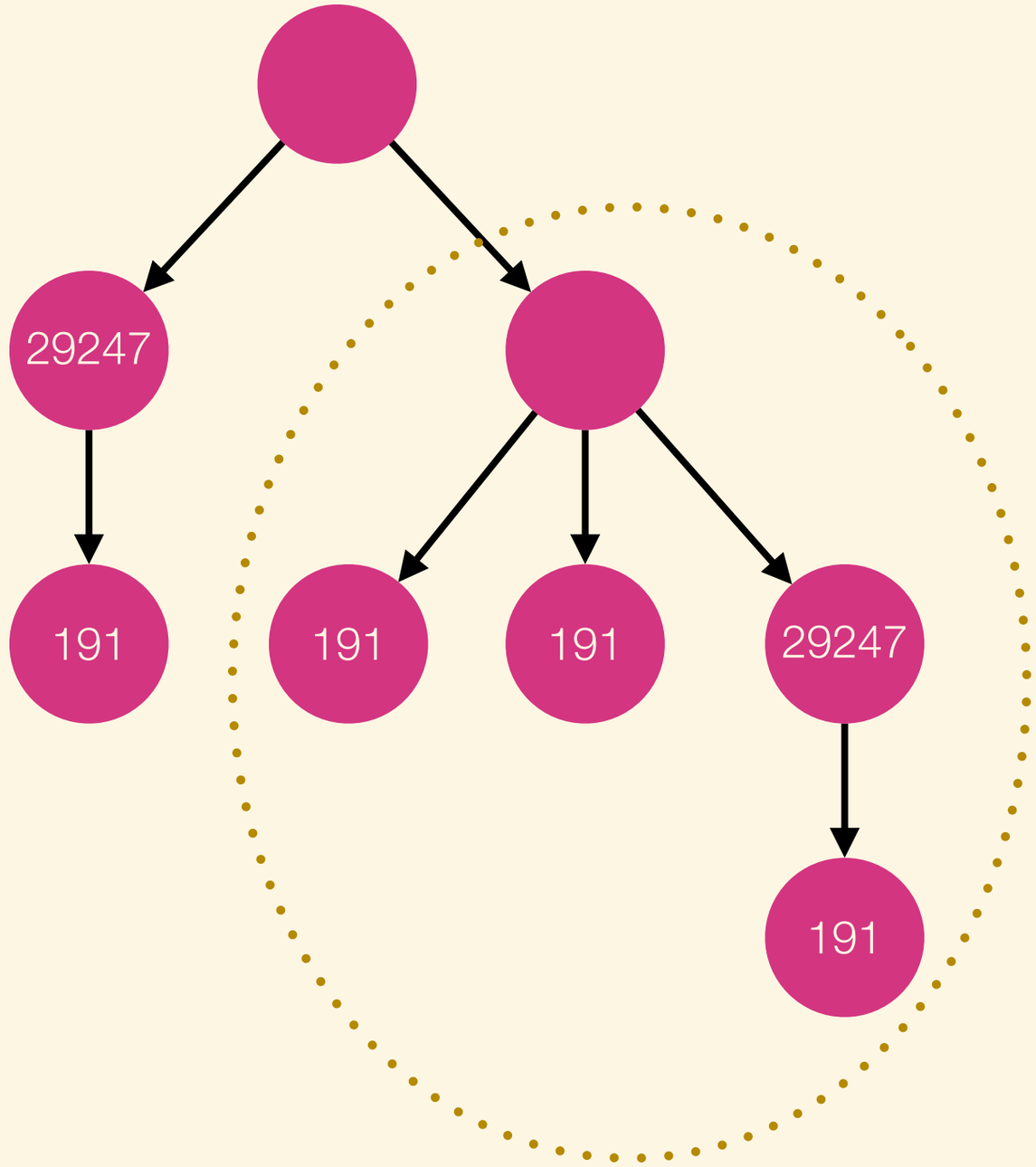


three-level sub-tree

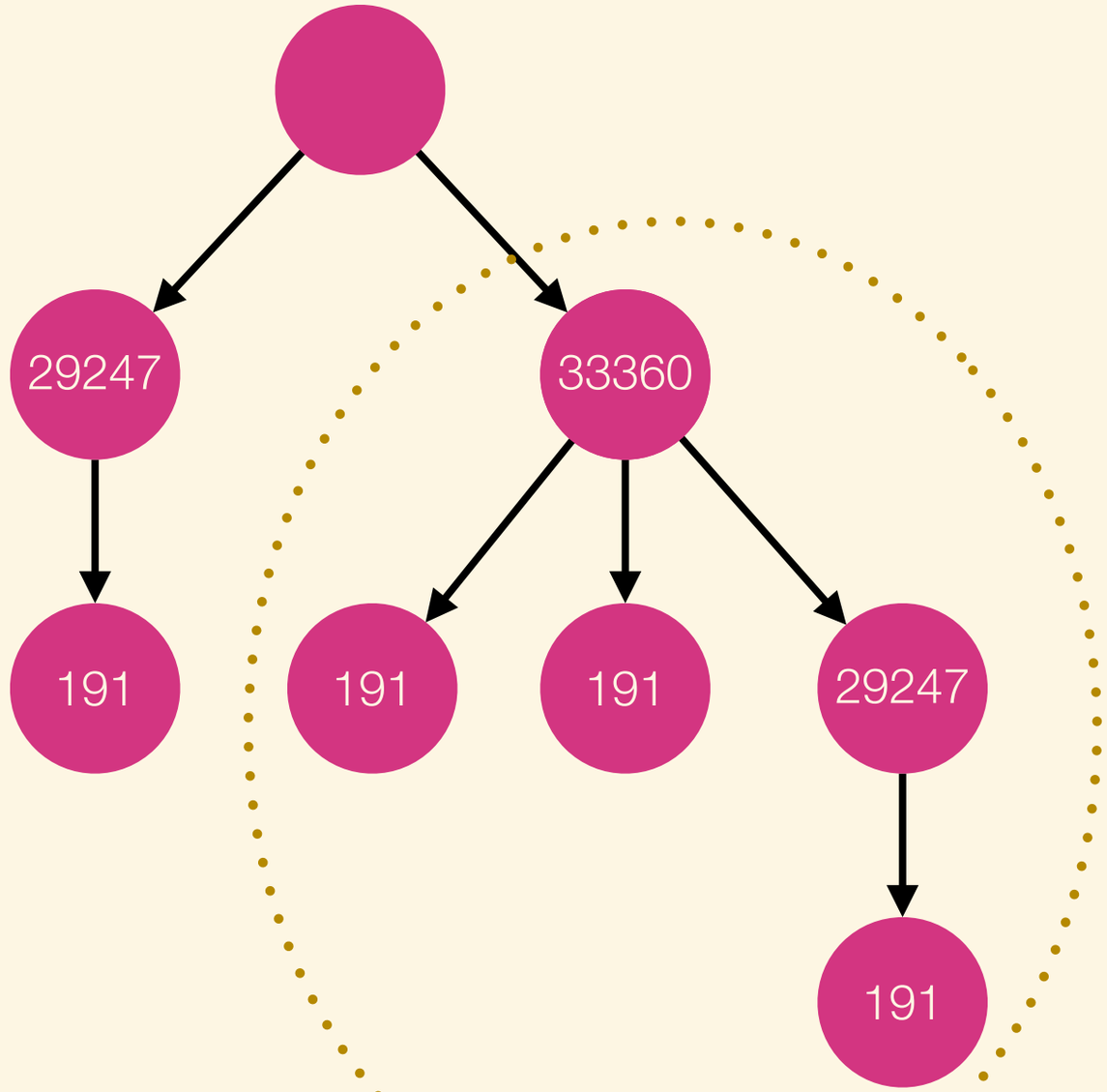
child = (191,191,29247)

↓ sort

child = (191,191,29247)



three-level sub-tree



child = (191,191,29247)

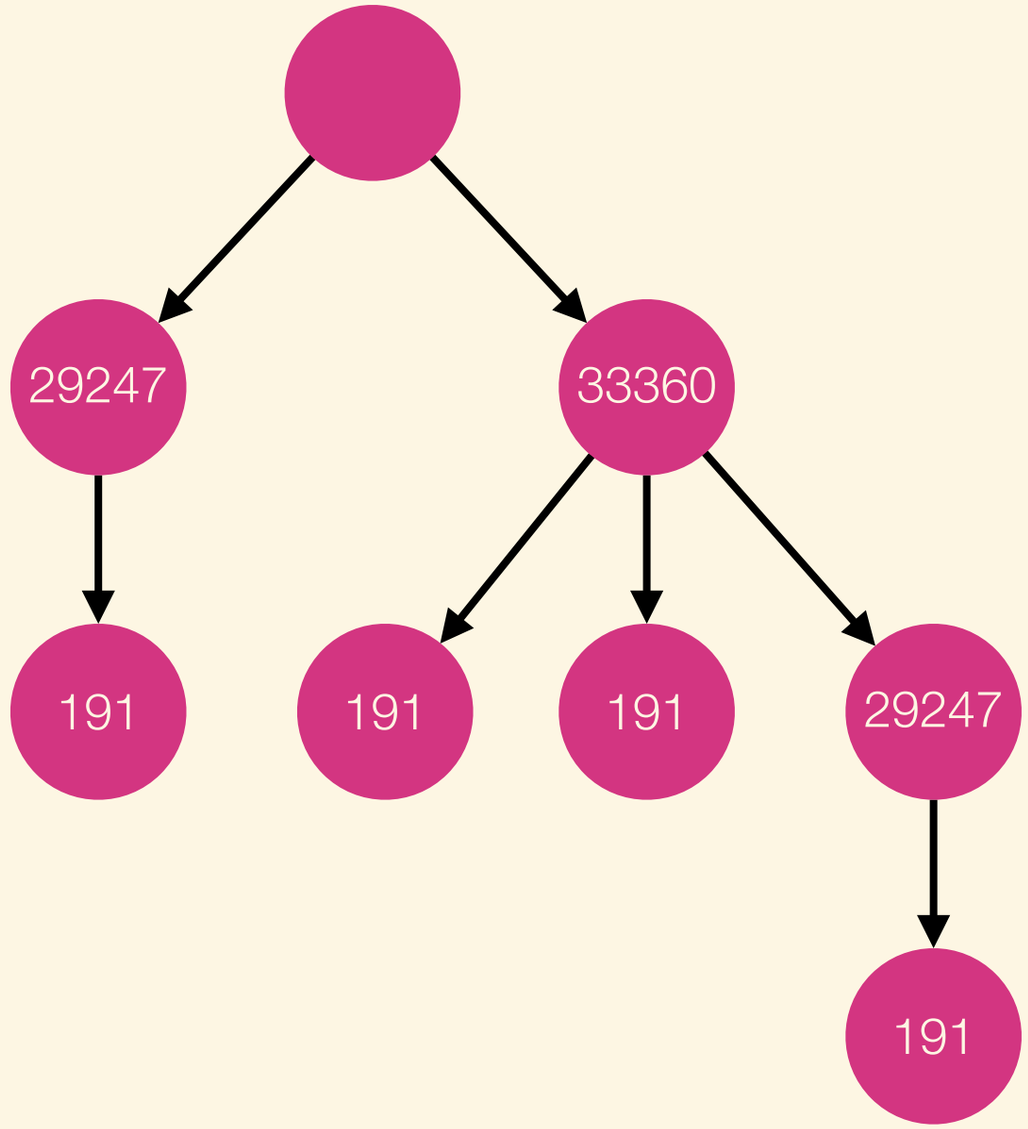
↓ sort

child = (191,191,29247)

(((((191 × 701 xor 191) mod 34943) × 701 xor 191) mod 34943) × 701 xor 29247) mod 34943 = 33360

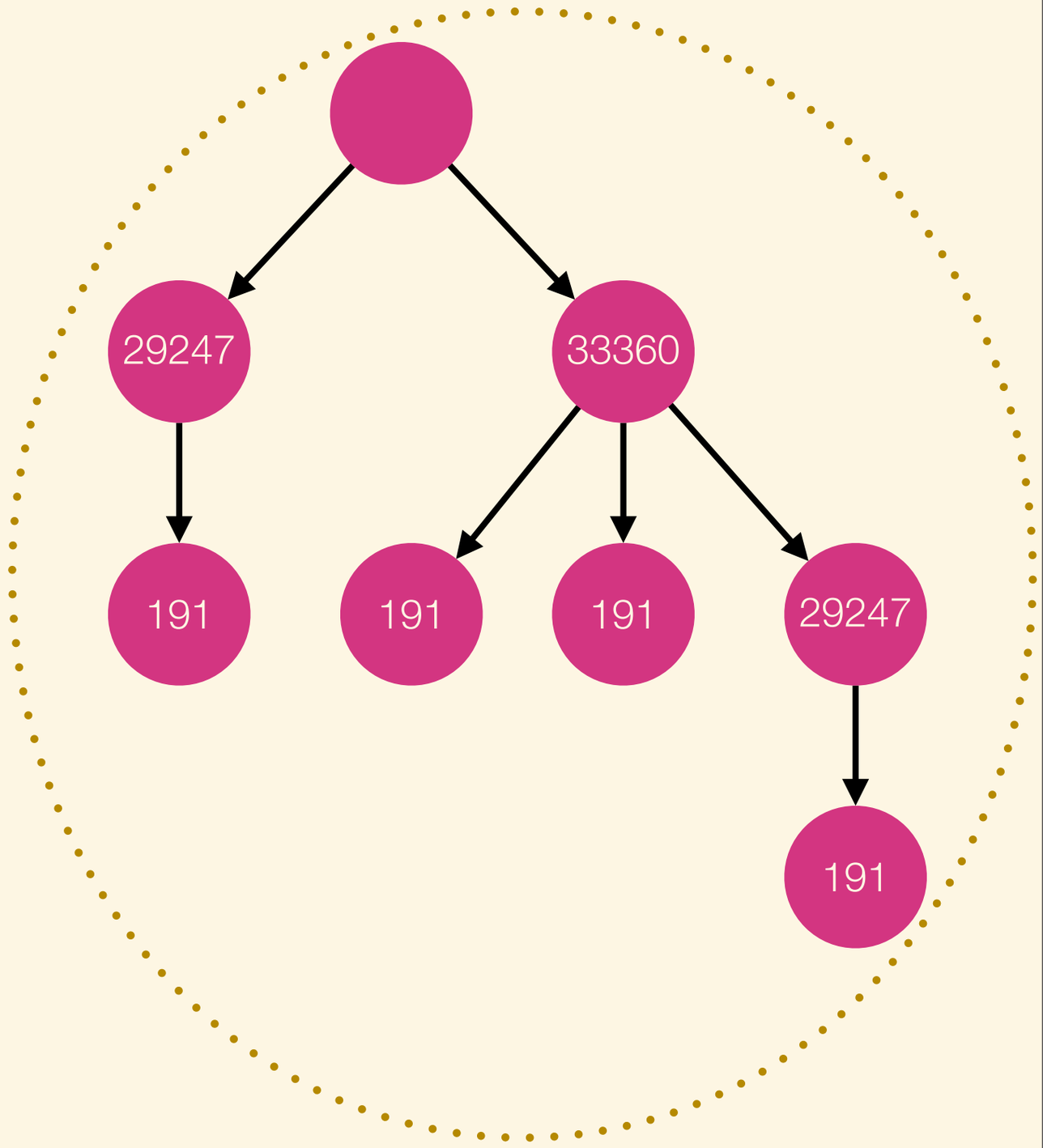
the total tree

child = (33360,29247)



the total tree

child = (33360,29247)

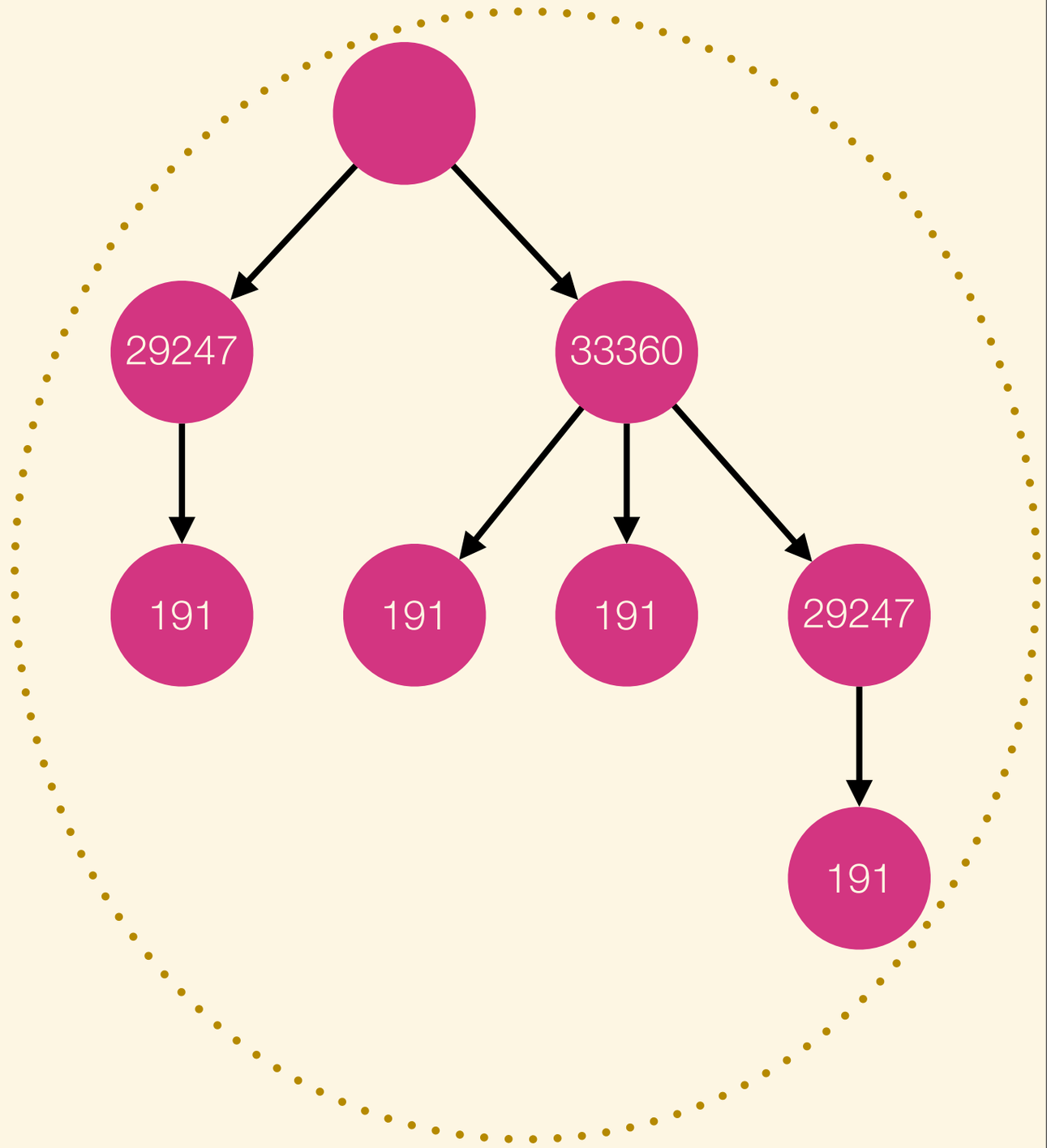


the total tree

child = (33360,29247)

↓ sort

child = (29247,33360)

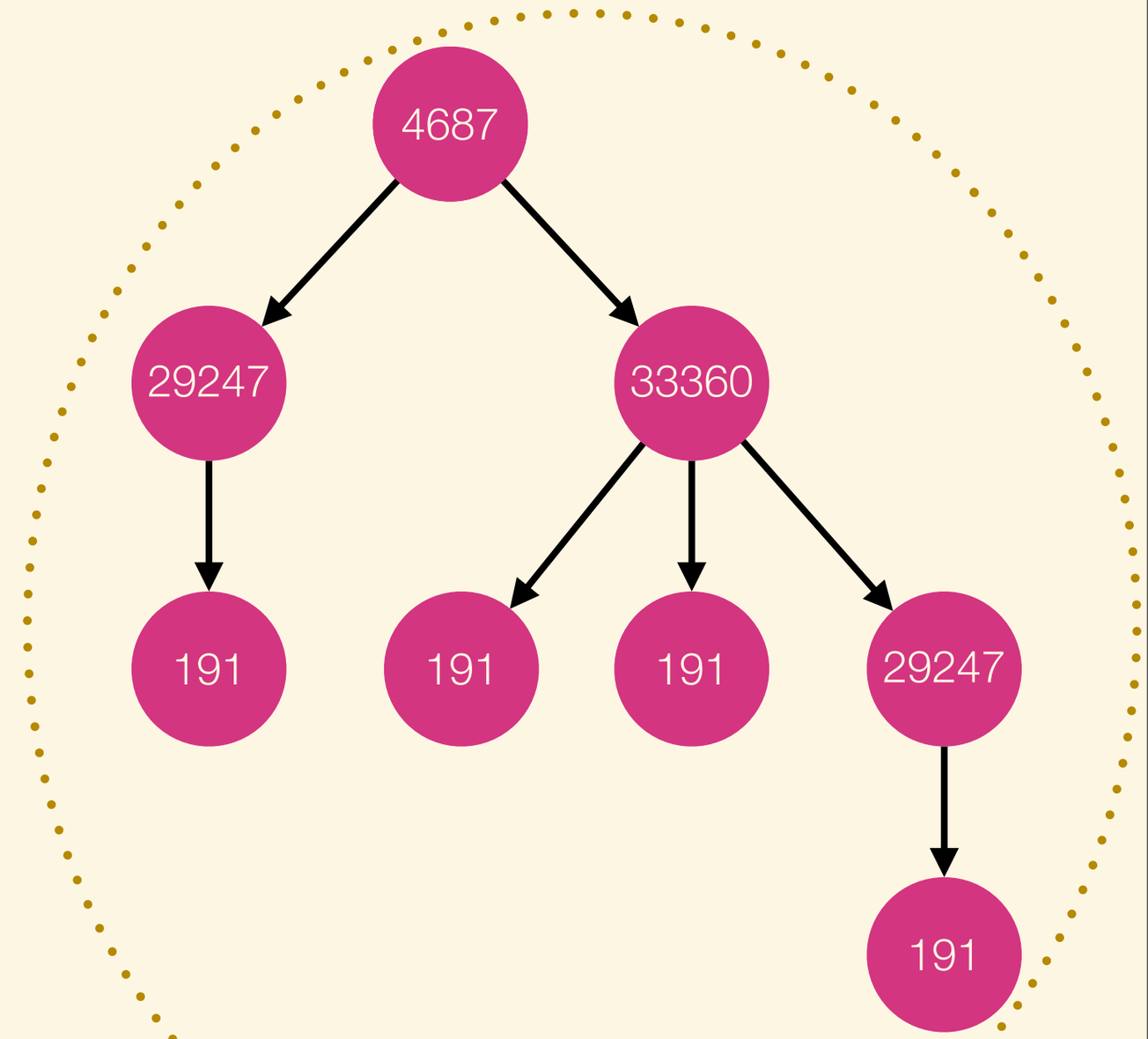


the total tree

child = (33360,29247)

↓ sort

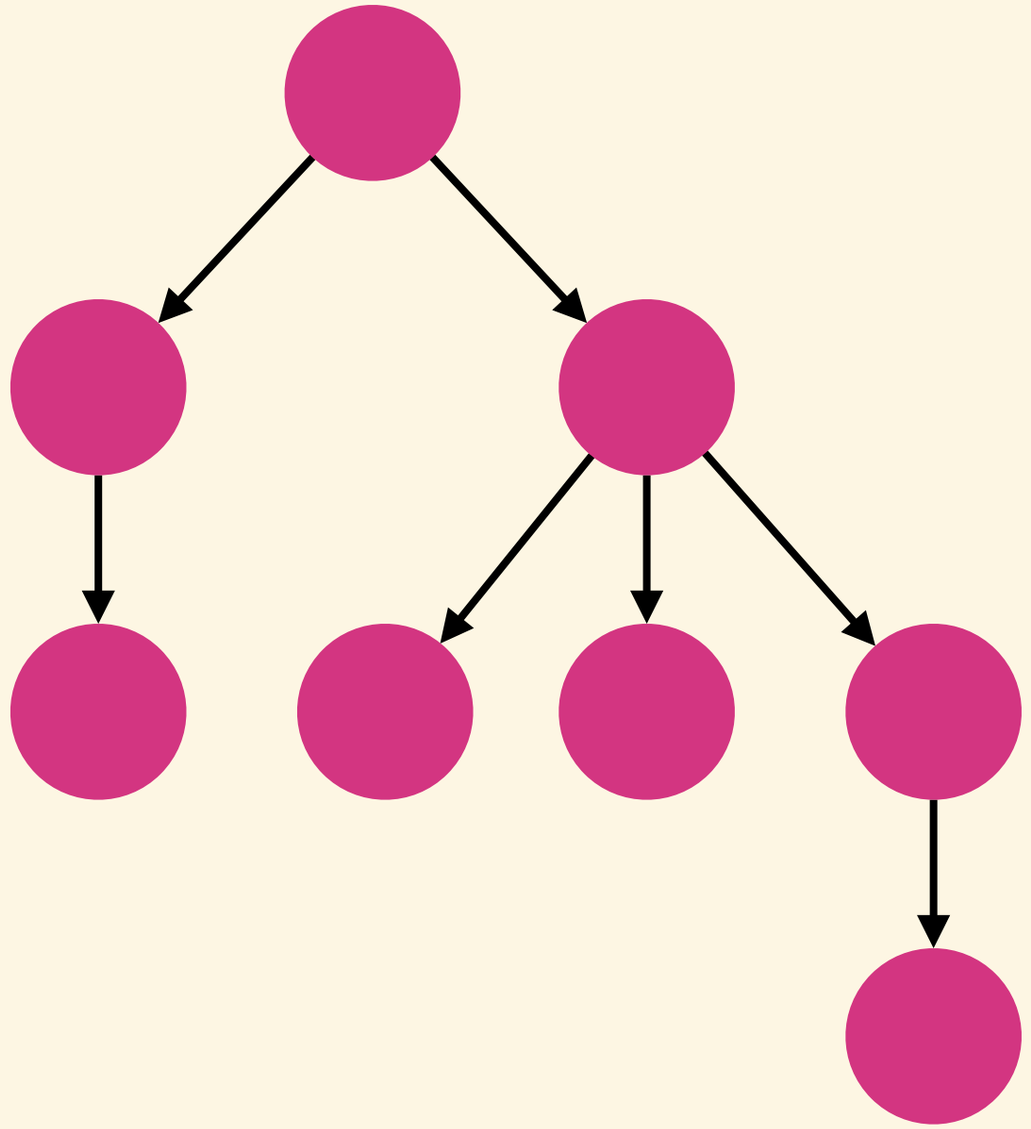
child = (29247,33360)

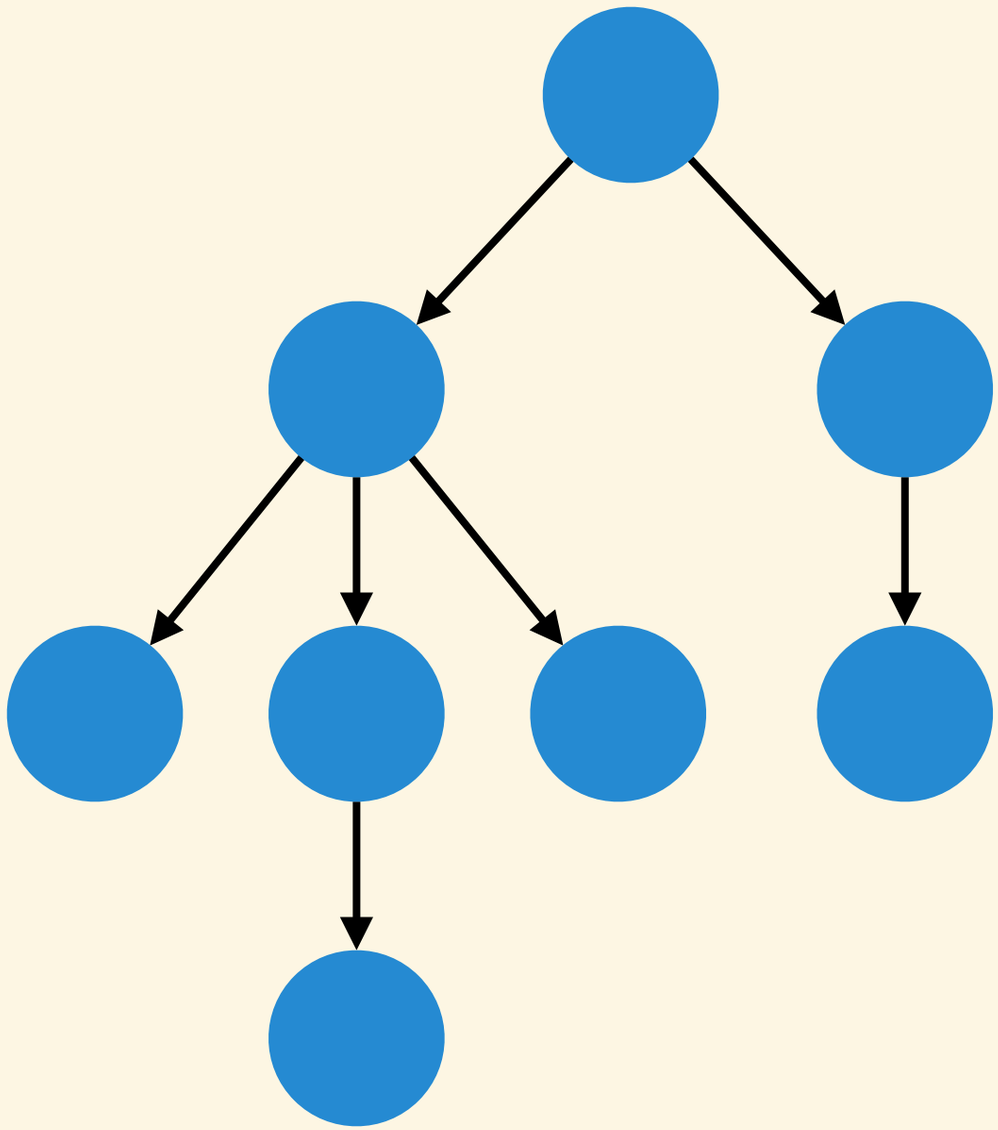


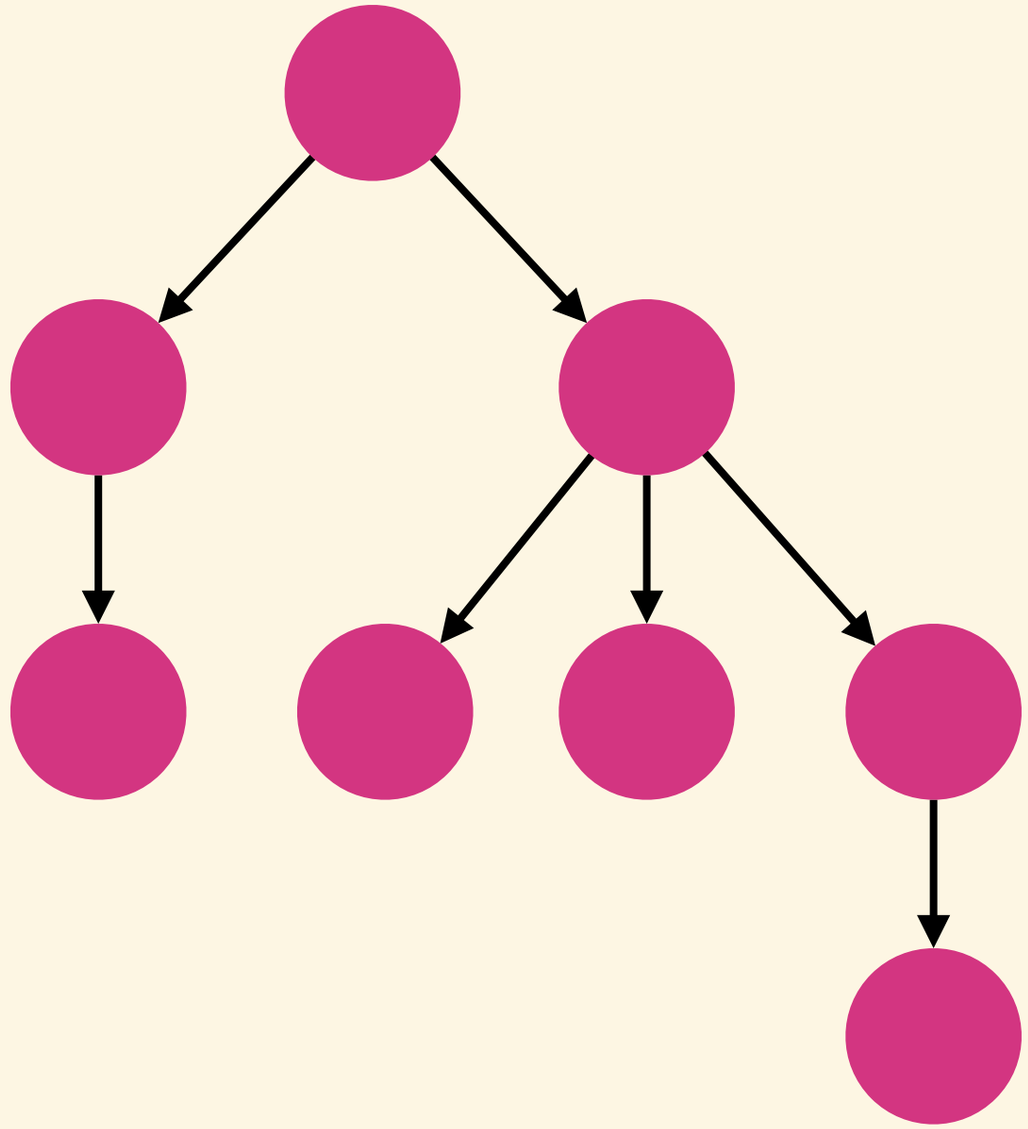
(((191 × 701 xor 29247) mod 34943) × 701 xor 33360)

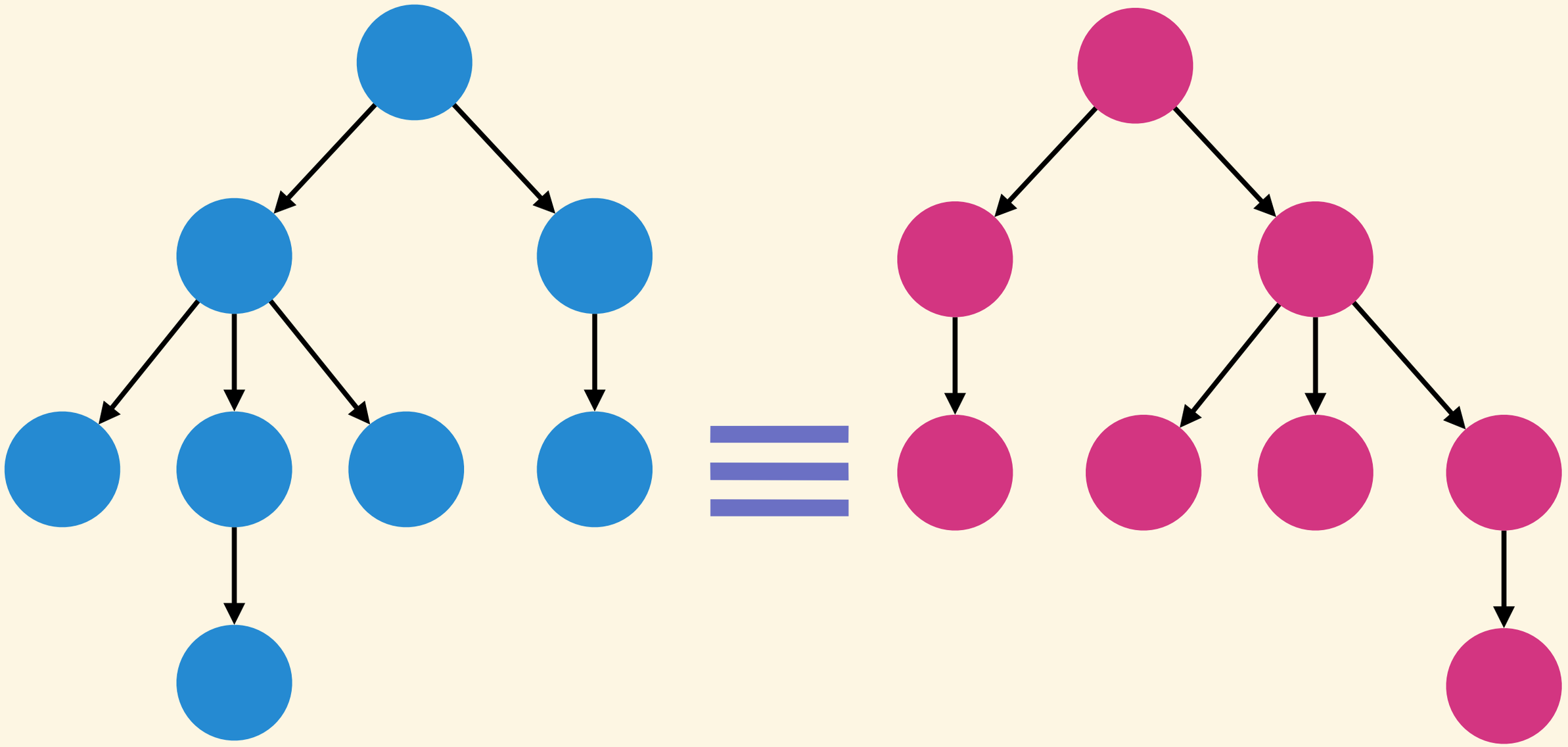
mod 34943 = 4687

hash value of the tree is 4687







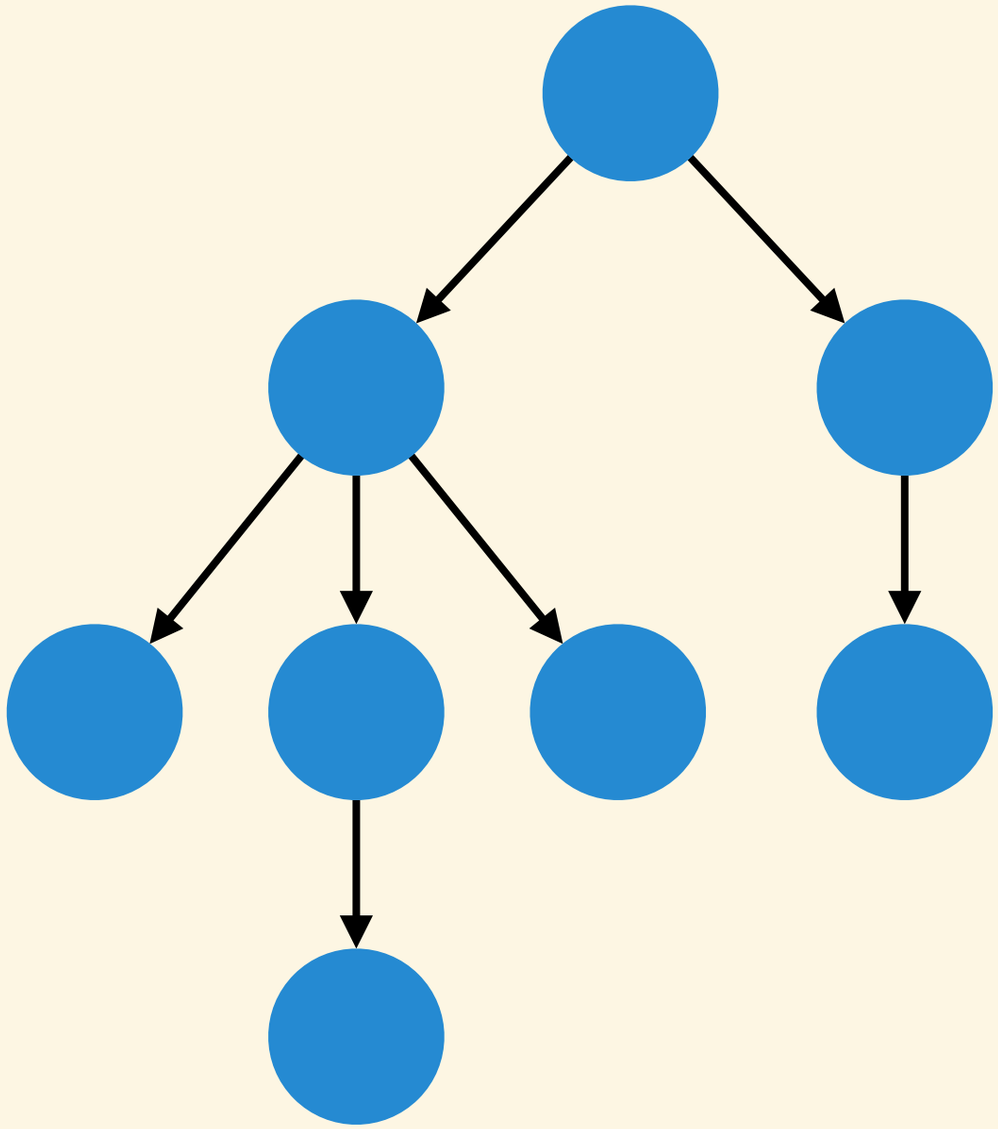


Algorithm

HASH_TREE(T):

1. hash all sub-trees
2. sort hash value of sub-trees (unique)
3. calculate hash value (any hash function)

Time Complexity



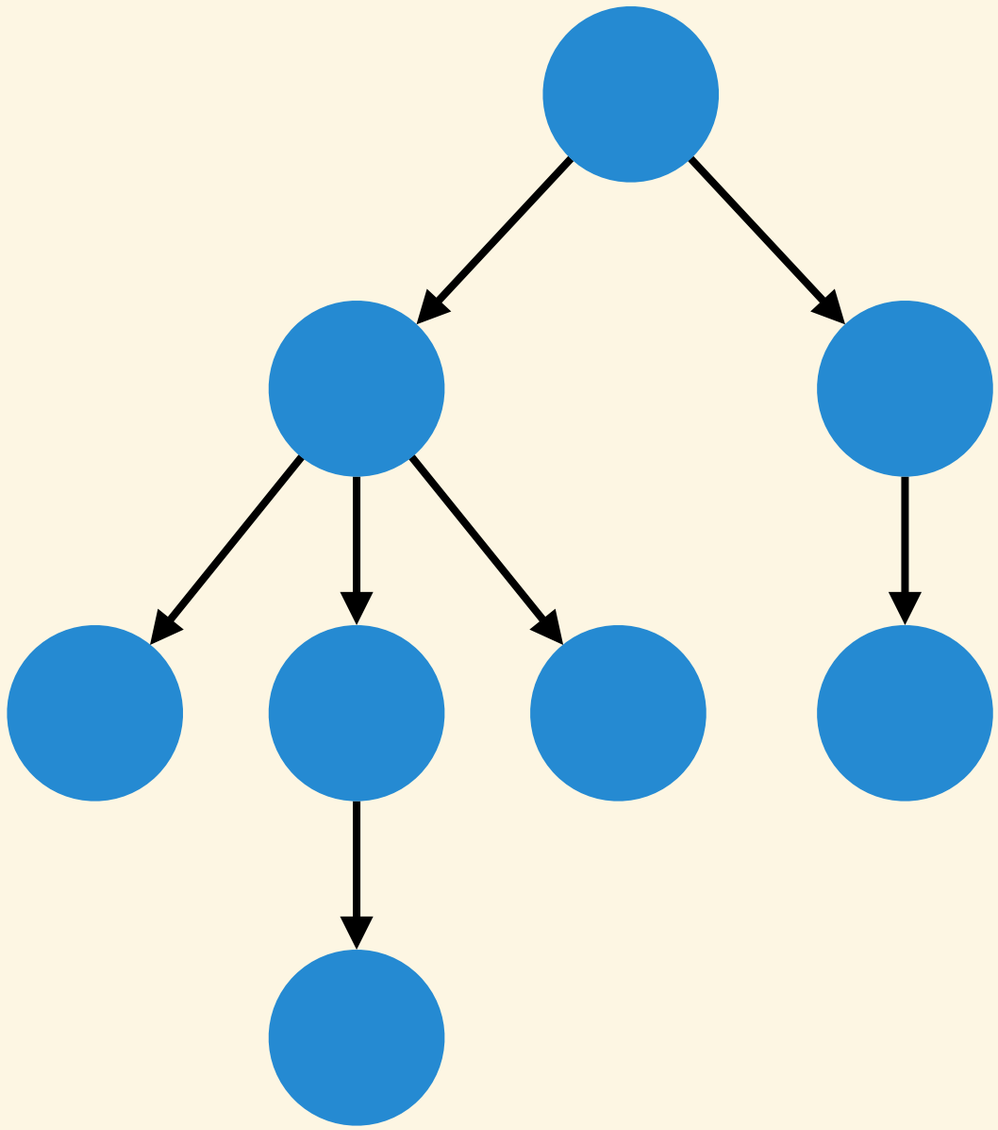
$O(Nlog_2N)$

N is number of vertices
height of tree

Source Code

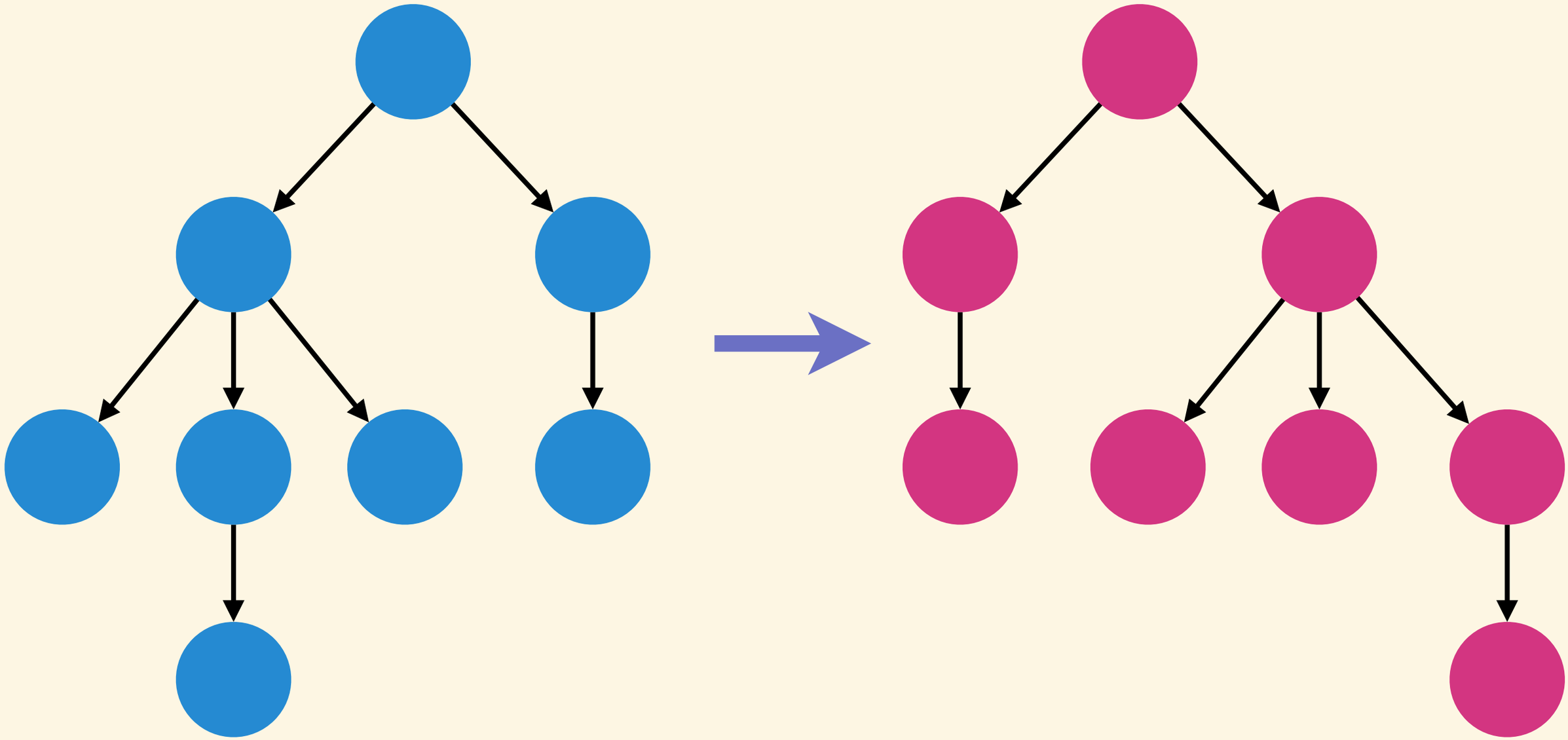
```
int hash( TREE &now, int root ) {
    int value = INIT;
    vector< int > sub;
    //get all hash value of subtree
    for ( int i = 0; i < now[ root ].size(); ++i )
        sub.push_back( hash( now, now[ root ]
[ i ] ) );
    //sort them to keep unique order
    sort( sub.begin(), sub.end() );
    //hash this this tree
    for ( int i = 0; i < sub.size(); ++i )
        value = ( ( value * P1 ) ^ sub[ i ] ) % P2;
    return value % P2;
}
```

Representation of Tree

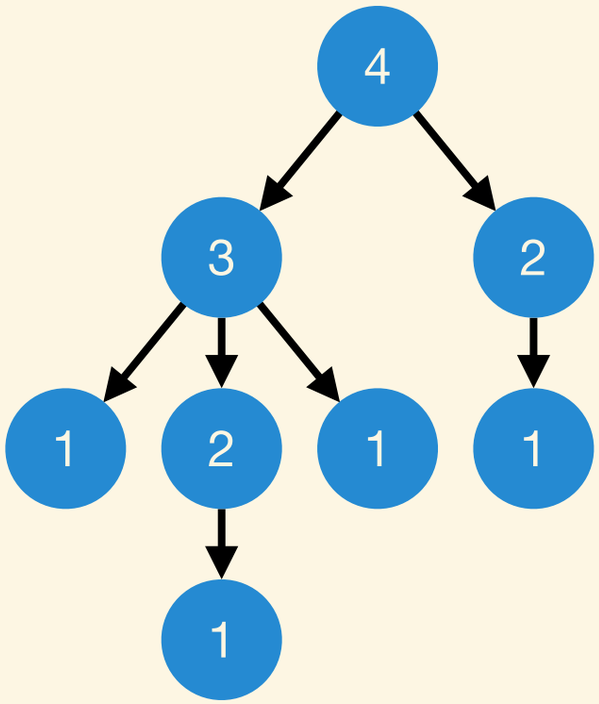


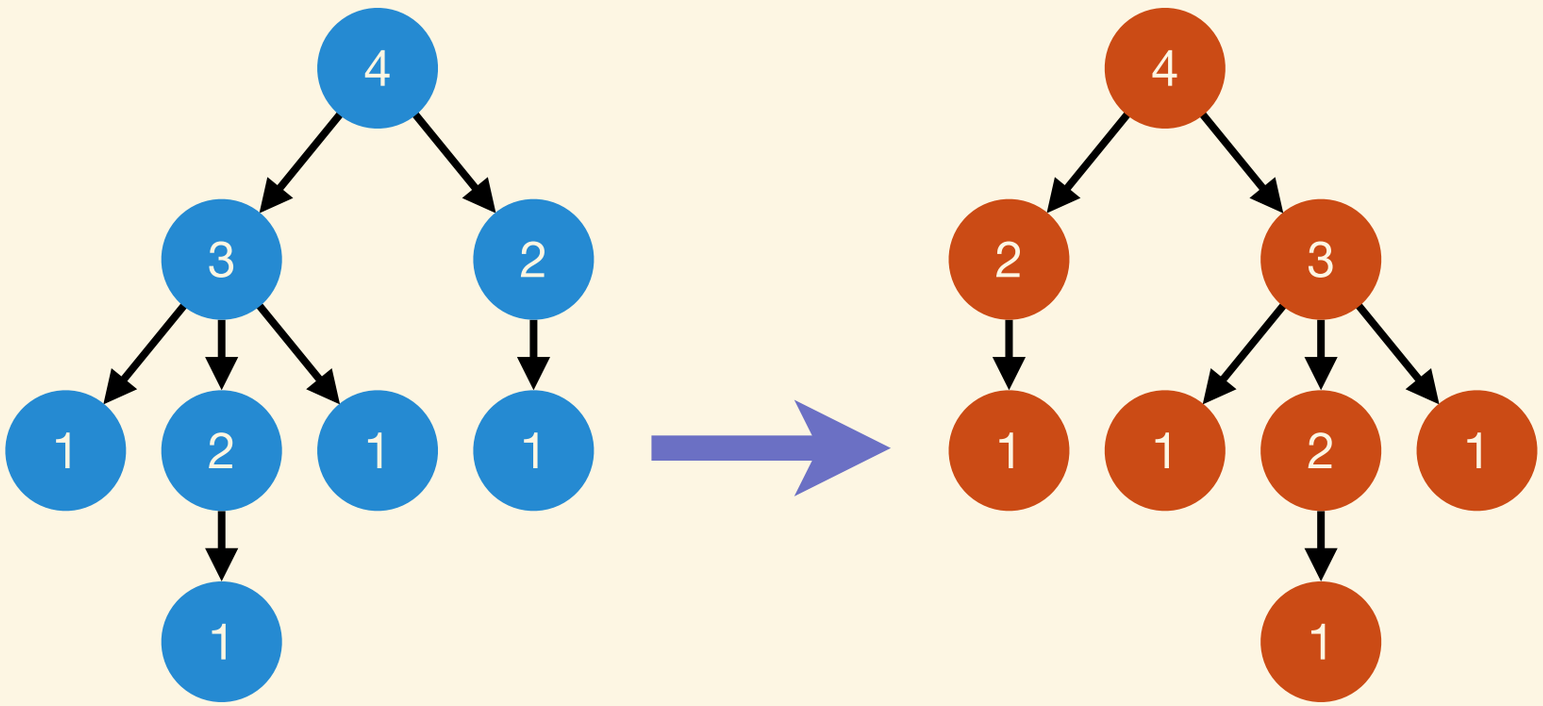
Let the height of left child less than the right one.

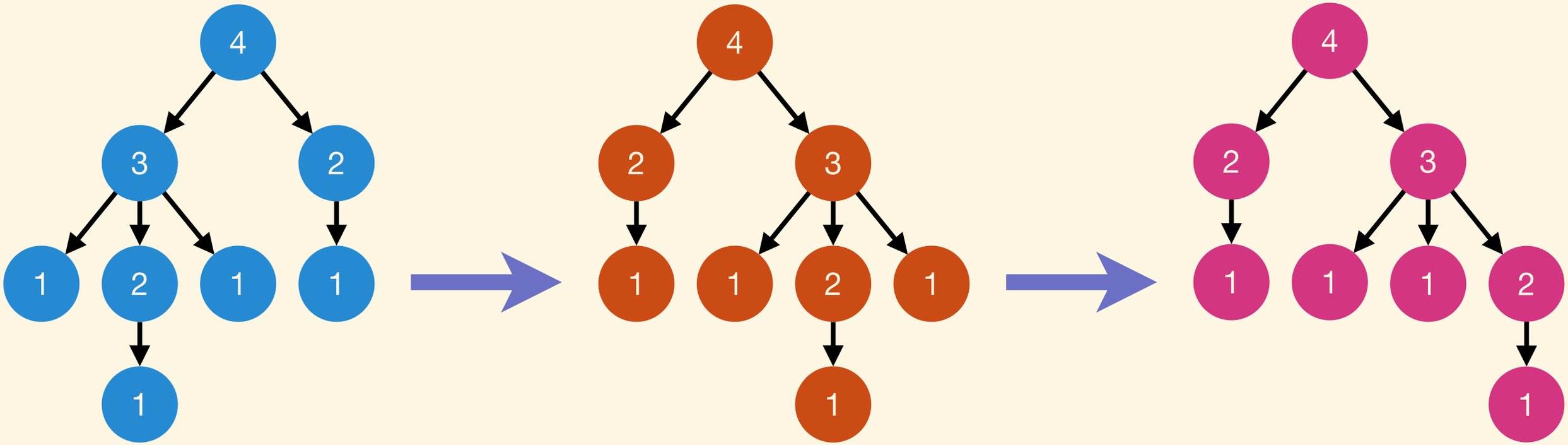
Representation of Tree



Let the height of left child less than the right one.


4
3
2
1
2
1
1
1

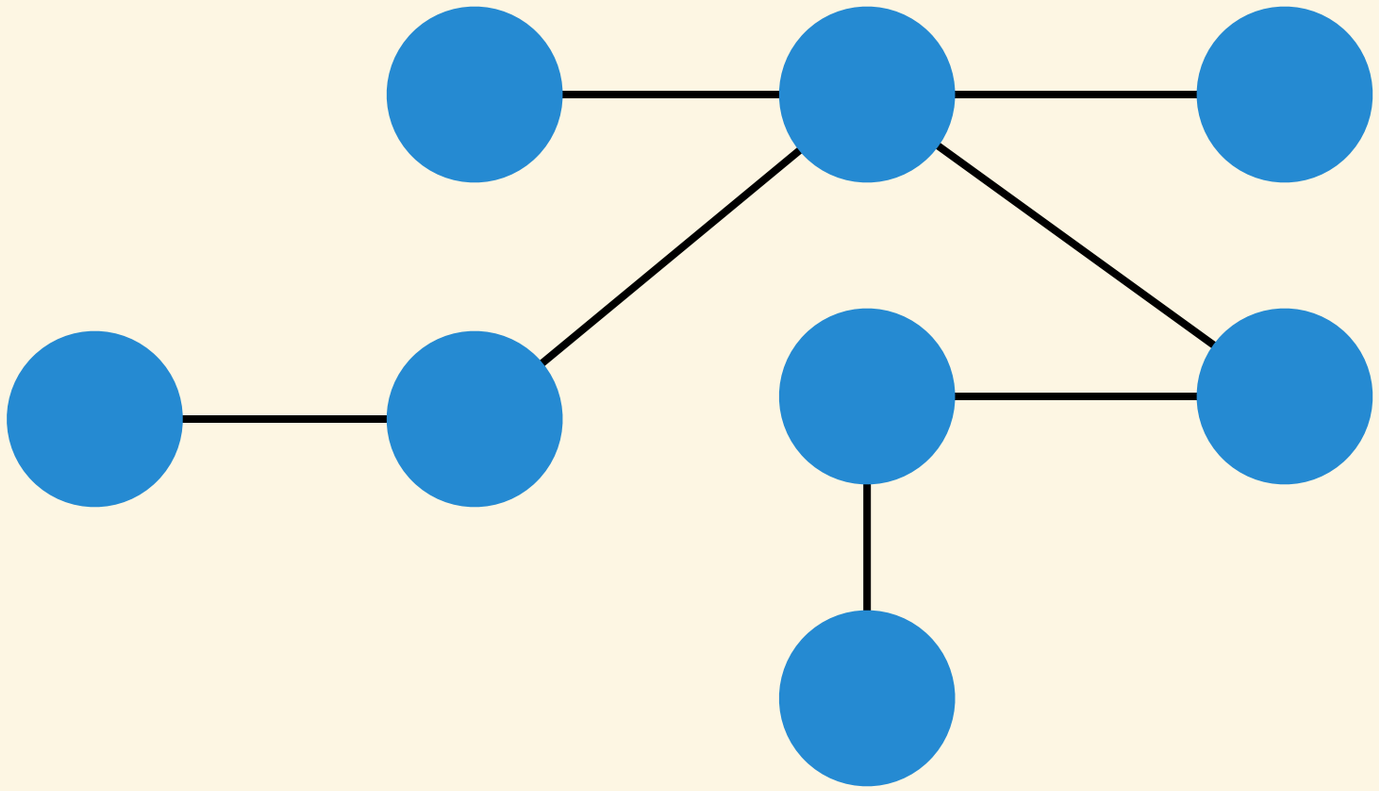

4
3
2
1
2
1
1
1
4
2
3
1
1
2
1
1


4
3
2
1
2
1
1
4
2
3
1
1
2
1
1
4
2
3
1
1
1
2
1

Algorithm

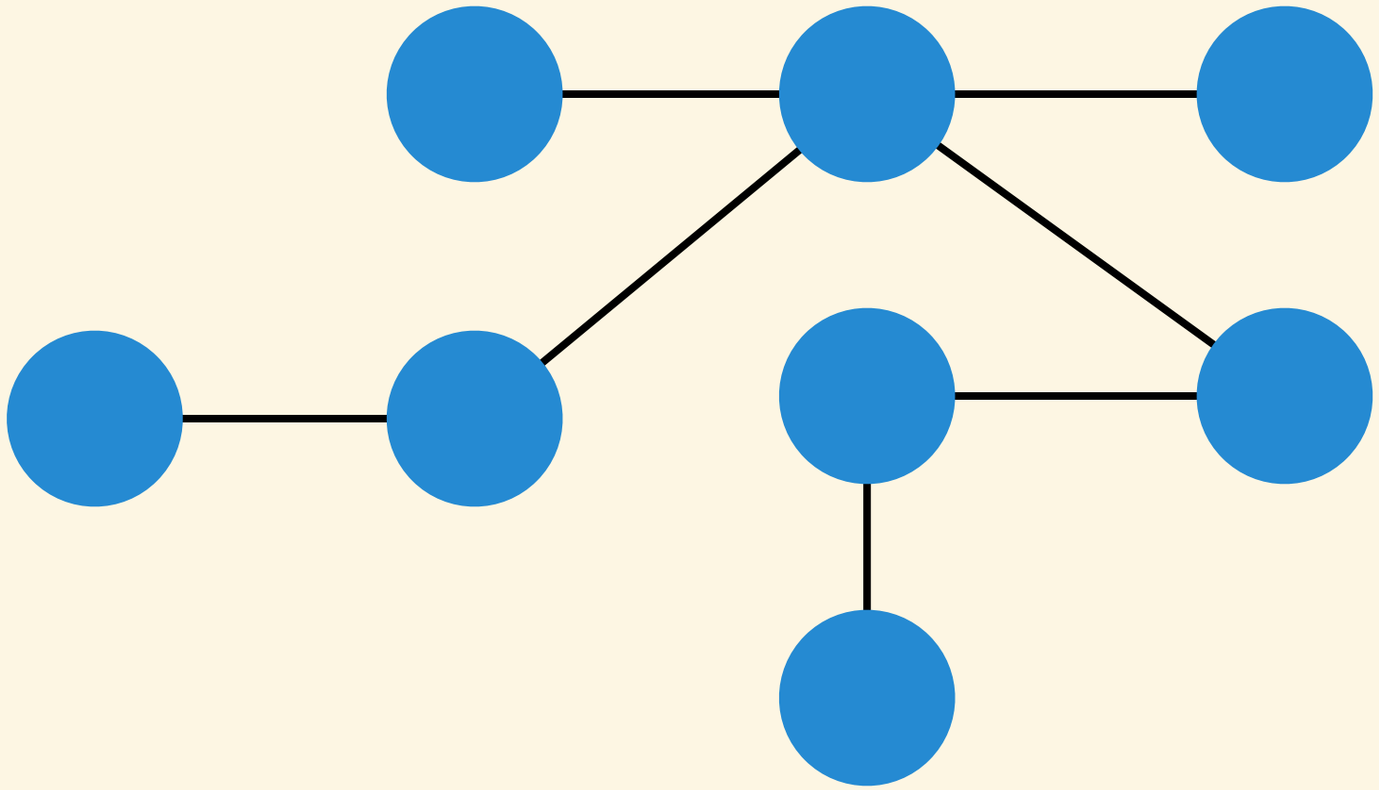
SORT_CHILD(T):

1. sort all sub-trees
2. compare the height
3. if height is equal, compare child recursively
4. put the lower at left and the higher at right

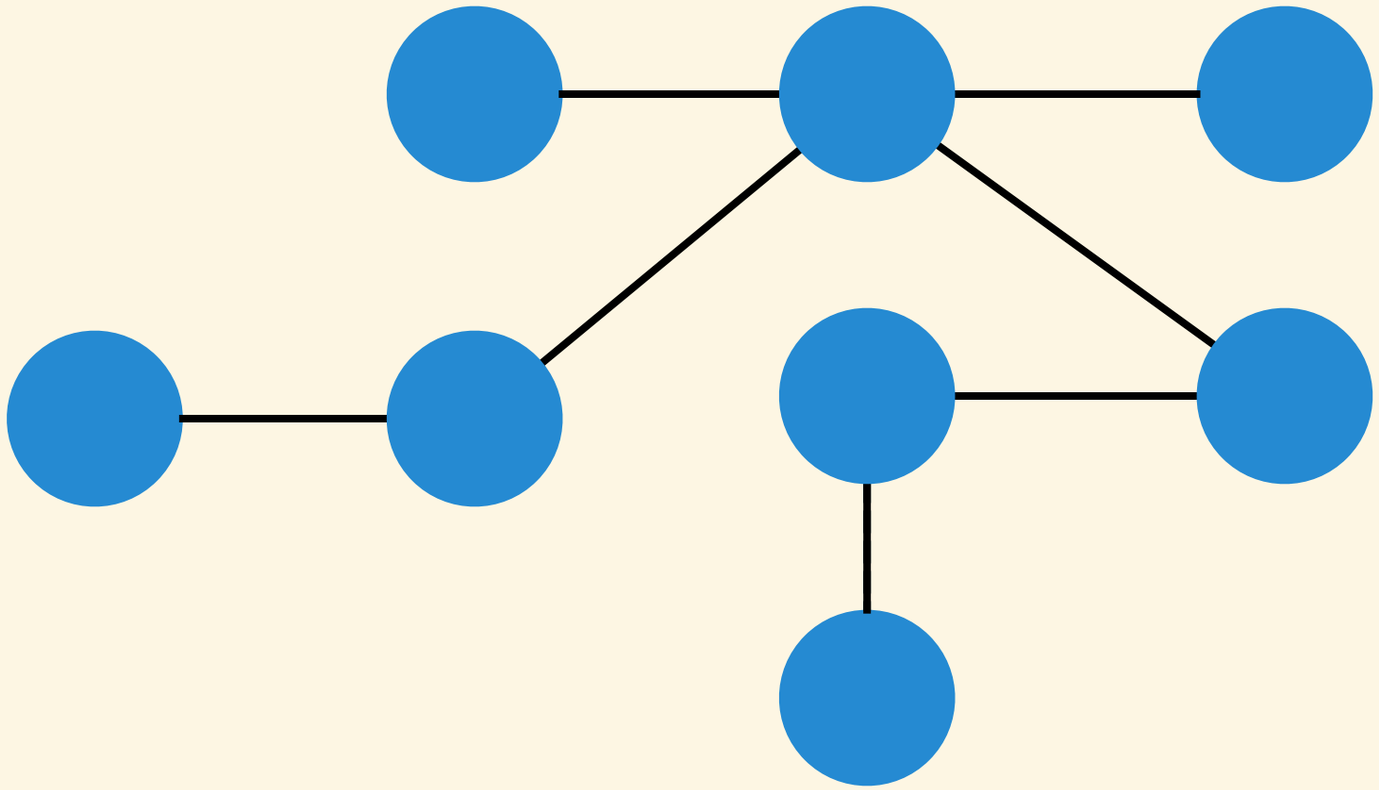


How about unrooted tree?

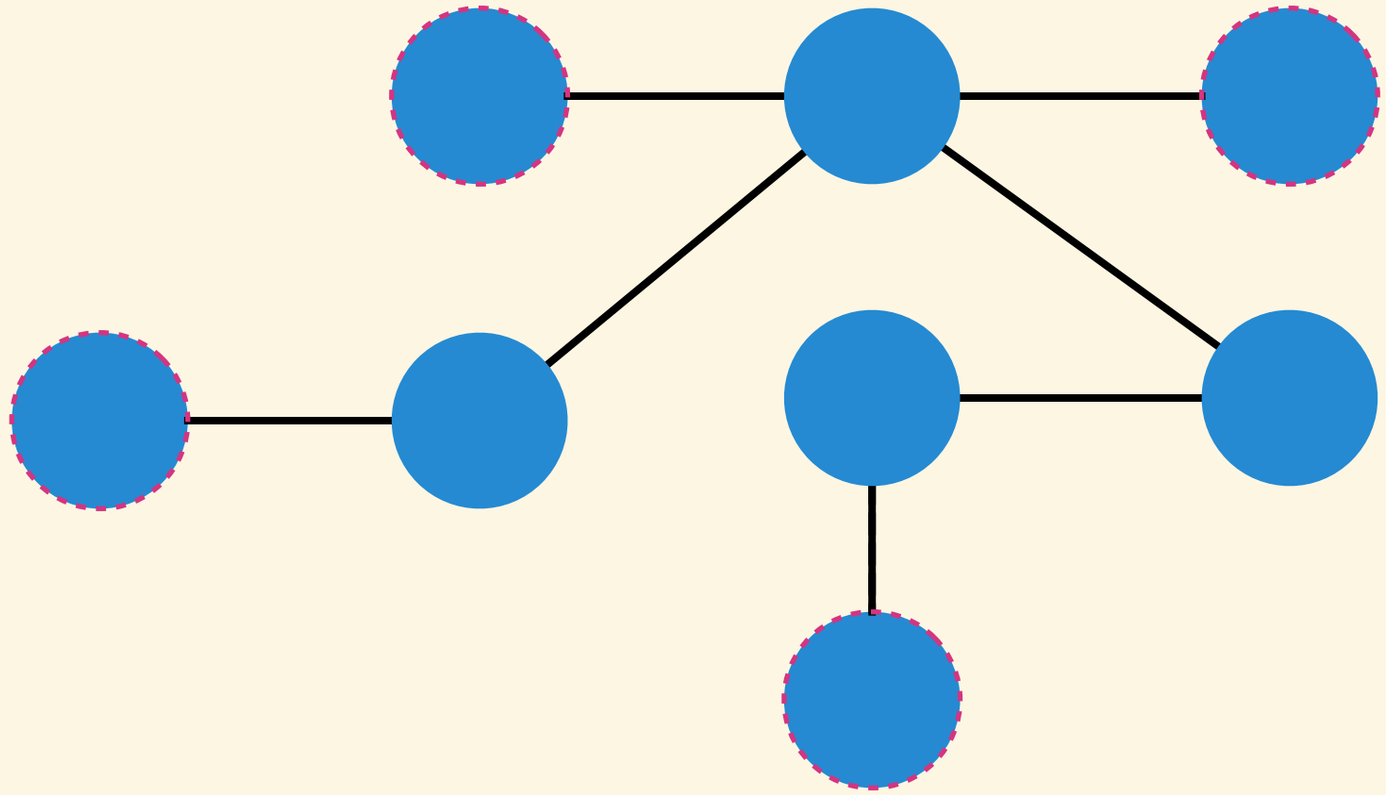
Find a Root



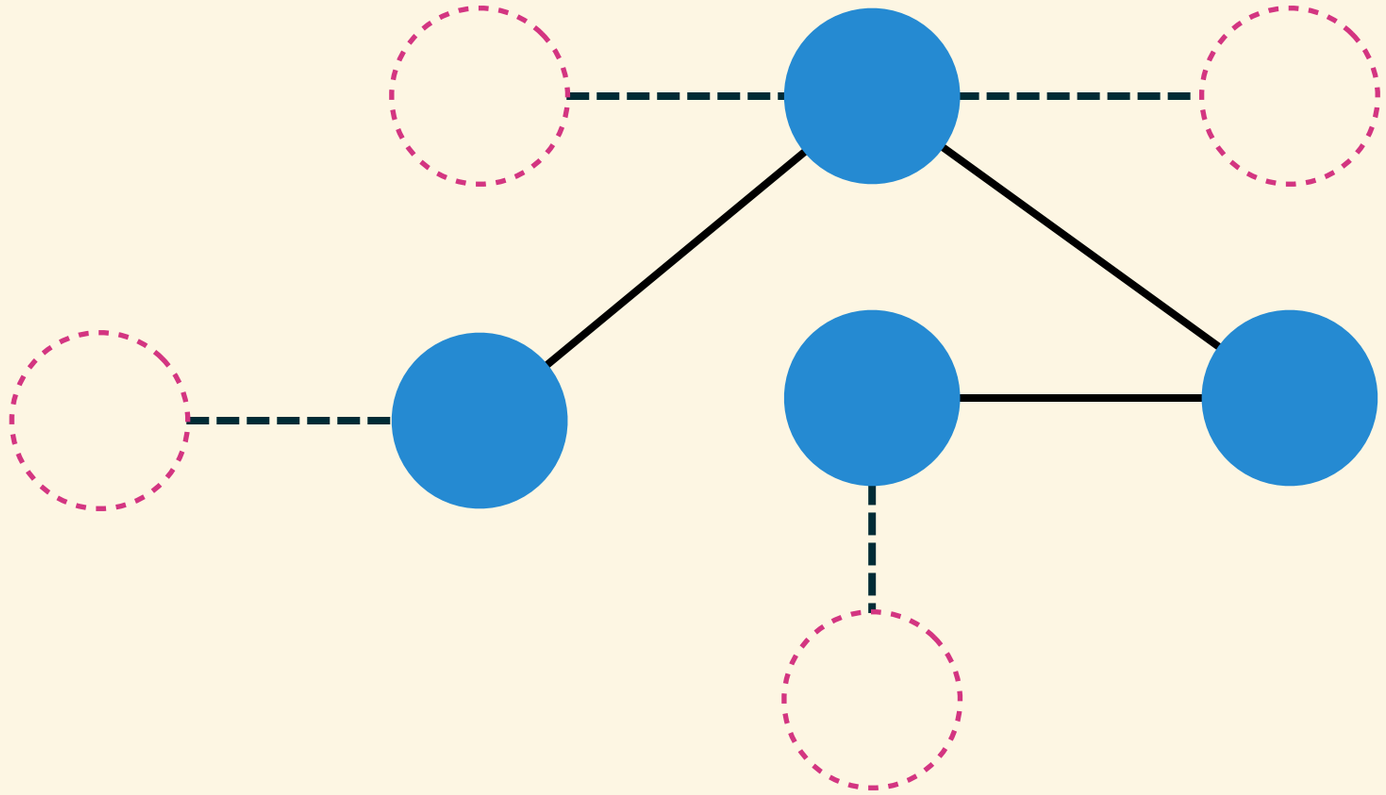
eliminate leaves



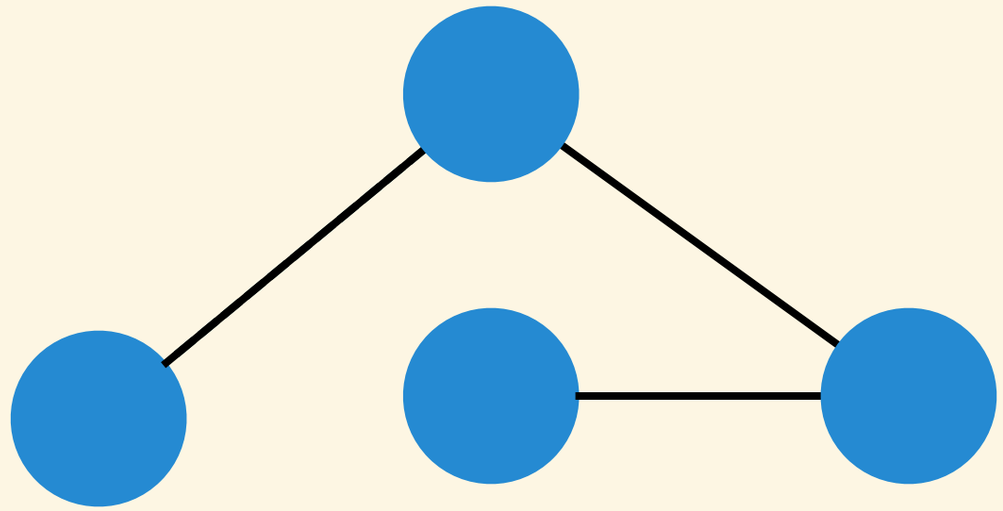
eliminate leaves



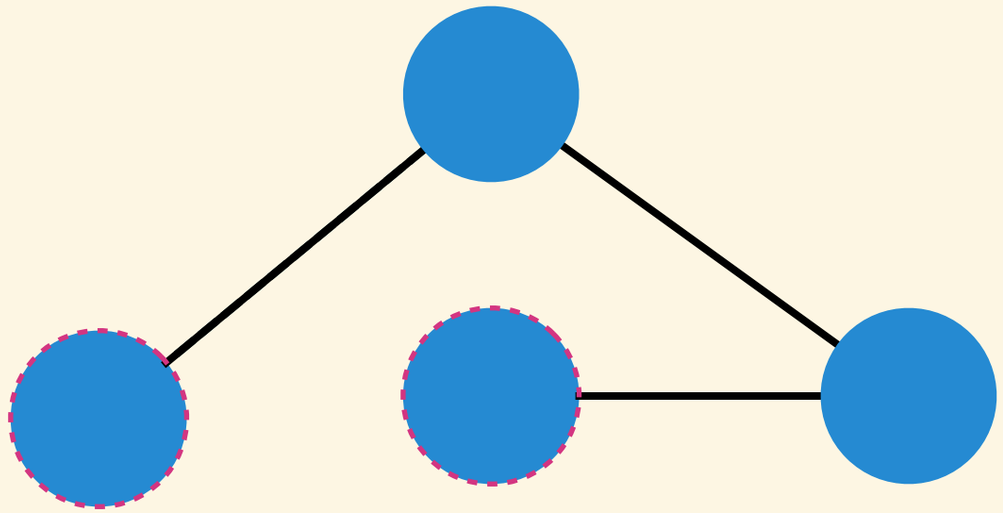
eliminate leaves



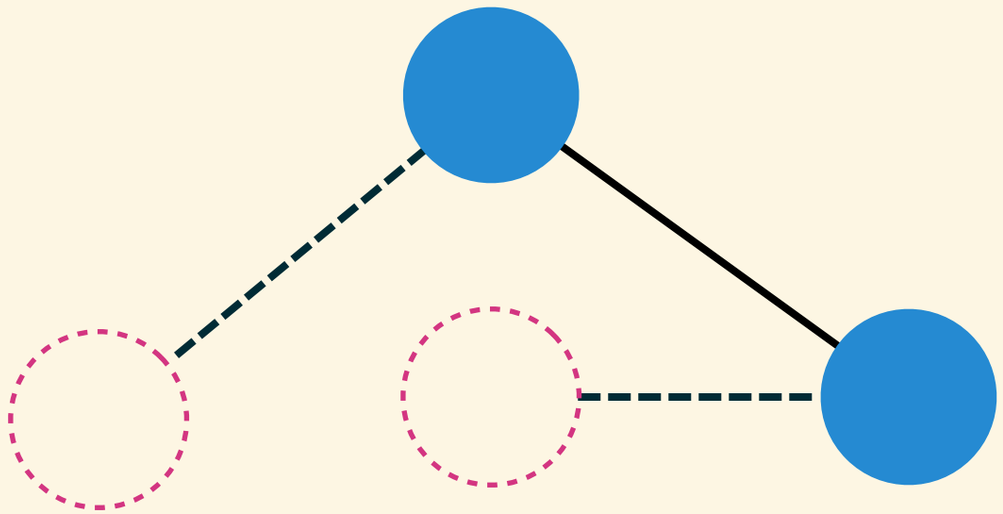
eliminate leaves



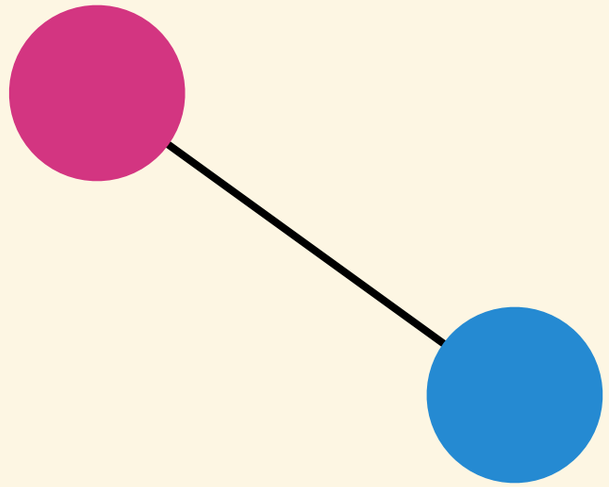
eliminate leaves



eliminate leaves

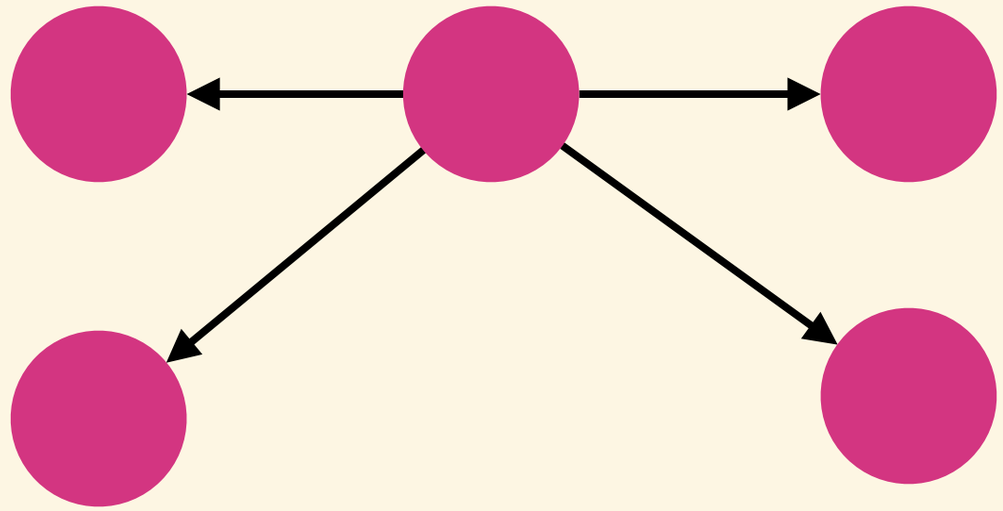


Enumerate Possible Root(s)

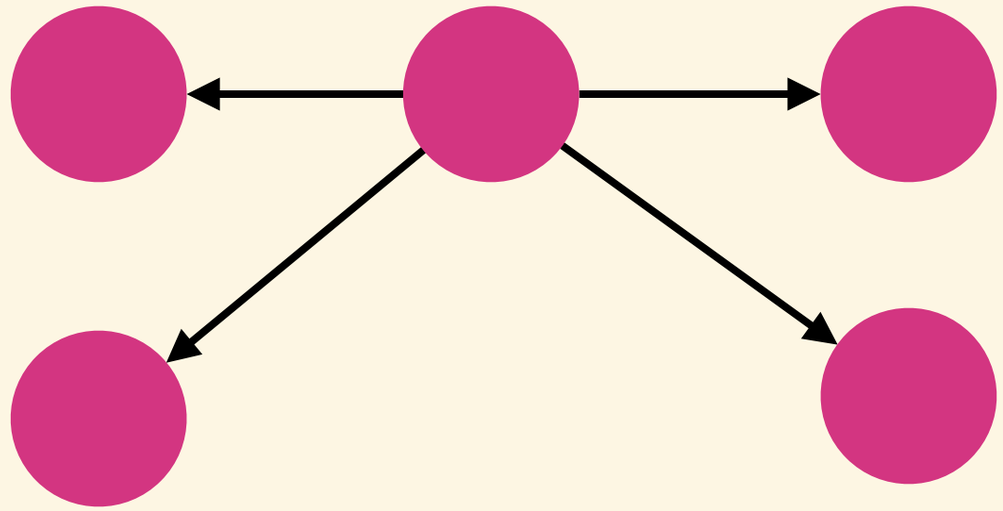


Rebuild The Tree

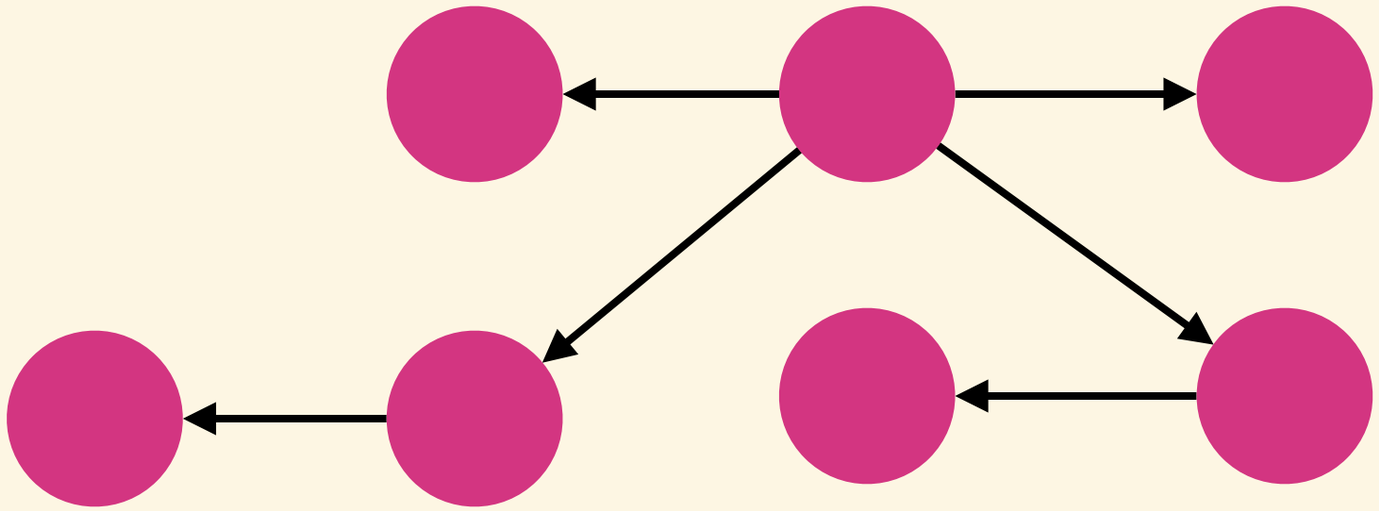
Rebuild The Tree



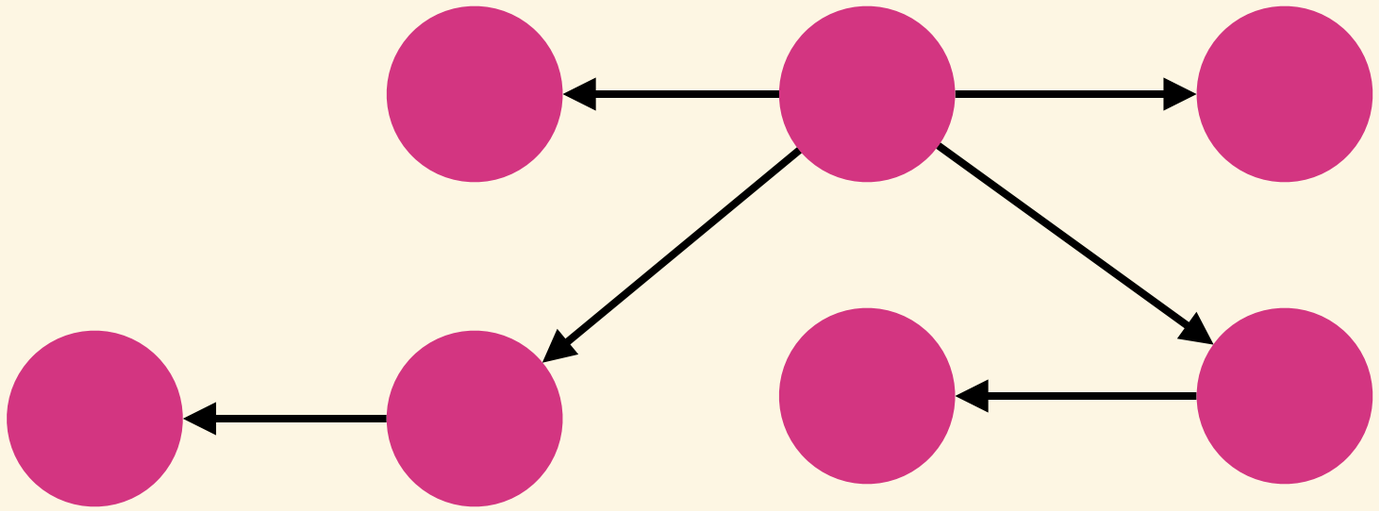
Rebuild The Tree



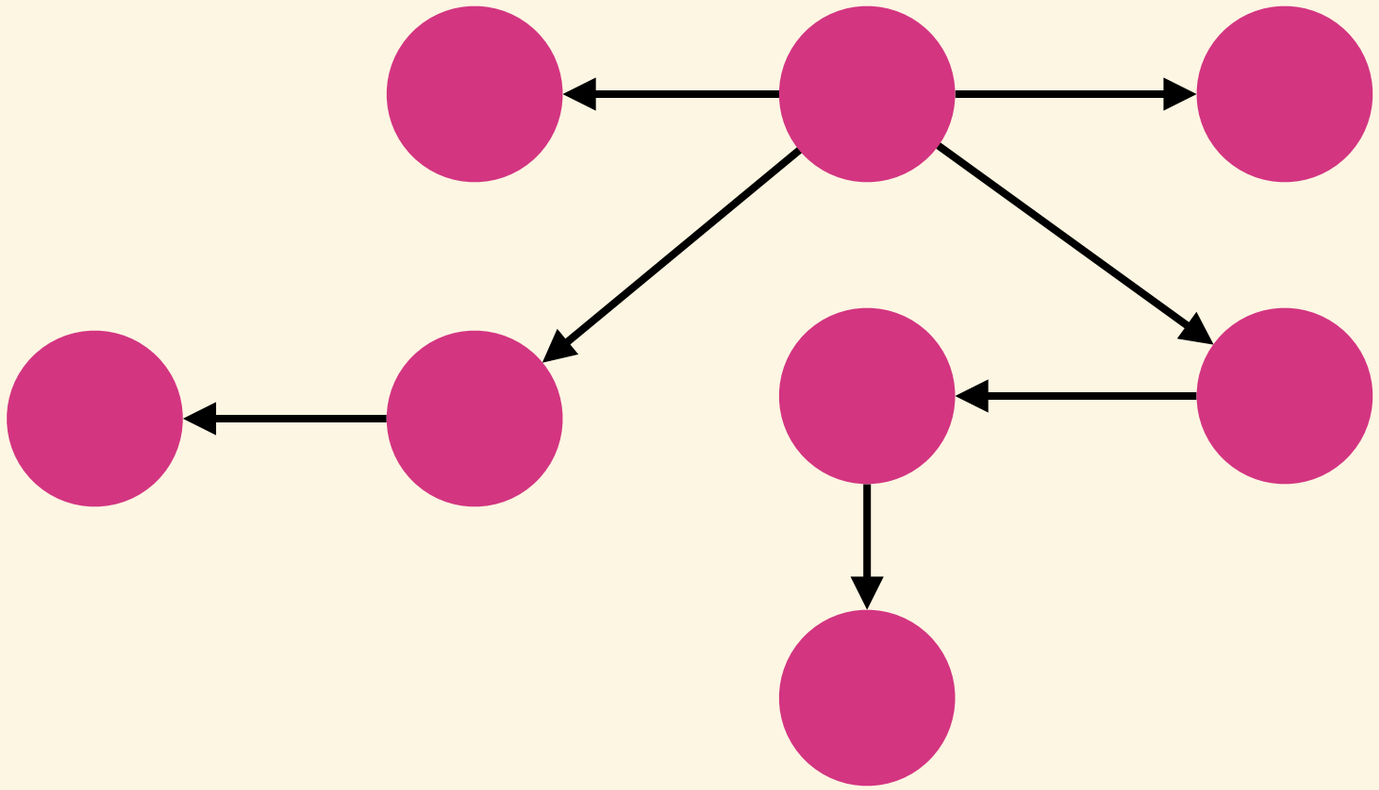
Rebuild The Tree



Rebuild The Tree



Rebuild The Tree



Apply the Isomorphism Detection

Practice Now

[POJ] 1635 - Subway tree systems

Thank You for Your Listening.