Centroid Decomposition

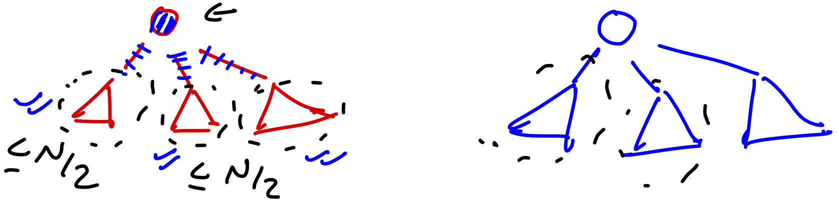
Course: https://unacademy.com/a/i-p-c-advanced-track

tanujkhattar@

Objective

- Centroid of a Tree
  - Definition
  - How to find the centroid of a tree
- Decomposing the “Original Tree” to get “Centroid Tree”
  - Implementation
  - Visualization
- Properties of the Centroid Tree
  - Show any path A -- B in the tree can be written as A -- Centroid -- B.
  - Maintain some information for the O(NlogN) paths to answer generic path Queries
- Problem Discussion

Centroid of a Tree



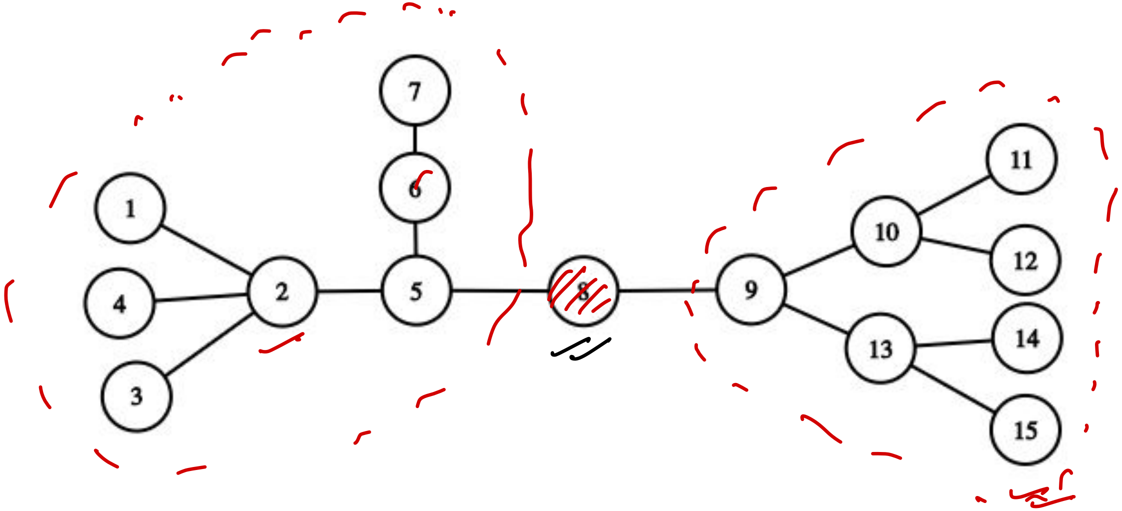
- **Centroid:** Given a tree with N nodes, a centroid is a node whose removal splits the given tree into a forest of trees , where each of the resulting tree contains no more than N/2 nodes.

Question: Which node is the centroid of this tree (on the right) ?

A. 5
B. 8
C. 9
D. 2

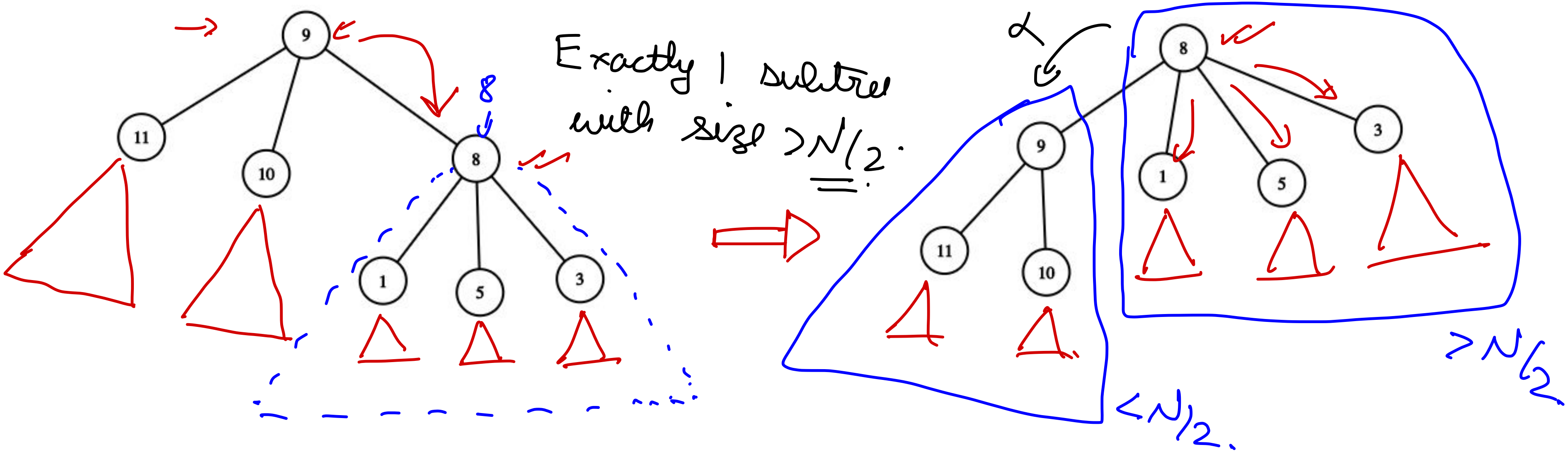
15

15/2 = 7



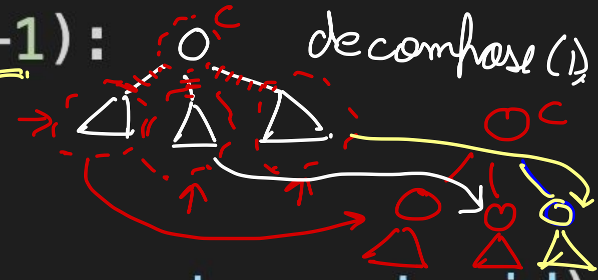
How to find a Centroid?

- **Theorem (Jordan, 1869) : For any given tree, the centroid always exists**
- **Proof:** Start with any arbitrary vertex and check whether it satisfies the property. If yes, we're done, otherwise there exists only one adjacent subtree with more than N/2 nodes. Consider the adjacent vertex u in that subtree and apply the same argument. Repeat, till you find the centroid.



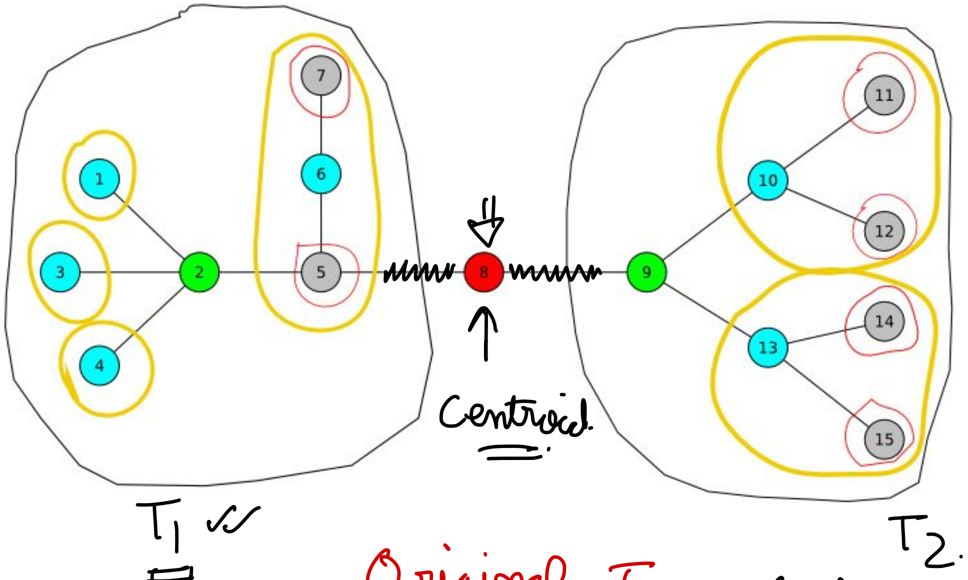
Decomposing the Original Tree to get "Centroid Tree"

```python
def decompose(root, centroid_parent = -1):
  centroid = find_centroid(root)
  if centroid_parent != -1:
    add_edge_in_centroid_tree(centroid_parent, centroid)
  for (adjecent_edge, adjacent_vertex) in G[centroid]:
    delete_edge(adjecent_edge)
    decompose(adjacent_vertex, centroid)
```



Decomposing the Original Tree to get Centroid Tree

Decompose(T) ⇒ Decompose(T_1) ; Decompose(T_2)



Question: Which node will be the root node of the Centroid Tree?

A. 2
B. 5
C. 8 ←
D. 9

Original Tree. (T)

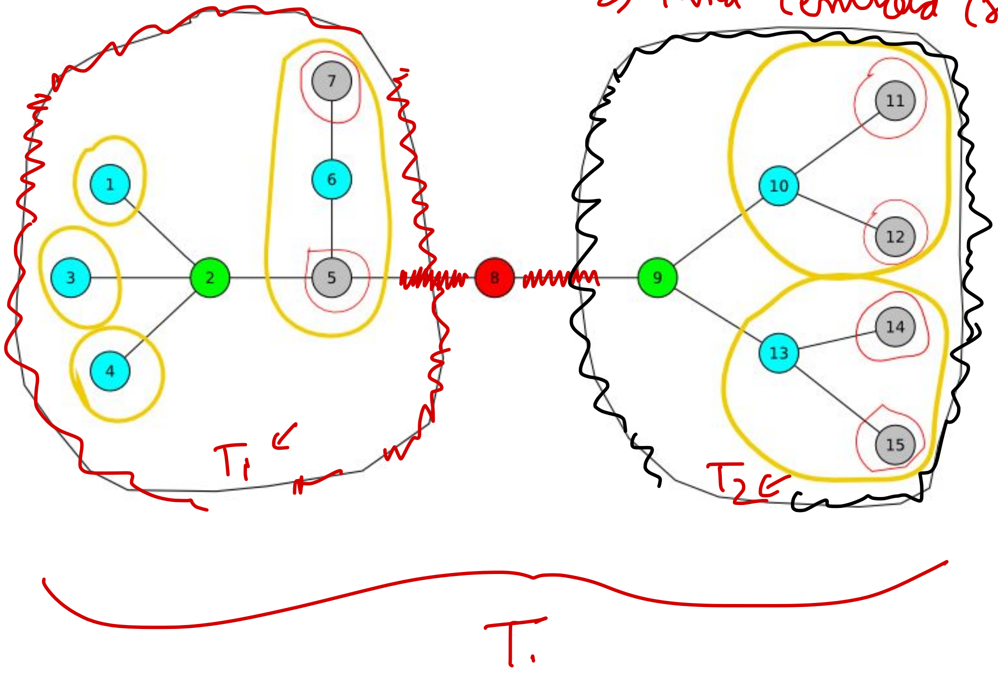
a. Decompose → Returns a rooted tree with centroid of the original tree as the root.

Decomposing the Original Tree to get Centroid Tree

1) Decompose (T)

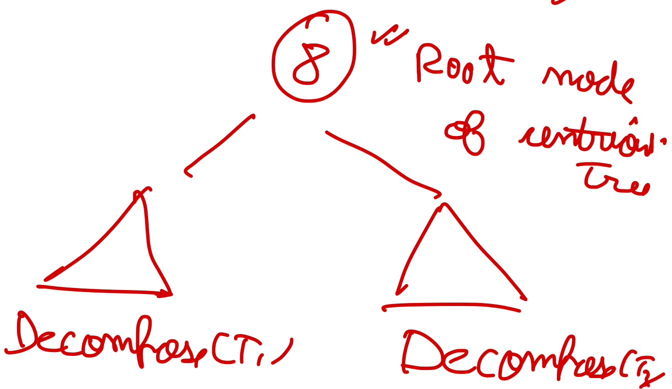
2) Find centroid (8)

3) Decompose(T_1) & Decompose(T_2)

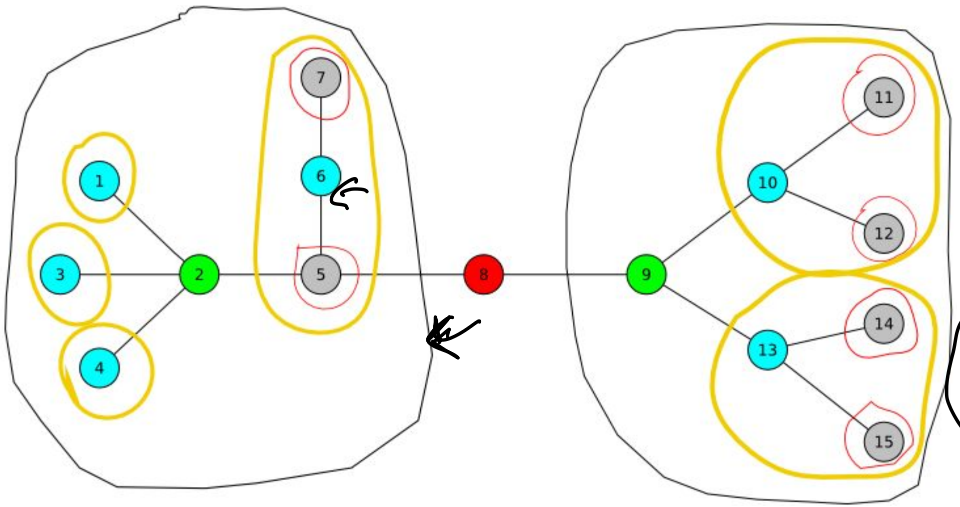


Question: How many children nodes would node 8 have in the Centroid Tree?

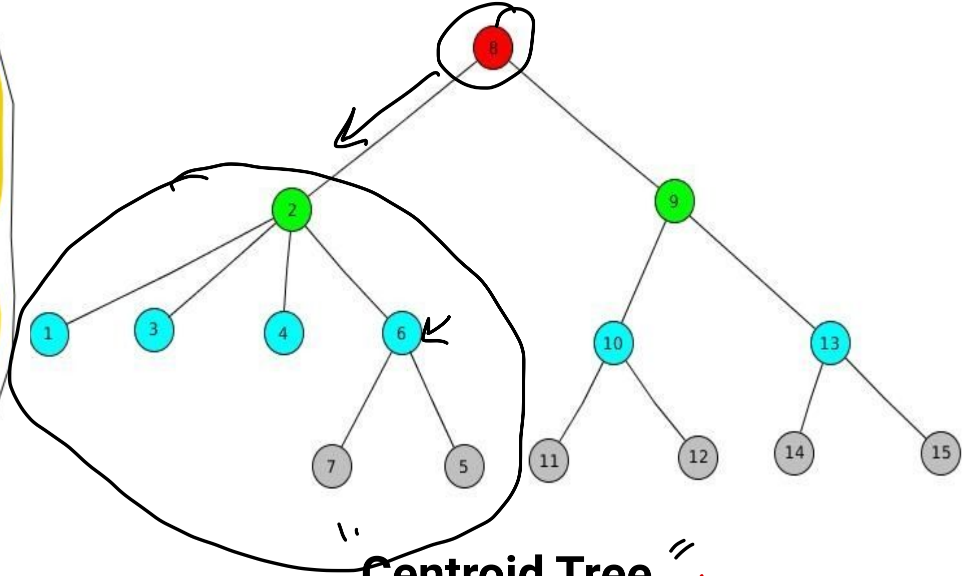
A. 1
B. 2
C. 3
D. 4



Decomposing the Original Tree to get Centroid Tree



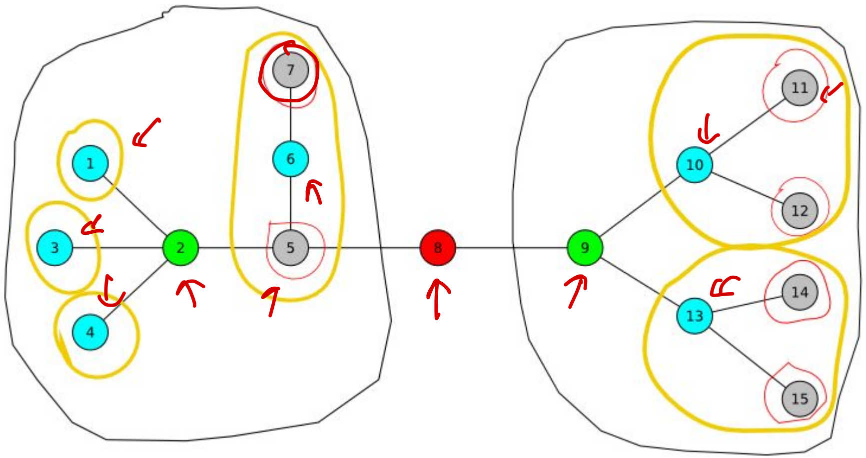
" Original Tree "

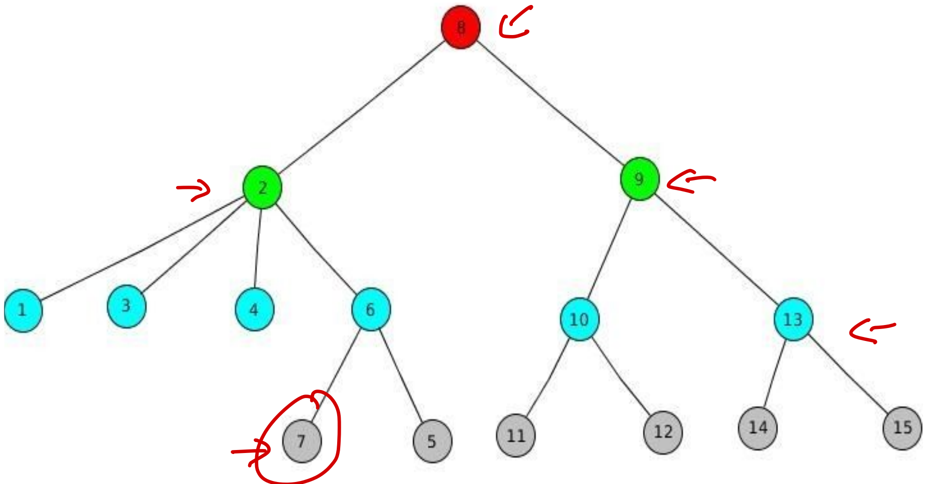


" Centroid Tree "

Properties of the Centroid Tree

- **Property-1: The Centroid Tree contains all N nodes of the original tree**
- Since each node will become a centroid of some smaller tree (maybe a tree consisting only of that one single node) , Hence the centroid tree formed would contain all the N nodes of the original tree.





Properties of the Centroid Tree

- **Property-2: The height of the centroid tree is at most O(logN)**
- Since at each step, the new trees formed by removing the centroid have size at-most N/2, the maximum no of levels would be O(logN). Hence, the height of the centroid tree would be at most O(logN).

Decompose(T)

D(T_1) D(T_2) -- D(T_k)

$N;\ N/2,\ N/2^2\ \text{---}\ N/2^{\log N}$

Size = 1

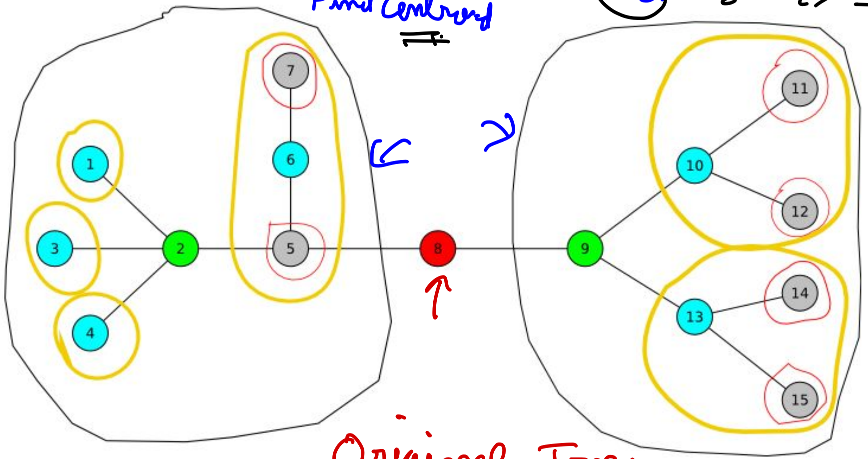
T → D(T_1), D(T_2)

Find Centroid

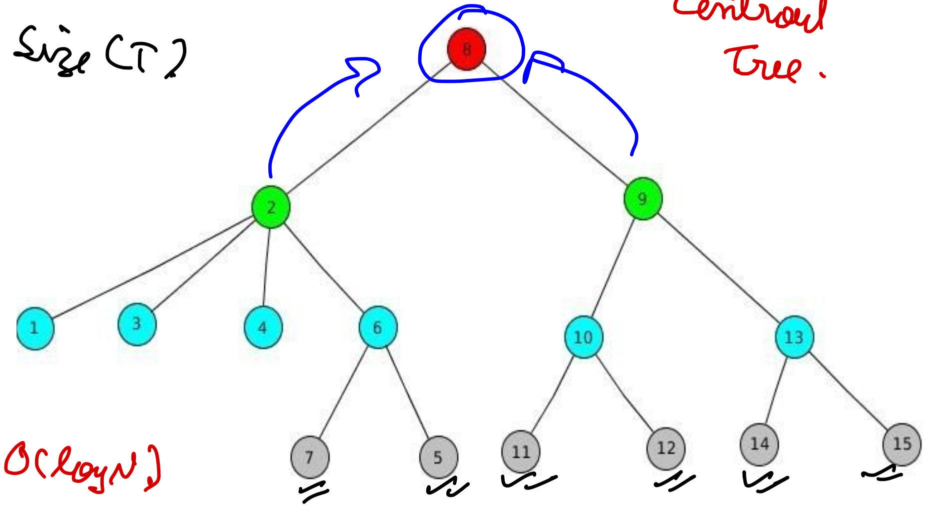
$Size(T_1) \le \frac{1}{2} Size(T)$

$Size(T_2) \le \frac{1}{2} Size(T)$

O(logN)



Original Tree.

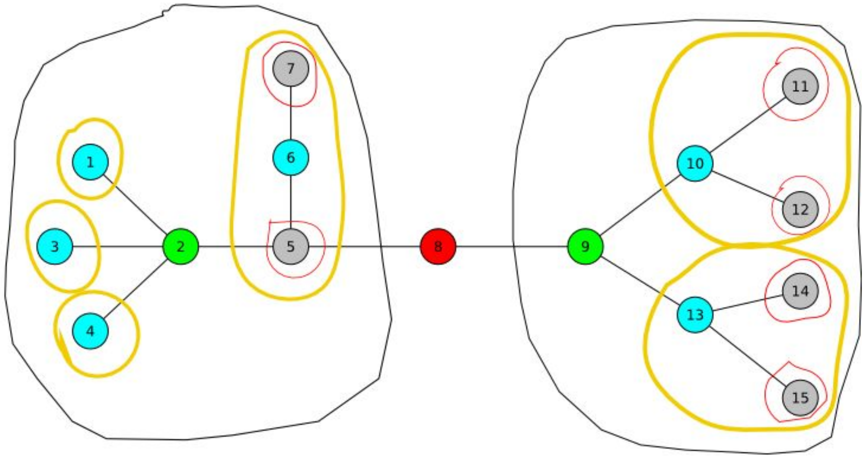


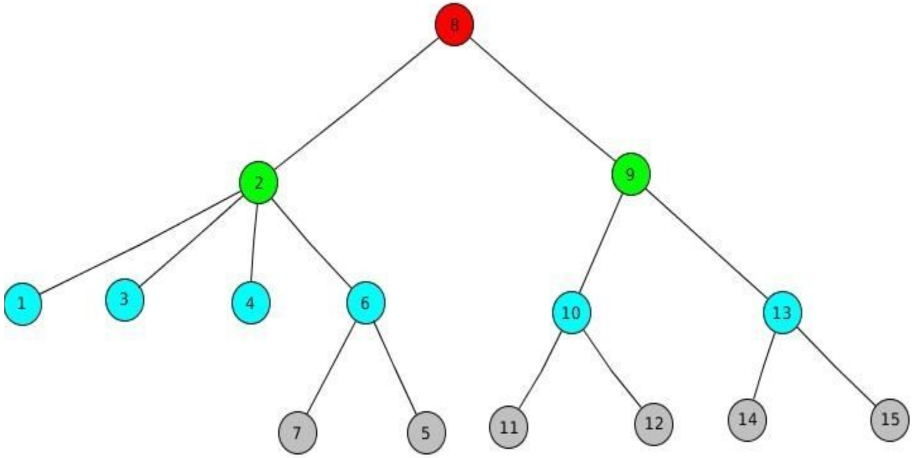
Centroid Tree.

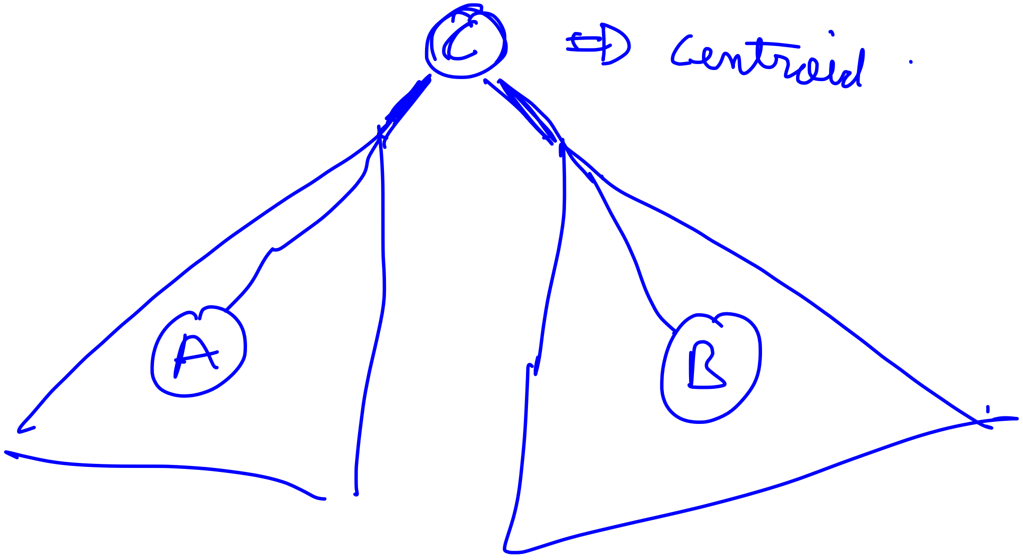
Properties of the Centroid Tree

LCA in Centroid Tree represents the first node which lies on path from A → B and were removed because it became a centroid.

- **Property-3: Consider any two arbitrary vertices A and B and the path between them (in the original tree) can be broken down into A-->C and C-->B where C is LCA of A and B in the centroid tree.**
- In the original tree, the first time A and B got disconnected was when we removed vertex C. Hence, A--B = A--C + C--B.

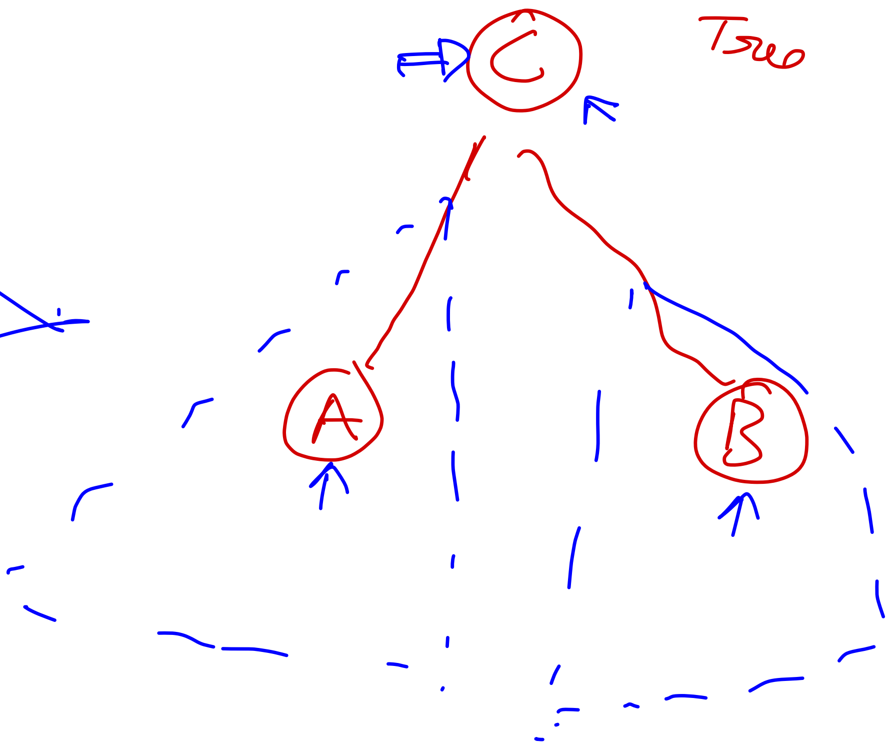






Original Tree.

Centroid Tree



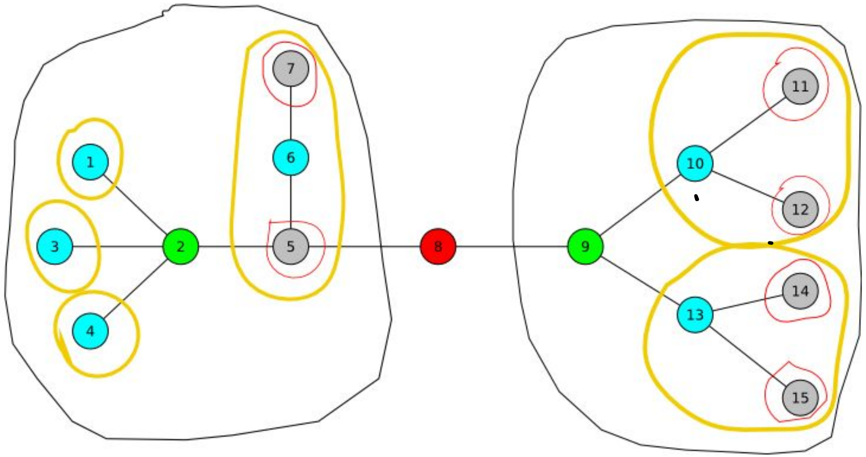
Properties of the Centroid Tree

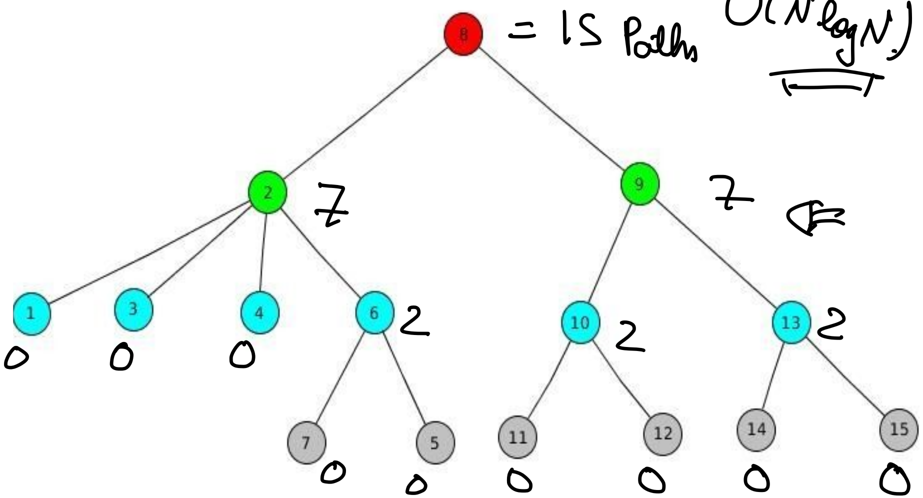
$N \rightarrow {}^{N}C_2 \sim O(N^2)$ Paths.

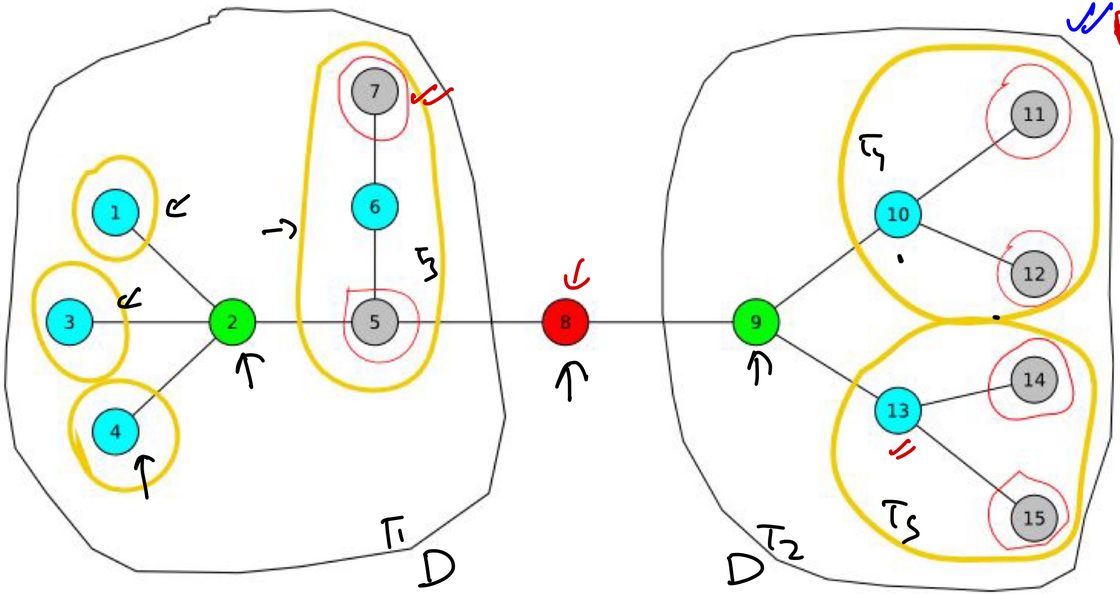
A, C

B, C

- **Property-4: Hence, we decompose the given tree into O(NlogN) different paths (from each centroid to all the vertices in the corresponding part) such that any path is a concatenation of two different paths from this set.**
- There are at-most O(N) special paths at each level - every path starts at the centroid of that subtree and ends in a vertex. Since there are logN levels, total number of paths are ~ O(NlogN).







Processed Path → (8,5); (8,2); (8,1)
(8,3) (8,4) (8,6)
(8,7) (8,9) (8,10)
(8,11) (8,12) (8,13)
(8,14) (8,15)

O(N log N)

T_1 : (2,5) (2,1) (2,3) (2,4)
(2,6) (2,7)

T_2 : (9,10) (9,13) (9,14) (9,15)
(9,11) (9,12)

T_3 : (6,7) (6,5)

T_4 : (10,11) (10,12)

T_5 : (13,14) (13,15)

Q: Dist(7,13) = Dist(8,7) + Dist(8,13)

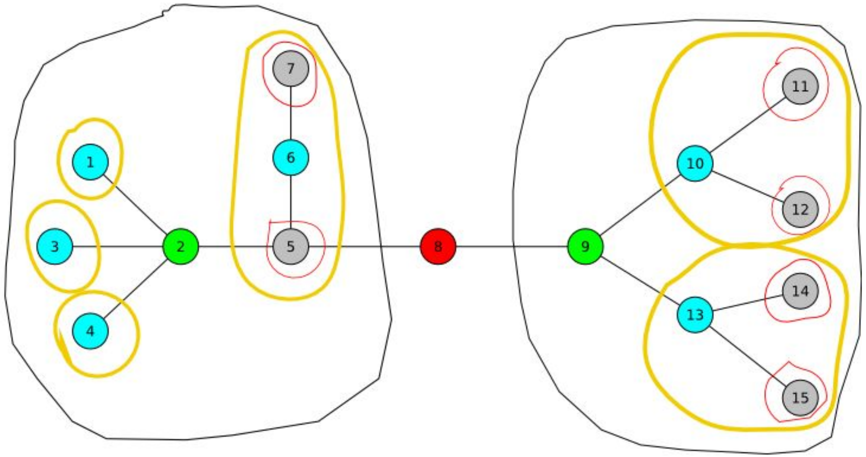
MinWTEdge(7,13) = min(MinWtEdge(8,7);
MinWtEdge(8,13))

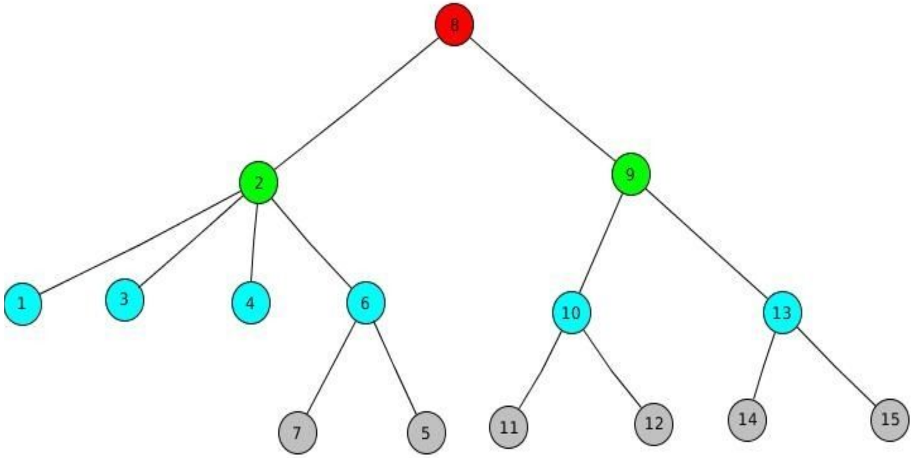
f(13,7) = combine(f(8,7); f(8,13))

Problem Discussion

Q1: Given a tree with N nodes and Q queries of the form u, v - return the sum of elements on path from u to v.

- Do Centroid Decomposition and precompute dist[LOGN][N]
- Find LCA of u,v in the centroid tree: ans = dist[level(LCA)][u] + dist[level(LCA)][v]





Problem Discussion

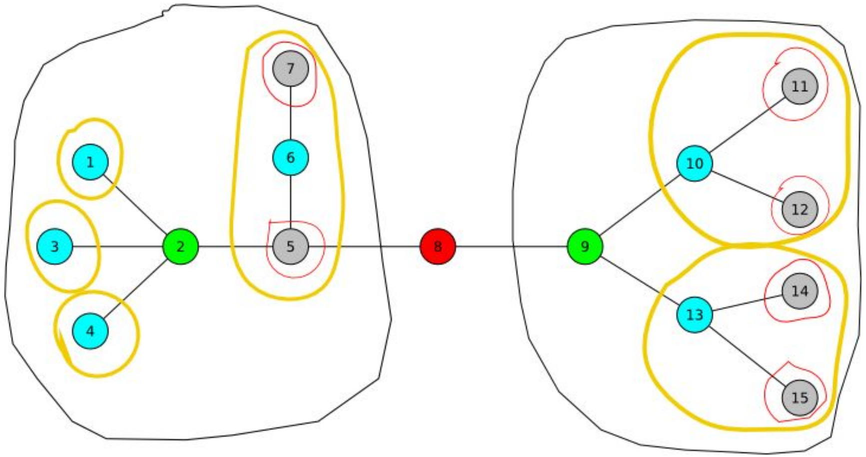
Q2: Given a tree with N blue nodes, there are Q queries of the form

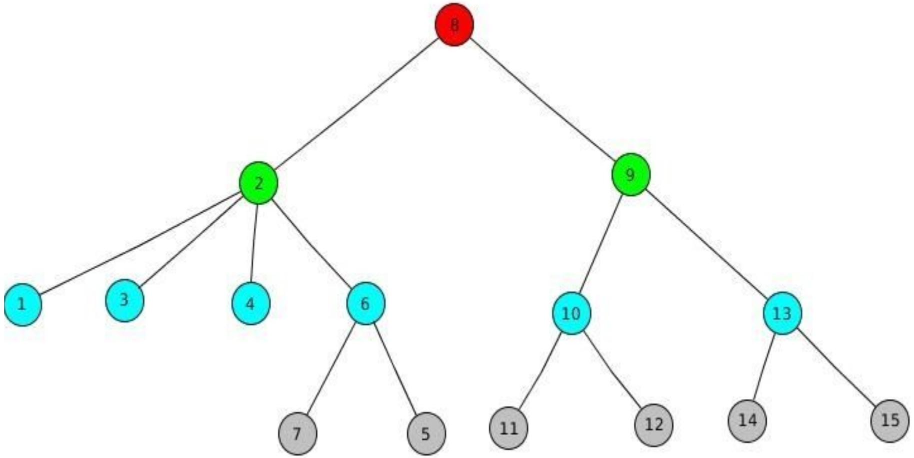
***Update v:* Paint node v red.**

***Query v:* Find the distance of closest red node to node v**

Problem Discussion

- Let ans[i] denote the min distance to a red node for the centroid "i" in its corresponding part.
- For each update, to paint a node **u** red, we move up to all the ancestors **x** of **u** in the centroid tree and update their ans as ans[x] = min(ans[x], dist(x,u)) because node u will be in the part of all the ancestors of u.
- For each query, to get the closest red node to node **u**, we again move up to all the ancestors of **u** in the centroid tree and take the minimum as mn = min(mn, dist(x,u) + ans[x]);





Problem Discussion

Q2: Given a weighted tree, find the no. of pairs of nodes the distance between which is a prime number.

Naive Solution

- Do a DFS and for every node, count the prime paths that pass through this node.
- O(SubtreeSize(x) * NumberOfPrimes) for every node x.
- In worse case, this could be O(N^2 * P).

Problem Discussion

Q2: Given a weighted tree, find the no. of pairs of nodes the distance between which is a prime number.

Solution Using Centroid Decomposition

- Do the same thing as earlier, but now by smartly rooting the tree only at centroids.
- For each centroid, we find the no of nodes at distance “i” from the centroid in it’s part and store it in dist[i].
- Iterate over all primes and find number of “matching paths” from other subtrees by looking at dist[Prime[j] – distance(i,centroid)].
- At each level, we spend O(N * P) -- therefore O(N * P * logN) total.

Further Reading

- https://tanujkhattar.wordpress.com/2016/01/10/centroid-decomposition-of-a-tree/