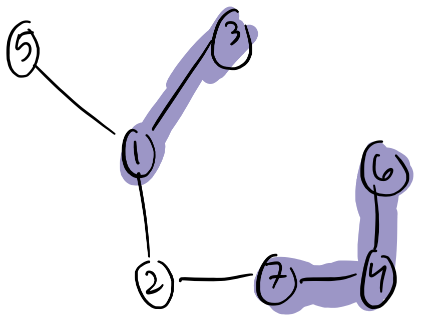
Query on a Tree

- For a given Query S the graph becomes a set of connected components for example



if $S = \{3, 1, 6, 4, 7\}$

- In each Connected component if the size is C_i then there is $\binom{C_i}{2}$ Pair of connected vertices

- If we Know for all connected Components their size the answer is simply $\sum_{i=0}^{\#cc} \binom{C_i}{2}$ where C_i is the # of Nodes in that component

- How do we actually keep track of connected components since performing a dfs per Query gives us $O(Q \cdot N)$ which is quite slow

- For each query keep a dsu of size $|S|$ important since making each dsu size $o(n)$ will lead to memory limit

- Then at each Node in the tree keep a set of all Queries that contain that node.

- Then we just perform a dfs & for all neighbours (u,v) if they both contain the same query then perform a join(u,v) for that dsu

- For efficient mapping of indices in each dsu I used a set

- The total runtime is bounded by the sum of all Query sets $O(\sum |S| \cdot \log |S|)$