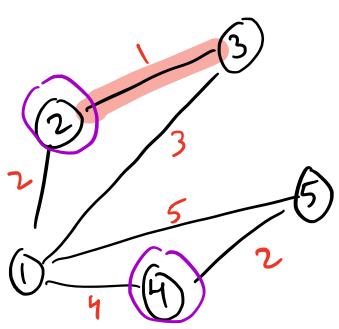
Stamp Rally

- Problem Summary: You are given an undirected graph with edge values 1-m. Answer Q queries for each query you are given x, y, z where x & y are two nodes you explore some nodes starting from x & y until you visit z unique nodes and take the maximum edge number, you minimize this number

Example



for Query 2, 4
if you move from 2->3 you visit 3 nodes {2, 4, 3}
and the value is 1

Bad solution: for each Query x, y, z you can do this

1. Iterate from 1 to m [call this e]
   1.1 Dfs starting from X & y and only take edges of number $\leq e$
   1.2 If we visit at least z nodes then e is the answer

- This works but is $O(Q) \cdot O(m) \cdot O(n+m)$ which in short is bad!! But its a good start
- First optimization: we are doing a lot of unnecessary work lets say we find an e value that works then we know all values $\geq e$ work as well. This property means we can use binary search. Now we reduce to $O(Q) \cdot O(\log m) \cdot O(n+m)$ still TLE

- Observation 2: This naive approach has a lot of redundancy. Instead of individually answering each Query lets try to solve all at once to save some work

- To do this lets start with a graph of N nodes but 0 edges. And now add the edges in sorted order.
- Lets say we have added i edges and we want to know if Query j has been achieved then we have 2 cases:
  Case 1: x and y are in the same component at time i then we just check if that component size is $\geq z$
  Case 2: x and y are in different components. In this case we add the sizes and compare to z
- This still is too slow taking $O(n) \cdot O(Q)$. But we can improve this with binary search
- For all Queries Q store two numbers L_i, R_i which is the current range of that Queries binary search. Then perform $\lceil \log(m) \rceil$ iterations of adding all edges in order. At the start of each iteration compute $M_i = (L_i + R_i)/2$ and at that time i check if the Query is valid and update L_i, R_i Accordingly

- To actually check component sizes use a disjoint union set to add edges in $O(1)$

- In total we have $O(\log(m))[O(Q) + O(M)]$ complexity