10) → KL algorithm.

- Uses difference between number of intracommunity edges & intercommunity edges as the optimisation function.
- A good partition contains more density within community.
- Simple.
- requires prior information, so inefficient
- Complexity of $O(n^2)$
- not suitable for large networks.

→ Faster Newman Algorithm.

- Speeds up the community mining process.
- Has a compexity of $O(mn)$
- Starts off with each node in a separate community.

$$Q = \sum C_{ii} - a_i^2$$

- Iteratively combines nodes to form bigger communities until some pre-defined halt condition / until one large community has been found. Since only neighbors considered, risk of local optimum
- Constructs the dendrograph in a bottom-up manner; $a_i = \sum_j C_{ij}$ where C_{ij} is the edge weight between node i & j

→ GA Algorithm:

- Uses simulated annealing in order to control local search based optimisation.
- Stochastic Global Search optimisation is used in this method. It finds global optimum
- Randomness is introduced into this model.

$$P = \begin{cases} 1 & \text{if } C_{t+1} < C_t \\ e^{-(C_{t+1} - C_t)/T} & C_{t+1} \geq C_T \end{cases}$$

where p is the probability of the node being in the graph.

C_{t+1}, C_t are parameters at time t & t+1 whereas T is the temperature.

Initial parameters: Starting configurations, temperature. (controls SA)

No prior information is required before starting. The time depends on SA's exec time.

Randomness is implemented by selecting/deleting/merging & splitting node groups.

(13)

→ (continuing KL algorithm)

In this method, different configurations are tested by randomly picking different nodes & by calculation of objective function.

The algo terminates when the evaluation metric does not improve.

→ Potts Algorithm.

A network is considered a multi-step Potts' model, with each node having a spin of value q.

The best partition is the most stable one with least energy.

In this stable form, the nodes with a

particular spin forms a community inherently.

∴ The aim is to split differently spinning models.

Further approximations to the objective function of the given algorithms by means of Monte Carlo optimisation techniques can lead to improvement in speed, while compromising on the accuracy a little.

13) Let Alice be 0.
Bob 1
Dave 2
Carol 3

Network X code.

```
import networkx as nx.
G = nx.Graph()
G.add_weighted_edges_from([(0,1,1),(0,2,1),
                           (3,0,1)])
# (from, to, weighted edge) denoting edge of weight 1
pos = nx.ego-graph(G)
ego_centre = 0
depth = 3    # friend of a friend of a friend
nx.draw (G, pos, with_labels=True, radius=
                 depth, centre =
                 ego_center)
# plots the ego centric graph for Alice.
```

```
def snowball(G, centre, max_depth,
             curr_depth, ~~visited~~ friends = [])
    if max_depth == curr_depth:
        return ~~visited~~ friends  # reached depth.
    if centre in friends:
        return friends.  # already traversed
    for node in G.neighbors(centre):
        friends.append(node)
        if node not in friends:
            snowball(G, node, max_depth,
                     curr_depth+1, friends)
            # recursive call.
    return friends
val =
snowball(G, ego_centre, 3, 0)
# function call.
print(val) # return result of data collection
           using same network G.
           Data collected is the people in the
           network
```

PART-B

1) There are 3 categories for the definition of communities.

Local Definition

→ Self referral: Focus on the vertices of subnetwork & only its neighbors.

Clique: maximally connected subgraph where each node is connected to k-1 other node.

k-clique: It is a clique in which the max. distance between nodes is always lesser than k.

k-plex: Maximally connected sub graph in which nodes are connected to all but atmost ~~two~~ k- of them.

→ Comparature: where edges between mutual connections within network is compared with external edges.

LS set: no. of neighbors inside > no. of neighbors outside community

Weakly connected graph: no. of connections inside a community > edge connecting outside.

Global definition

The community is defined wrt the overall network.

Null model: Randomly distributed. Similar structurally to original w/o bed no

properties of its own

It is created by introducing rando mness into the overall network, by reassigning edges to vertices randomly.

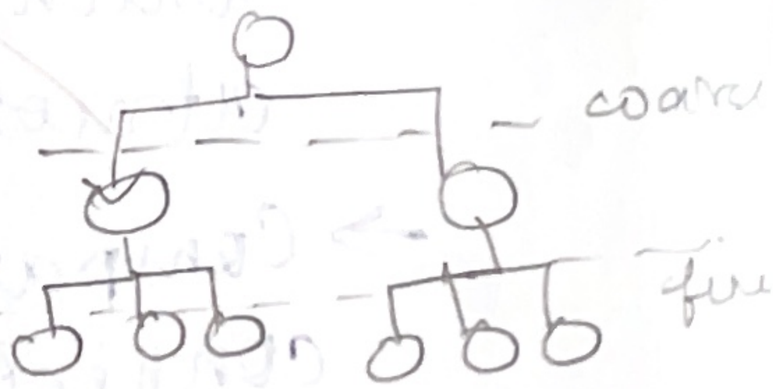
If there is a large difference when compared to original N/W, it is termed a community.

Based on Vertex Similarity

Similar vertices lie on the same level of the hierarchical dendrograph structure. By taking a horizonal line, various comm. unities can be obtained.

Granularity is fine when the line is low; ~~high~~ coarse when high.

the similarity of vertices is determined quantitatively.



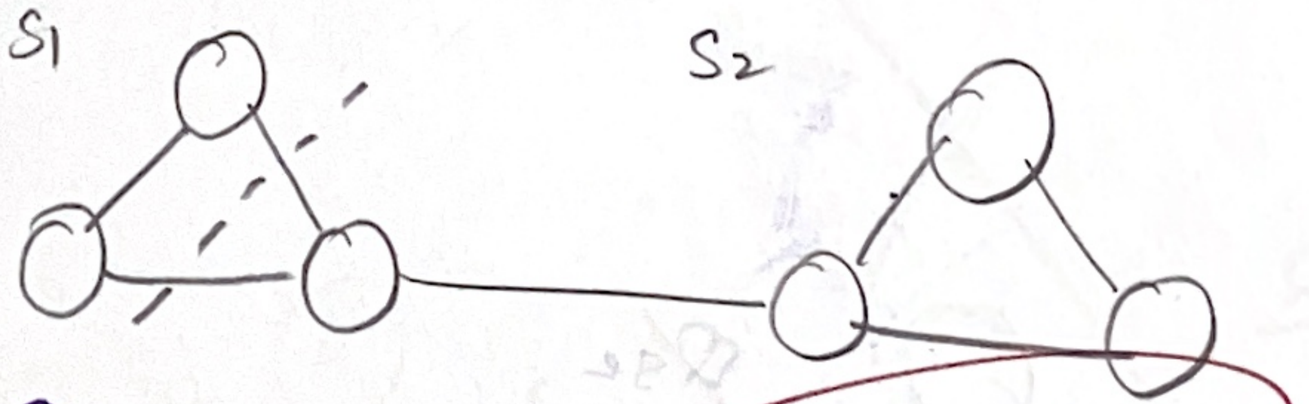
8) modularity Quality function

$$Q = \sum_{1}^{nm} \left[\frac{ls}{L} - \left(\frac{ds}{2L} \right)^2 \right]$$

where L is the total no. of edges in community.
ls no. of edges within community.
ds → degree of nodes' summation.

$Q = \sum$ (Actual fraction of edges inside graph − Expected Value of edges for graph with same vertices)

Split 1



L = 7

Q_{S_1}

ls = 1

ds = 4

Q_{S_2}

ls = 4

ds = 10.

$$Q_{S_1} = \frac{1}{7} - \left(\frac{4}{14} \right)^2$$

$$= 0.14 - 0.0816$$

$$\Rightarrow 0.061224$$

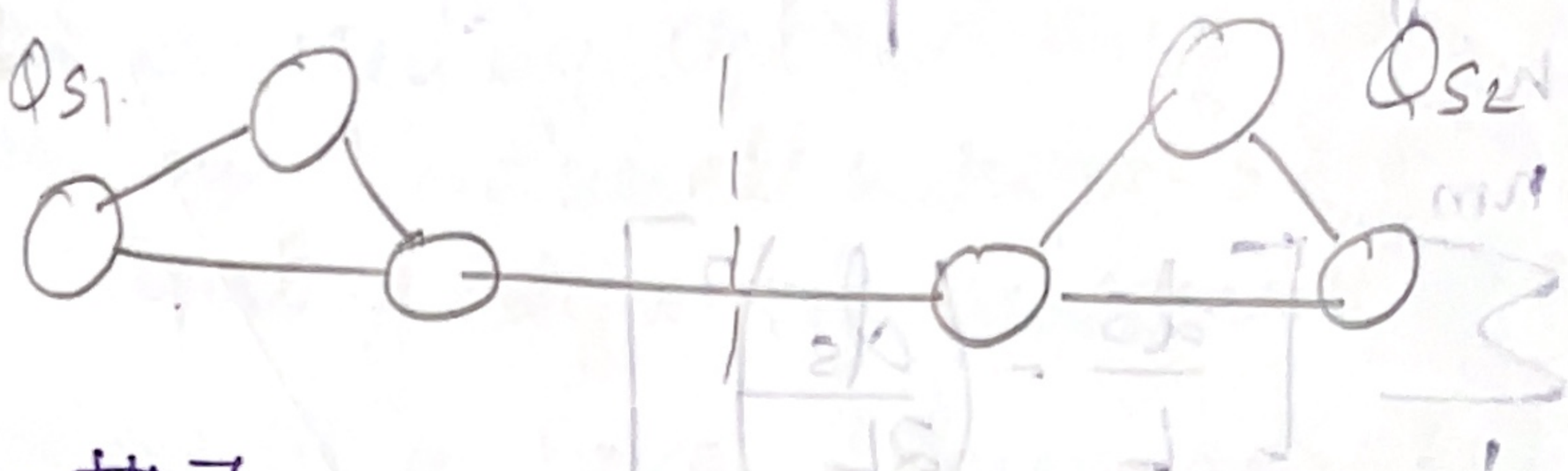
$$Q_{S_2} = \frac{4}{7} - \left(\frac{10}{14} \right)^2$$

$Q_{S_2} = 0.06122$

$= 0.5714 - (0.71)^2$

$Q = Q_{S_1} + Q_{S_2} = \cancel{1.22414}$ 0.12244

Cut 2.



L = #7.

Q_{S1}

$l_S = 3$

$d_S = 7$

$$Q_{S1} = \frac{3}{7} - \left(\frac{7}{14}\right)^2$$

$$= 0.178$$

Q_{S2}

$l_S = 3$.

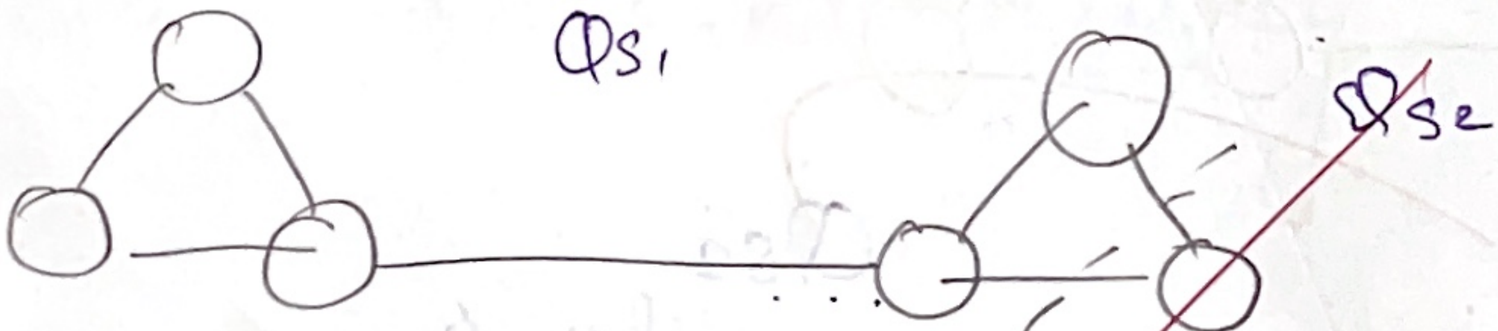
$d_S = 7$.

$$Q_{S2} = \frac{3}{7} - \left(\frac{7}{14}\right)^2$$

$$= 0.178$$

$$Q = Q_{S1} + Q_{S2} = 0.357$$

Cut 3



L = 7

$Q_{S1} \Rightarrow$

$l_S = 5$

$d_S = 12$.

$$Q_{S1} = \frac{5}{7} - \left(\frac{12}{14}\right)^2$$

$$= -0.02$$

Negative

$Q_{S2} \Rightarrow$

$l_S = 0$

$d_S = 2$.

$$Q_{S2} = \frac{0}{7} - \left(\frac{2}{14}\right)^2.$$

$$= -0.0204.$$

PART-A

1. Non-directional
   - friends on facebook
   - connections on linkedin
   - reddit friendships
   - retweeting on twitter.

2. The relations for the scenario "children playing together" are.
   → Ego-centric network for a particular child
   → friendship network for all the children to visualise their mutual friendships

3. It is important because
→ Content Filtering: similar people exist in similar networks & share taste.
Useful for recommendation systems
→ Visualisation of large scale graphs for information flow & difference
→ Understanding properties and structure of the graphs.

4. $\sum_{i=0}^{nm}$ (Fraction of the actual no. of edges within community − Expected number of edges for an equivalent null model graph)

$nm \rightarrow$ no. of communities.

Originally

$$\sum_{i=0}^{nm} \frac{ls}{L} - \left(\frac{ds}{2L}\right)^2$$

$L \rightarrow$ total no. of edges.

$ls \rightarrow$ links within N/W comms

$ds \rightarrow$ degree summation:

Optimisation

5) → converts the NCMP into an optimisation problem; optimise the objective function.

→ Obj function: criteria, evaluation fn, etc

→ Further types: local search based & spectral search.

→ Eg: Faster Newman, KL algorithm

Heuristic

→ Does NCMP as a heuristic based problem taking some assumption/heuristic measure.

→ Eg: MFC considers that flow between communities should be greater.

→ Eg: WH Algorithm, Girvan-Neuman Algorithm.

6) import networkx.

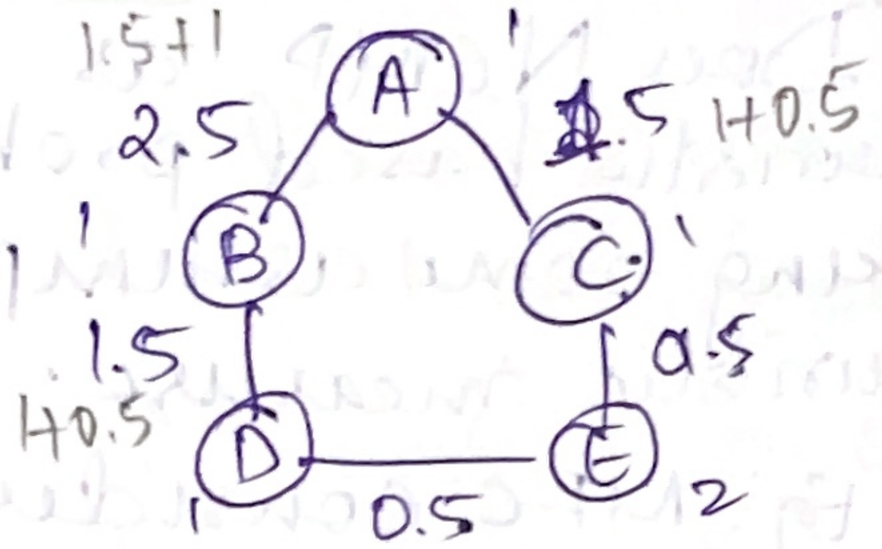
```
# initialise
New_G = networkx.Graph()
```

~~for from, to, edata in G.edges():~~

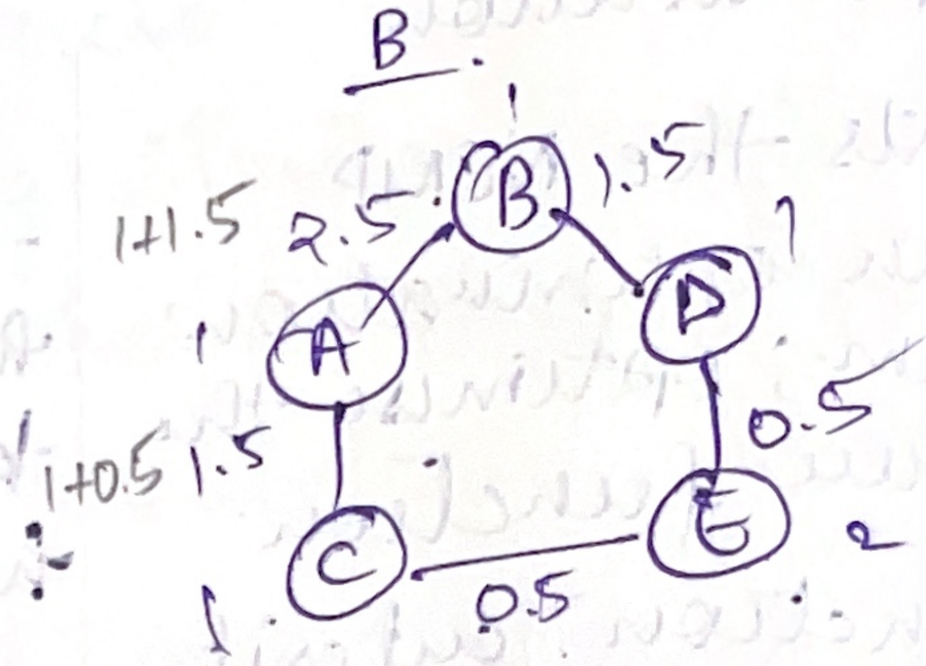
```
for node in G.node():
    if G.degree(node) > 2: cont
        continue # do not add.
    new_G.add_node(node) #add in new graph
```

9)

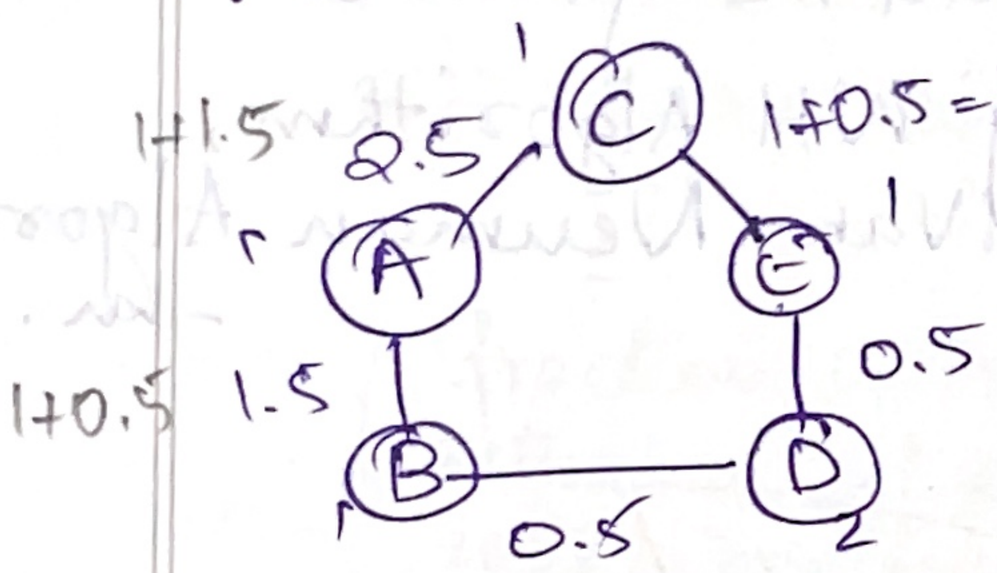
A.



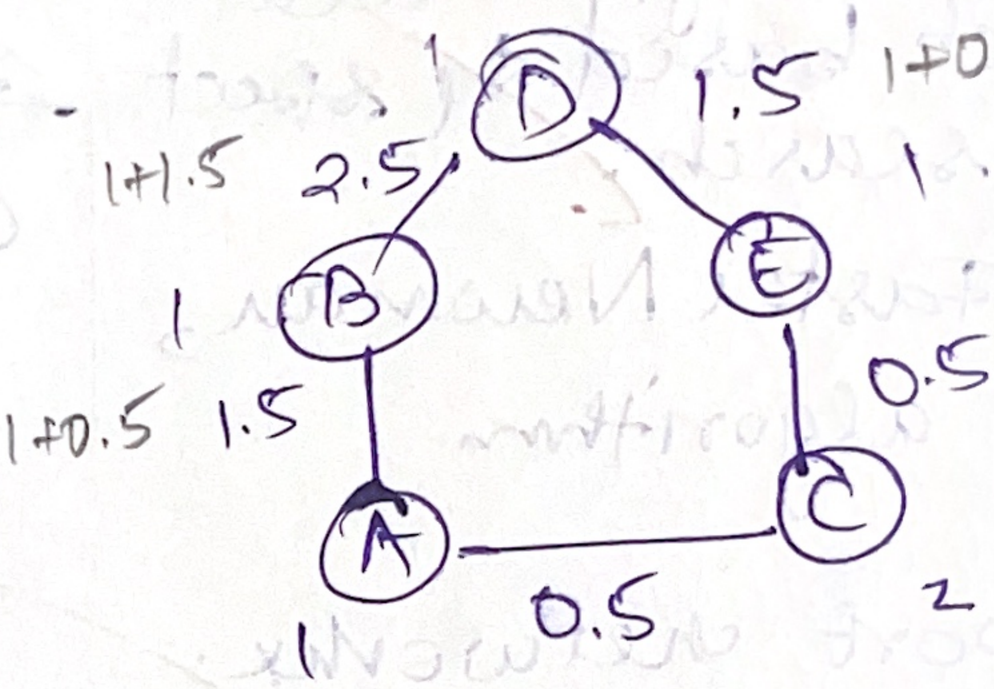
B.



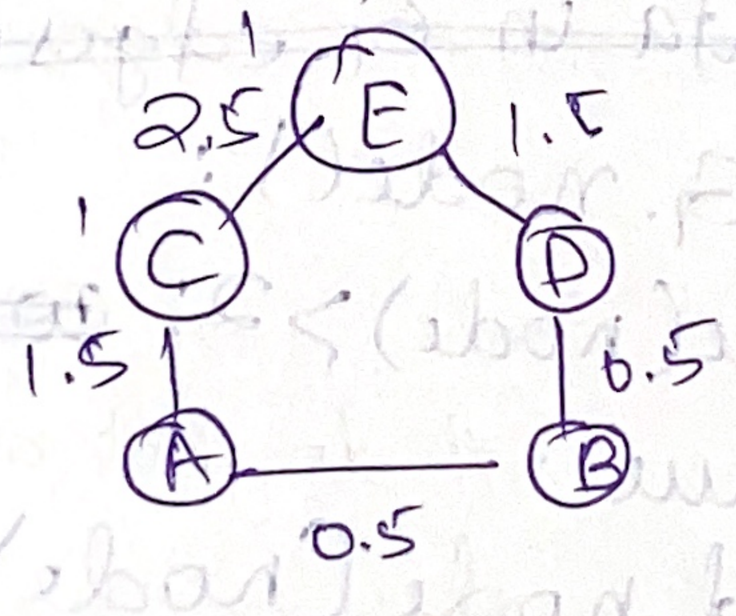
C



D



E



AB = 2.5 + 2.5 + 1.5 + 1.5 + 0.5 = 8.5

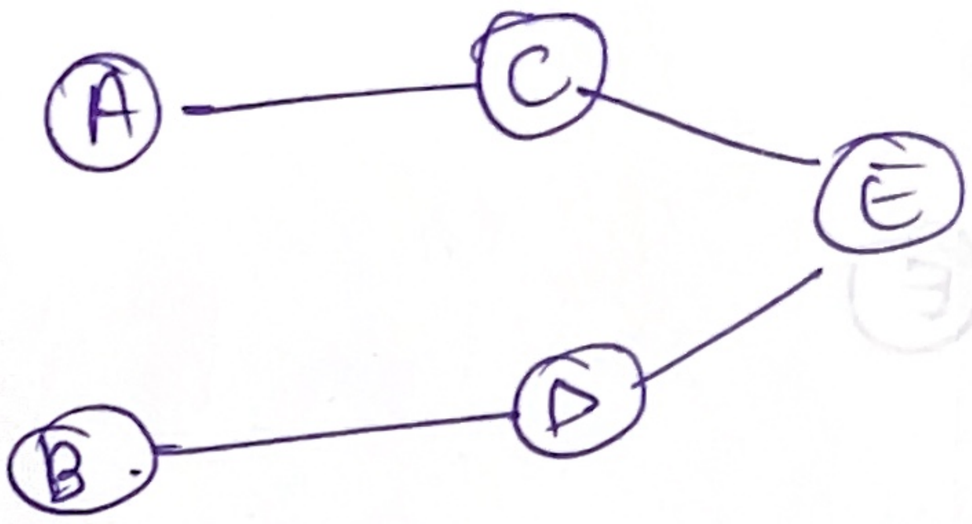
AC = 1.5 + 1.5 + 2.5 + 0.5 + 1.5 = 7.5

BD = 1.5 + 1.5 + 0.5 + 2.5 + 0.5 = 6.5

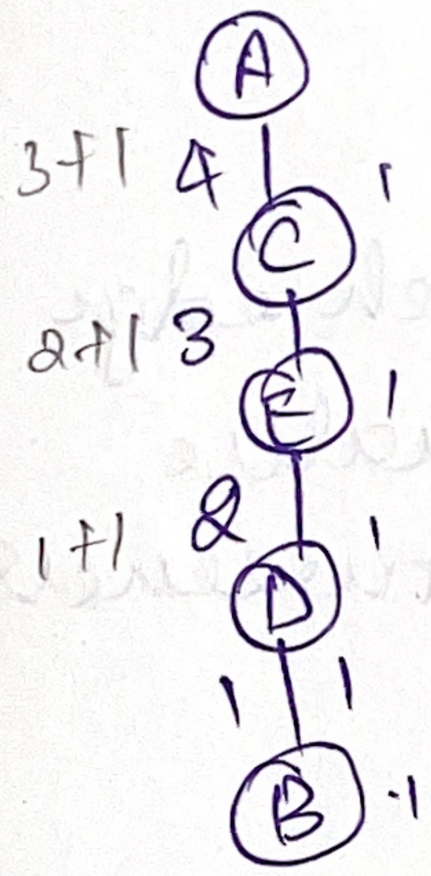
CE = 0.5 + 0.5 + 1.5 + 0.5 + 2.5 = 5.5

DE = 0.5 + 0.5 + 0.5 + 1.5 + 1.5 = 4.5

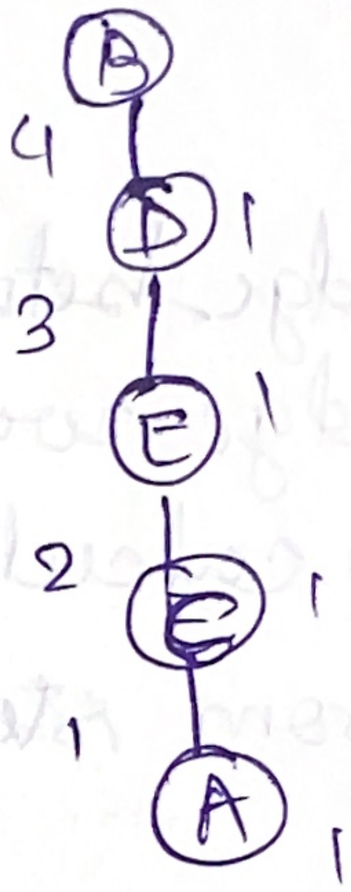
Remove AB.



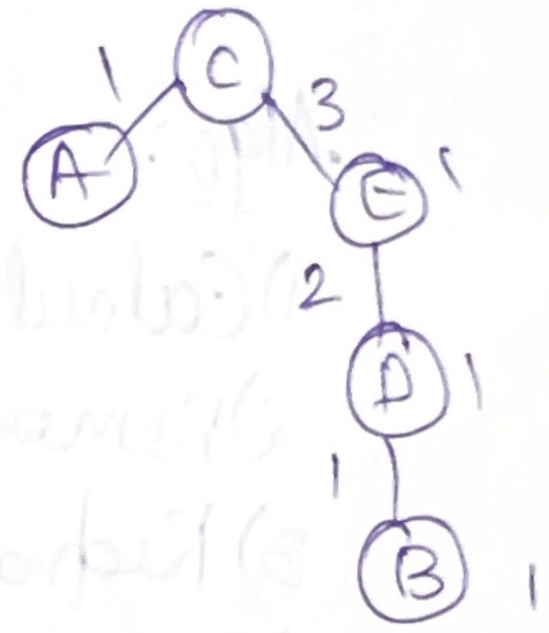
A



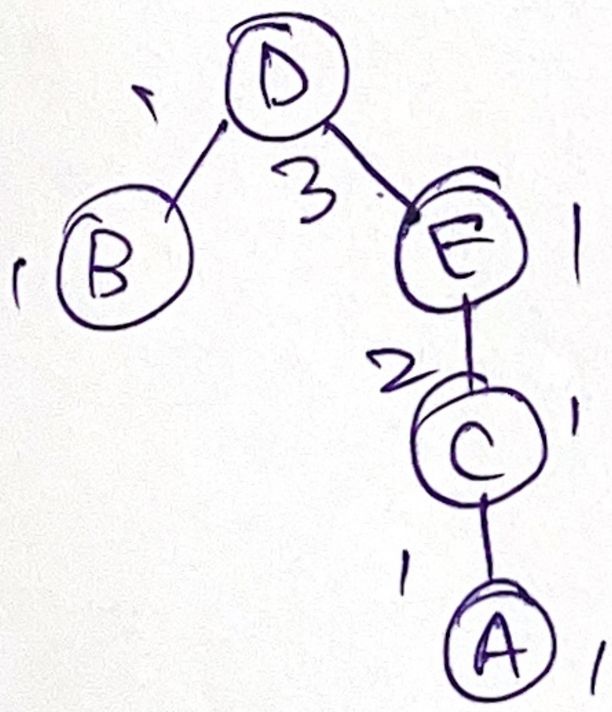
B.



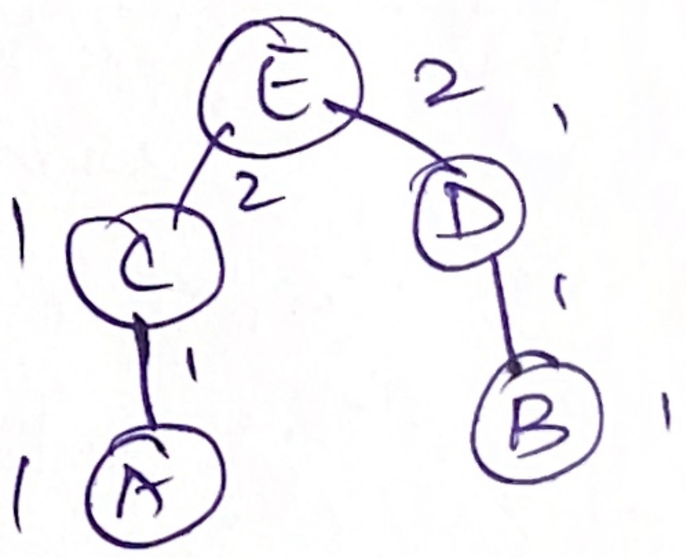
C



D.



E



AC = 4+1+1+1+1 = 8

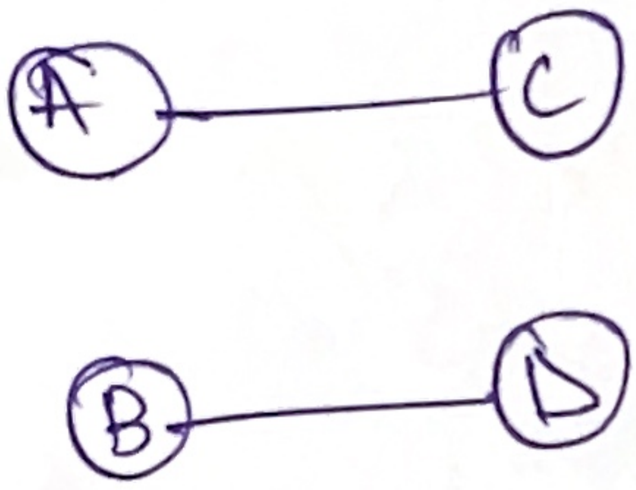
CE = 3+2+3+2+2 = 12.

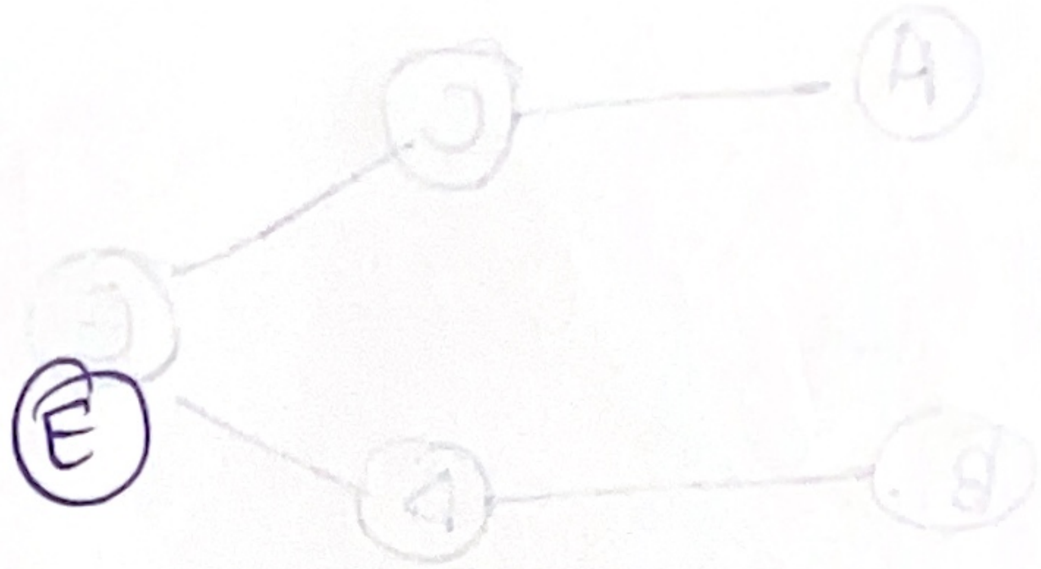
ED = 2+3+2+3+2 = 12

DB = 1+4+1+1+1 = 8.

Cut CE & ED.

∴ max. edge centrality





final graph.

Algo:

1) Calculate edge betweeness for each edge
2) Remove edge with highest value
3) Redraw & calculate edge betweeness
4) Iterate from step 2.