2-1 We can reduce this question to a max-flow Problem, to answer it effectively.

Consider, the comedians, boats to be nodes C_i $[i=1,2,\ldots k]$ and non-comedians to be nods P_i $[i=1,2,\ldots 2K]$ in a directed graph

Now draw, directed edge from a non comedian node to all the comedian nodes, whose jokes they can tolerate; and assign each edge with a capacity 1.

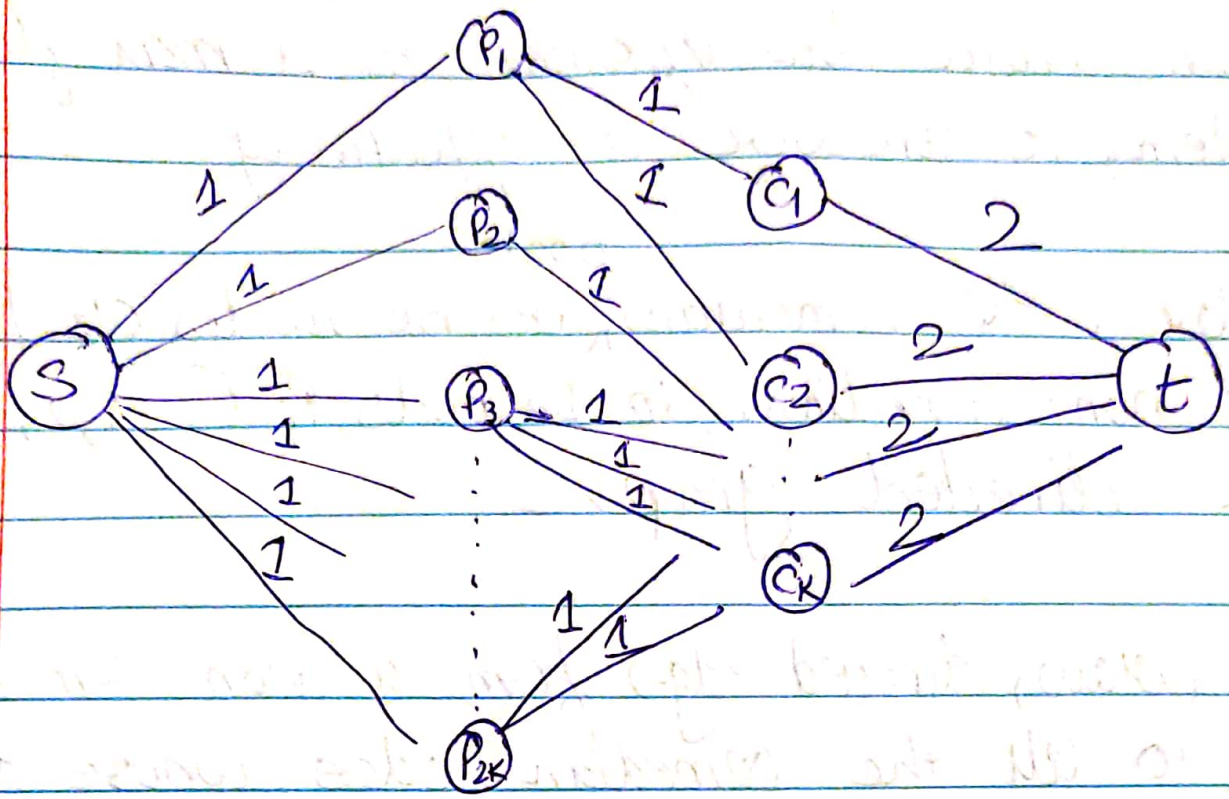
Consider, the Source node - s & Sink node - t

Draw edges of capacity 1, from the source nodes to non-comedian nodes. These show the number of non-comedians on the ship.

Draw edges of capacity 2, from each comedian nodes to the sink node. These, show the people assigned to the rescue boats.

The following is the depiction of such an s-t flow

Finding, max-flow should give us a way to evacuate everyone on the ship



Q-2 Given a graph G_1, with node capacities, we define H as follows:-

- For every node v_i, $i \in [1, \ldots K]$ in G_1, introduce two nodes v_i^{in} & v_i^{out} in H.

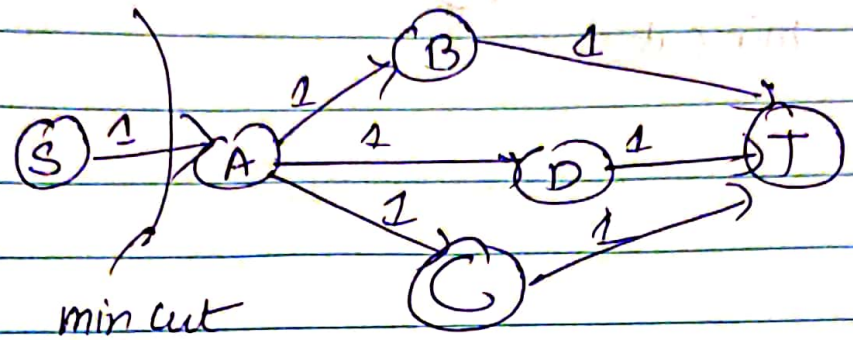
- Now, for every $(v_i, v_j) \in G_1$, we add an edge (v_j^{out}, v_i^{in}) in H, with infinite capacity; and we add an edge (v_i^{in}, v_i^{out}) with capacity c_v.

Thus, we see a 1-1 correspondance between G_1 & H, and we can use Ford-Fulkerson algorithm on H, to find an s-t flow.

Thus, we can use Ford Fulkerson can be used on G_1 as well

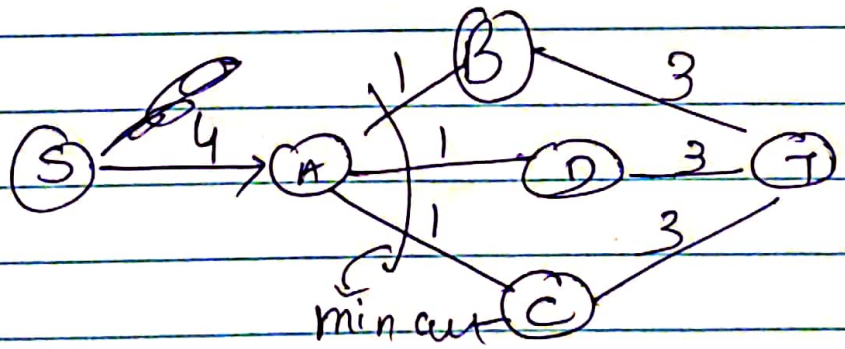
Q.3 There are 2 cases to consider here

Case ①: Edges have same ~~weight~~ capacity

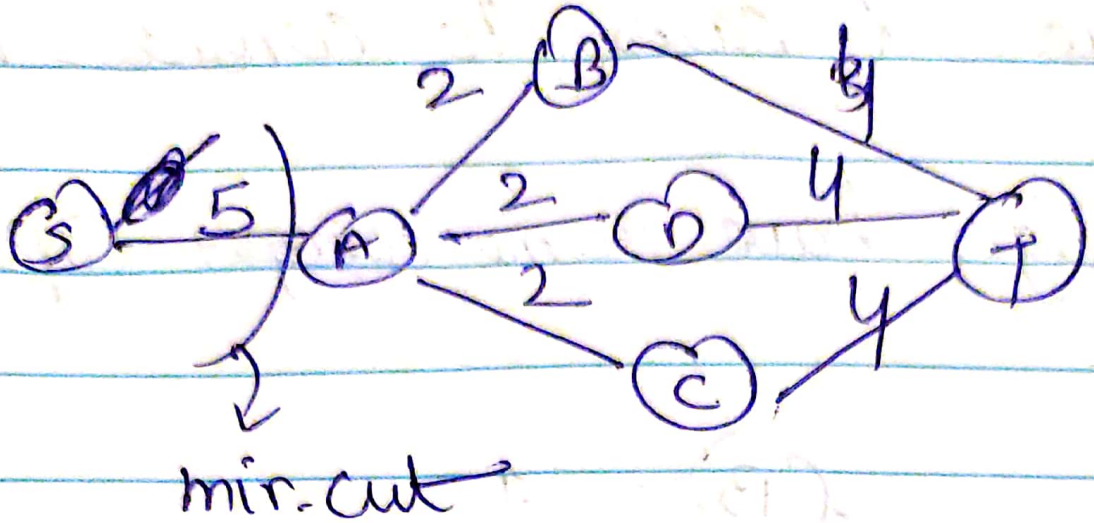


If capacity is increased by 1, ~~min~~ min cut will have increased capacity of 2; ~~which~~ ~~will~~ ~~is the same~~ thus; remaining the same

Case ② Edges have different capacity

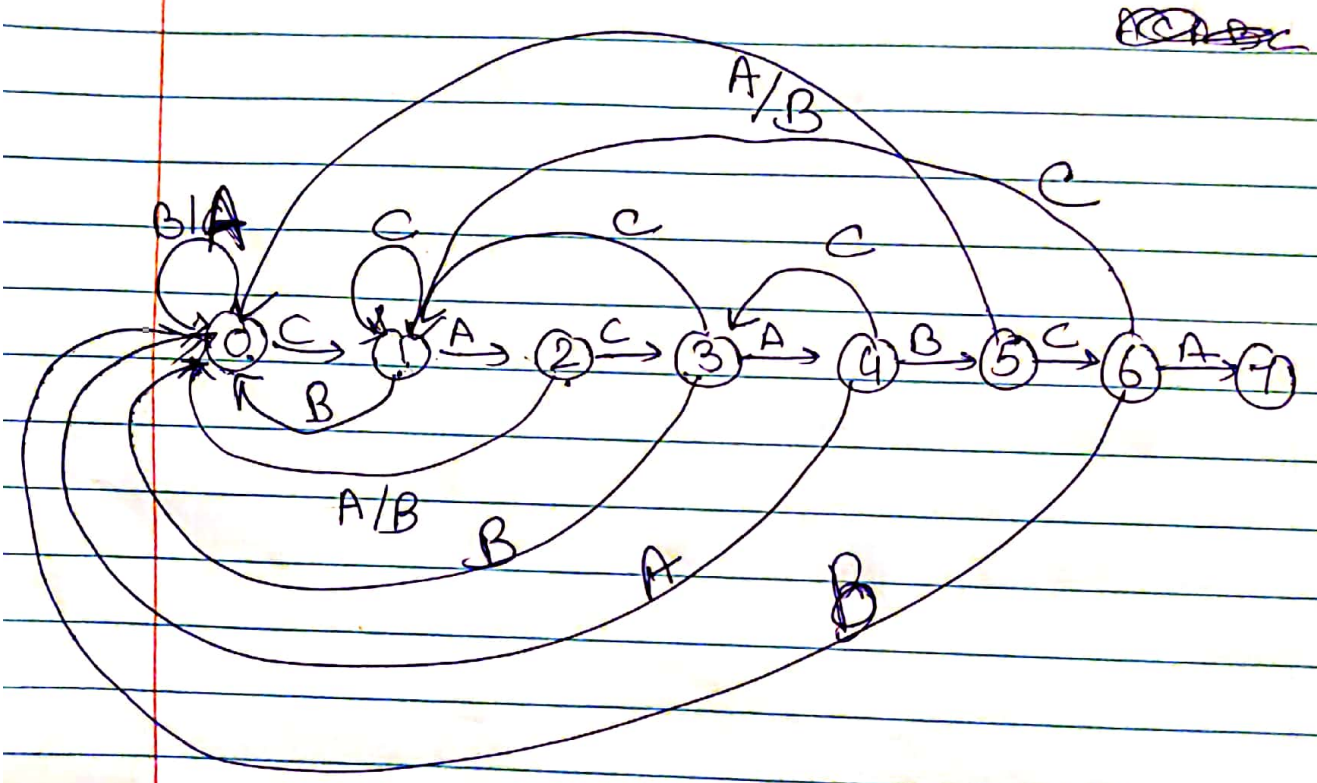


In this case, min cut has value 3. But if capacity of each edge is increased by one, the min-cut changes to 5 as shown.



Q-4 CACABCA

	0	1	2	3	4	5	6	
A	0	2	0	4	0	0	(7)	← Final state
B	0	0	0	0	5	0	0	
C	1	1	3	1	3	6	1	



Q-5 Rabin-Karp algorithm Searches in a given String and returns the index where the pattern found. It uses hashing to Search the pattern in the String.

Now, consider the pattern with indexes [0,1,2... m-1] that has index $j \in \{0, 1, \ldots m-1\}$

- Take the new pattern → pattern [1,.. j-1] + Pattern[j+1,.. m] from the given pattern by skipping the j^{th} ~~Pattern~~ character and find the hash of the new pattern.

- Also, for each loop iteration, take the Substring denoted as ⇒ S[i,... i+j-1] + S[i+j+1,... i+m] from the substring by skipping the j^{th} character and calculate the hash of the new substring

- Compare both hash values, if they are same return index i, else continue with next iteration of i