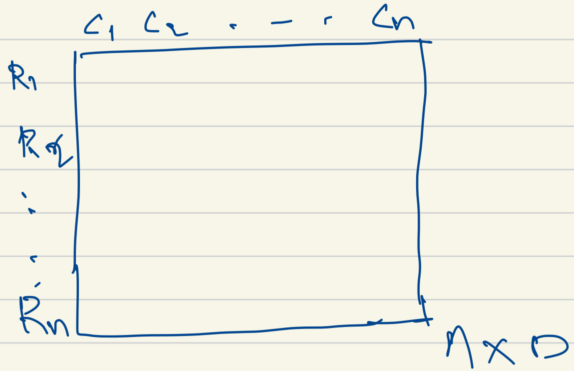
PS 4.

Q.1

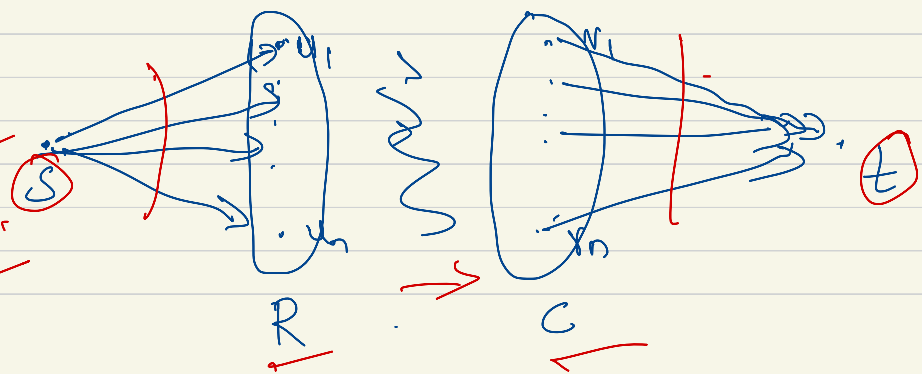


Reduce to max-flow min-cut.

$cap((s, u_i)) = R_i$

$cap((v_j, t)) = C_j$

$cap((u_i, v_j)) = 1$.



Algorithm

① Construct the flow network

② Run FF algorithm to compute the max flow
(can assume w.l.o.g that flow on every edge is integral)

③ $\forall i, j$, $\hat{M}_{ij}$ = flow value on (u_i, v_j).

Correctness

Need to show:

① If there is a feasible matrix, then the algorithm outputs it correctly.

② If no feasible matrix, ~~then~~ algorithm fails also.

Proof of I $M \Rightarrow$ feasible matrix.

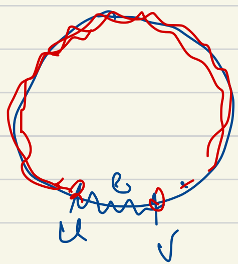
- Set $f((u_i, v_j)) = M_{ij}$
- other edges, set flow to capacity.

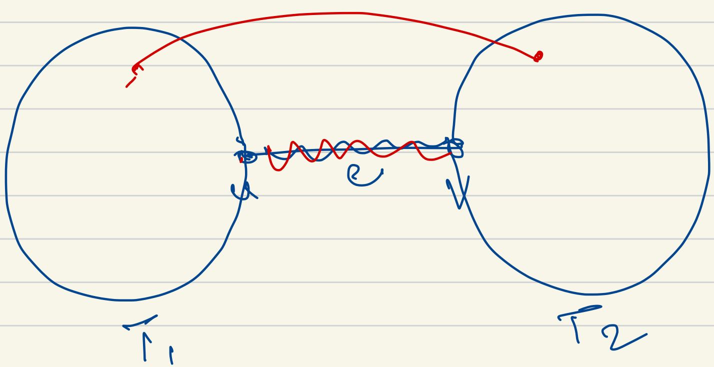
Feasibility $\rightarrow$ defn. of row sums, col sums and the flow network construction.

Max flow $\rightarrow$

Q. 3 T – MST containing an edge e

C – cycle in the graph and e = max weight edge in the cycle.





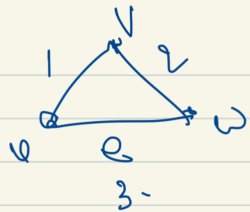
Q. 4.

Suggestion:

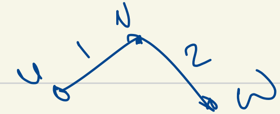
1. Run Kruskal on G. $\rightarrow$ outputs T_1
2. Run Kruskal on G
   - starting with the edge e included.

   $\rightarrow$ outputs T_2.
3. $wt(T_2) > wt(T_1)$ output no MST containing e

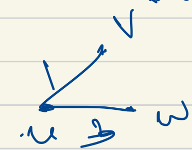
   else output Yes.



T_1 —



T_2 —



~~Q. 6.~~

Suggestion:

(1) Find a min s-t cut using Ford-Fulkerson
say C, $v_0 := \text{cap}(C)$

(2) Iterate over all $e \in C$, do the following.

- increase cap(e) by 1
- Leave the remaining edges untouched
- Compute min-cut again /max-flow
- say v_e is the capacity.

③ If $\forall e \in G$, $v_e > v_0$, output min-cut unique.

else

output min cut not unique.

Correctness:

① G, has a unique min cut.

$C \to$ unique min cut.

$\forall \tilde{C}$ - s.t. $cap(\tilde{C}) > cap(C)$

$\}$

② Non-unique min cut

$\exists\, e \in C$ s.t increasing the val. of e does not increase the cap of the min cut.

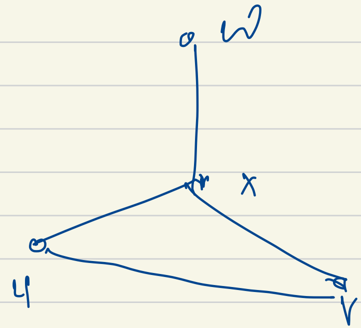
- C, C_1 - be two min cuts, $C \neq C_1$

$\Rightarrow \exists e \in C$ s.t $e \notin C_1$.

$c_e \leftarrow$ increase the cap of e by 1.

$v_e = v_0$.

0.5



for now: want simple paths (no repeated edge)

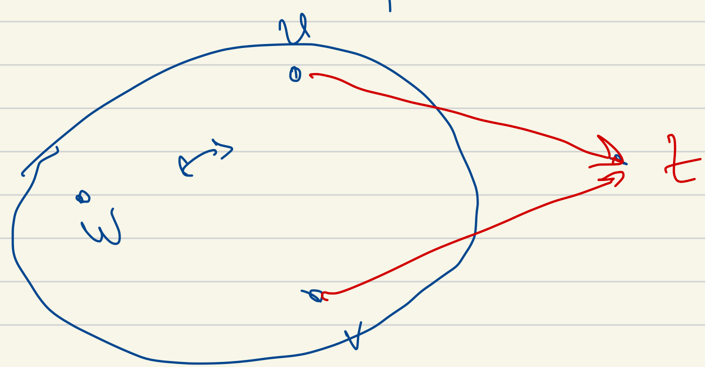
Umang Bhaskar (TIFR)

Setting up a flow network:

u v

w

1. make all edges bidirectional.
2. capacity 1 in each direction
3. Add a sink t, add edges (v, t), (u, t) capacity 1.



4. Set w → source
5. compute max flow between w and t.

Q: Flow network, cap 1.
Do there exist k-vertex disjoint paths from s → t in G.

Soln: Reduce vertex disjoint version to the edge disjoint version:

G, s, t, u → $\tilde{G}, \tilde{s}, \tilde{t}, \tilde{u}$