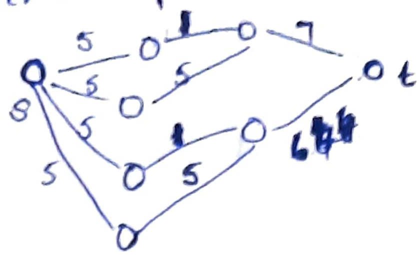
~~AAD~~

Problem - set 4

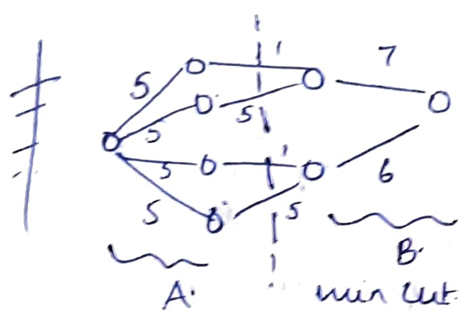
Name - Malla Sailesh
Roll number - 2021101106

(1)

counter example -

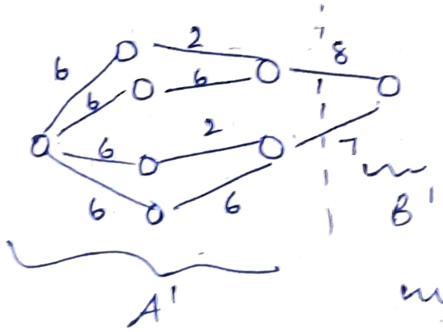


min cut (A, B)



min cut = 1 + 5 + 1 + 5 = 12

Now increase 1 in every edge



6 + 6 + 2 + 2 = 16

min cut = (A' B') = 8 + 7 = 15

Thus disproved the given statement.

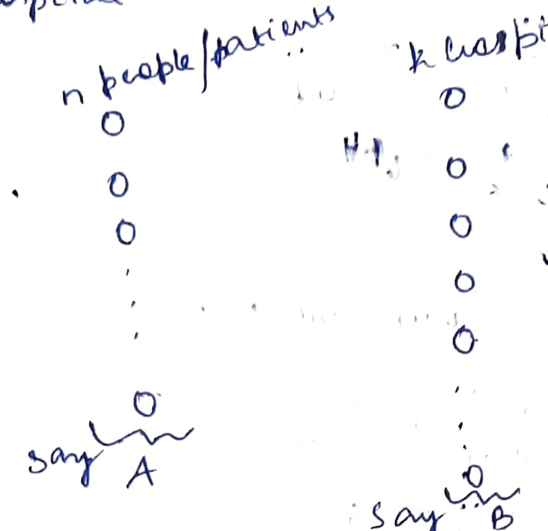
Thus min s-t cut (A, B) may vary depend on the weight of graph.

see, here all the weight of edges ~~weight~~ in the network are increased by same amount. ~~But At~~ But still the number of edges joining A, B for all A, B may vary.

Thus the min s-t cut (A, B) ~~is still a~~ may / may not be a min s-t cut. Thus disproved.

(2)

This problem is similar to maximum matching because in total n people must go to all hospitals where ~~Only~~ atmax $\lceil n/k \rceil$ people or patients need to go 1 hospital.



we ~~need to~~ will draw an edge between person and hospital iff that person can go to that hospital in less than equal 30 min (half hour). with capacity 1 as only (1 person) goes.

→ Now we consider ~~sink~~ source s, sink t

→ Now connect ~~all~~ edges from s to all nodes in A with capacity 1

→ And connect ~~all~~ edges from all nodes in B to sink t with capacity $\lceil n/k \rceil$

because atmax $\lceil n/k \rceil$ persons go to a hospital.

Now in final graph find the maximum flow.

If maximum flow = = n (i.e. n persons can go to hospital satisfying the given condition)

then it is possible

to take all patients to hospital.

else it is not possible.

proof:
key point: all hospital nodes connect to sink t through edges with max capacity $\lceil n/k \rceil$.

so because of this if more than $\lceil n/k \rceil$ goes to a particular hospital then it doesnt go to sink t.

Thus this satisfies the condition given in the question.

so as our goal is to find max flow which if equal to n. Then satisfies.

(1) This checks if all them can reach hospital in less than 30 min. (by check the number of edges b/w A, B) → should be atleast n.

(2) sum of no. of people went to hospitals. (by checking the number of edges between B and sink t) ↳ should be n.

Thus atleast n flow must be there. min cut = max flow.

Thus max flow gives the correct answer.

Note: max flow is done is polynomial time, construction of graph also done in polynomial time, so this is ↳ so satisfy

(3) let the modified graph G_1.

~~we~~ It's clear that the max flow in G_1 is (max flow in G) or (max flow in G − 1)

let max flow in G is F.

so max flow in G can be F, F − 1

we know, ~~say~~ if max flow = a.

proof in G: max flow = min cut = F (in G)

⇒ min cut = F. (in G)

so ∃ (A, B) such that min S − t path is F.

Now if some other edge has reduced 1 unit capacity.

If there is only pair (A, B) then decreasing edge by 1 still have cut greater than equal to F.

If corresponding (A, B) edge has been reduced to then flow is F − 1

Thus flow of G_1 can be F or F − 1 whereas flow of G = F (given).

—

Now suppose consider the flow of G_1 is F_1 (say).

To obtain G from G_1 we must add 1 to an edge that we reduced initially to get G_1 from G.

to G.

Thus in the final residual graph of G if the edge we want to reduce the capacity has 0 flow in direction of edge then it implies the edge is completely used in flow so if reduce the capacity of such an edge the value (max flow) is F − 1.

If edge(u,v) capacity in the final stage is > 0. ~~then~~ ~~reducing~~ in residual graph when u → v ∈ E. Then reducing 1 length will not effect the flow

⟹ value (max flow) = F

~~(or)~~

(or)

~~4)~~ we decreas the value of edge by 1. So now lets find an augmenting path. If not found ok. if found then the value of flow changes i.e F − 1

because if value of edge is positive then it clear the decreasing it by 1 dont anyway contribute to flow ⟹ F.

If value of edge in residual graph is 0 then the edge is (x) completely used in ~~by~~ the flow. so now decreasing it to −1 and now find an augment path.

⟹ F − 1 (is the value of max flow).

Augment path finding is O(m+n)

proved

(4) we know maximum flow = minimum cut.

Here the agenda is to decrease the maximum i.e decrease the min cut.

As all the edges have capacity of 1

the min cut has (A, B) = l (say)

iff there are l edge between A, B.

min cut = min(Σ edges from A to B)

each edge has capacity 1

$\Rightarrow$ min (edges from A to B)

to decrease the maximum flow we need to decrease the minimum cut. i.e. decrease edges from A to B

first we need to find the A, B such that it gives min-cut

(done in polynomial time.

now if no. of edges in b/w A, B is less than K, ~~max f~~ min cut = max flow = 0

else mincut = max flow = l - k.

here we will remove edges which contribute min cut).

This works because we want to decrease max flow

$\Rightarrow$ we want to decrease min cut (A, B)

$\Rightarrow$ i.e. we want to decrease edges because capacity of each edge is 1

Thus shown.

Remaining all cuts have value greater than min. So to minimise more i.e. to decrease the min value is better choice.

$\not\Rightarrow$

eg. $C = \min(a, b)$. $a < b$.
we can decrease any value by k.
decreasing a by k decreasing C

similarly.

(5) We will find the min-cut of G.
and say the value is equal to (F).
→ let say (A, B) gives the min-cut.
Now ~~if~~ we will start increasing
to ~~every~~ edge ~~and~~ ~~check~~ between A
and B and check if for any
increase in edge if we still get
min-cut as (F) in any increase
of edge capacity Then not unique
otherwise unique.

proof:- This works because if there
are 2 min cut (say) in Graph G.
let (A_1, B_1), (A_2, B_2) give the
min-cut. F. Now we are increasing
the value of an edge in b/w A_1 B_1.

If that edge is not in (A_2 B_2)
then (A_2 B_2) gives the min-cut.
~~say that edge is there.~~
else value increase by 1.

In this case ex. consider another
edge independently in b/w (A_1 B_1) and
increase its capacity by 1.
If that edge is not in (A_2 B_2)
then (A_2 B_2) gives the min-cut
else value increase by 1.

lly we again consider an edge
in b/w (A_1 B_1). so on we go
until that edge not in (A_2 B_2) or
all edges b/w A_1 B_1 are over

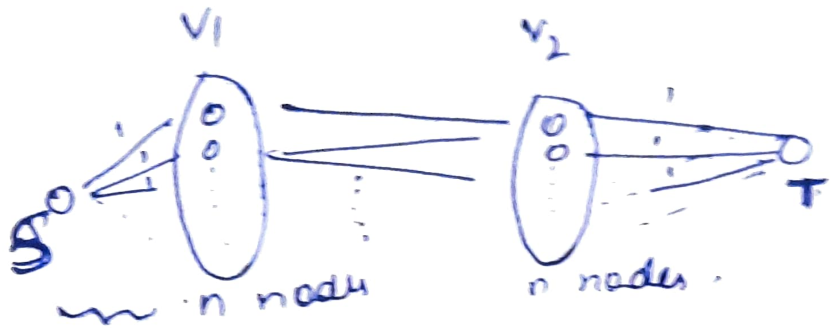
In this case for value
of min-cut ~~both~~ of (A_1 B_1) & (A_2 B_2)
to be equal there should
be no more edges b/w (A_2 B_2)
different from (A_1 B_1).

In that case (A_1, B_1) (A_2 B_2)
same.

Thus (unique min s-t cut),
If any ↑ in edge capacity of edge b/w
min-cut found wont ↑ the min-cut
value then no unique min s-t cut
otherwise unique min s-t cut.

Thus proved.

(b)



In bipartite matching only edges b/w nodes in V_1 to nodes in V_2.

~~cut~~ ~~b/w~~

cut b/w S and V_1 = n (clearly)

cut b/w V_1 and V_2 atleast $|E|$ ← because 1 added to all edges in E.

cut b/w V_2 and T = n (clearly)

min cut = max flow ~~= min~~ (1r

case-1 If min-cut < n. then obviously there are less than n edges in b/w V_1 V_2. which implies no perfect matching.

If min-cut == n. then perfect matching else. non perfect matching

proof :- . case 2: shown that min-cut < n. its not perfect matching.
min- cut > n. is not possible.
because. there is a min-cut between. (S) (V1) (or) (V2) (T) which. is equal to n. so min-cut <= n.

we shown. that min-cut < n. is not perfect matching.

Thus now we. should only show min cut == n gives perfect matching.

=> outgoing flow from s => n.
=> possible iff the flow = 1 in all edges from. s to V_1
so. ~~factor~~ incoming flow to all edges. in V_1 is 1 . Thus.
by conservation of flow outgoing flow from all ~~edo~~ nodes in V_1 is 1
— (1)

=> incoming flow from . T = n.
= possible iff the flow = 1 in all edges from V_2 to T. so outgoing flow to all edges in V_2 is 1. Thus
by conservation of flow incoming flow to all nodes in V_2 is 1.
— (2).

from (1) (2) it is obvious that there exist a perfect matching.

Thus proved