PAGE No.
DATE / /

AoA week 10 Notes

Applications of Max Flow Algo.

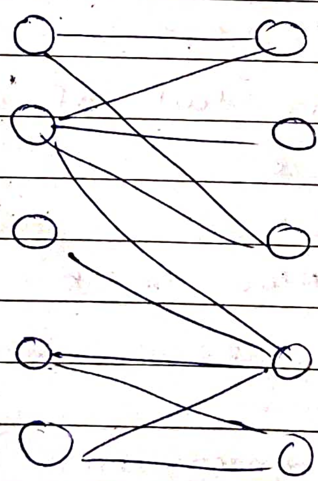
Network Flow

* Bipartite Matching problem.

A bipartite graph, $G=(V,E)$ is an undirected graph whose node set can be partitioned as $V = X \cup Y$ with property that every edge $e \in E$ has one edge in X and other in Y.

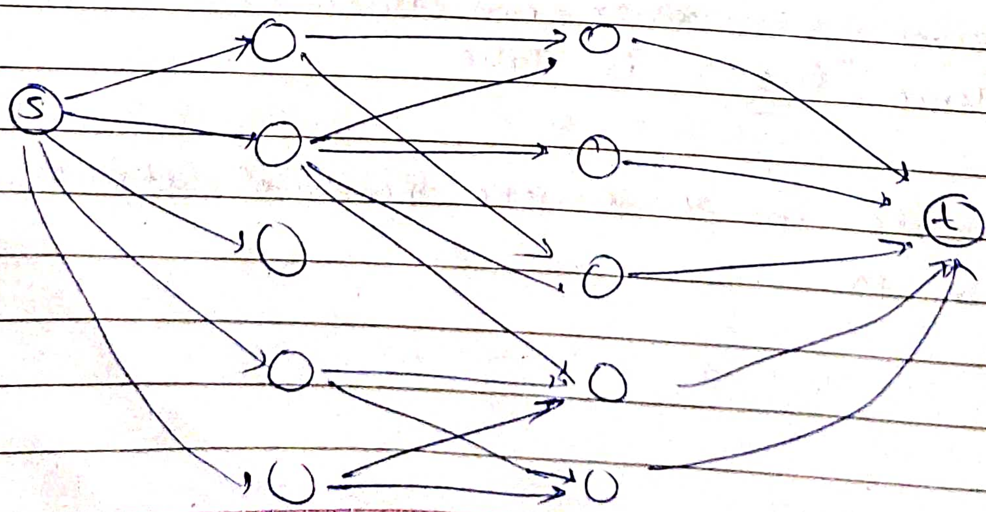
A matching M in G is a subset of the edges $M \subseteq E$ such that each node appears in atmost one edge in M.

G



Given graph G find matching M of largest possible size.

G'



with all capacities equal to 1.

Now Find Max-Flow in G'

suppose there is a matching in G consisting of k edges $(x_{i1}, y_{i1}) \ldots (x_{ik}, y_{ik})$.
Flow across each edge is 1

$\therefore$ By satisfying capacity and conservation condition

Flow = K

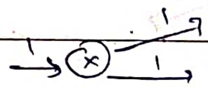
Conversely if there is a flow f' in G' of value K, there will be K edges.

(7.34) → M' contains K edges

(7.35) Each node in X is the tail of atmost one edge in M'

<u>Proof:</u> suppose x is tail of two nodes

But incoming flow to $x = 1$



Not satisfying conservation condition.

Hence assumption not true

Hence (7.35) is True.

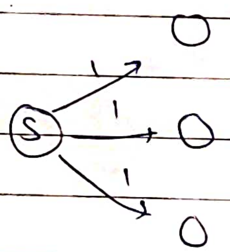
(7.36) Each node in Y is the head of atmost one edge in M'

(7.37) The size of maximum matching in G is equal to the value of the max Flow in G', and the edges in such a matching in G are the edges that carry Flow From X to Y in G'

(7.38) The Ford Fulkerson Algo can be used to find maximum matching in a bipartite graph in O(mn) time

PROOF: $O(C \cdot m)$

$$C = \sum_{e \text{ out of } S} c_e$$



$\therefore C = 1 * n$

no. of nodes.

$O(n \cdot m)$

Augmenting paths are also called alternating paths in the context of finding a maximum matching.

What if there is no perfect matching?

- Till now we will find matching using max Flow and then find if matching is perfect or not

- We can also see to find a cut of capacity less than n.

If capacity is less than 'n' then Flow < 'n'

$\therefore$ No perfect matching

PAGE NO.
DATE / /

Consider a subset of nodes $A \subseteq X$
$\Gamma(A) \subseteq Y$ → denote all nodes that are adjacent to nodes in A.

(7.39) If a bipartite graph $G = (V, E)$ with two sides X and Y has a perfect matching, then for all $A \subseteq X$ we must have $|\Gamma(A)| \geq |A|$.

(7.40) Assume that the bipartite graph $G = (V, E)$ has two sides X and Y such that $|X| = |Y|$. Then the graph G either has a perfect matching or there is a subset $A \subseteq X$ such that $|\Gamma(A)| < |A|$. A perfect matching or an appropriate subset A can be found in $O(mn)$ time.

* Edge Disjoint Path problem. (Directed Graphs)

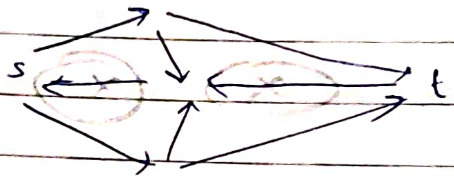
We say that a set of paths is edge-disjoint if their edge sets are disjoint, that is no two paths share an edge, though multiple paths may go through some of the same nodes.

problem: Given a directed graph G with $s, t \in V$, find max number of edge disjoint s-t paths in G.

Plan: Design a Flow network G' that will have a flow v(f) = K if there are K disjoint s-t paths in G.

Moreover Flow in G' should identify the set of edge-disjoint paths in G.

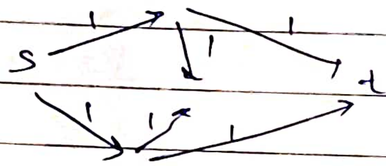
G



No incoming edge on source

No outgoing edge from Sink

G'



- Run max Flow on G'
- v(f) will equal max number of edge-disjoint s-t paths.
- f will identify edges on path.

(7.41) If there are K edge disjoint paths in a directed graph G from s to t, then value of maximum s-t Flow in G is atleast K.

(7.42) If F is a 0-1 valued Flow of value v, then the set of edges with Flow value f(e) = 1 contains a set of v edge-disjoint paths.

(7.43) There are K edge-disjoint paths in a directed graph G from s to t if and only if the value of the maximum value of an s-t Flow in G is atleast K.

(7.44) Ford Fulkerson Algo can be used to Find a maximum set of edge-disjoint s-t paths in a directed graph G in O(m·n) time.

(7.45) In every directed graph with nodes s and t, the maximum number of edge-disjoint s-t paths is equal to the minimum number of edges whose removal seperates s from t.

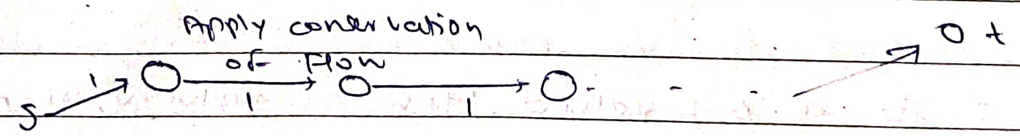
To prove: converting G to G' and Finding max Flow in it gives edge disjoint path in G.

We will show that there are K edge disjoint paths in G if there is a Flow of value K in G'

Proof:

(A) If we have K edge disjoint s-t path in G, we can Find a Flow of value k in G'

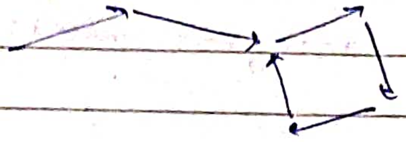
(B) If we have a Flow of value K in G', we can Find K edge point s-t path in G.



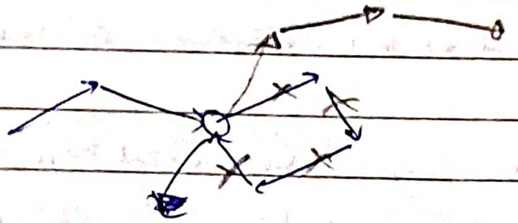
After reaching we have Flow 1

No same exploring is done For other K-1 paths.

problem in this kind of exploring



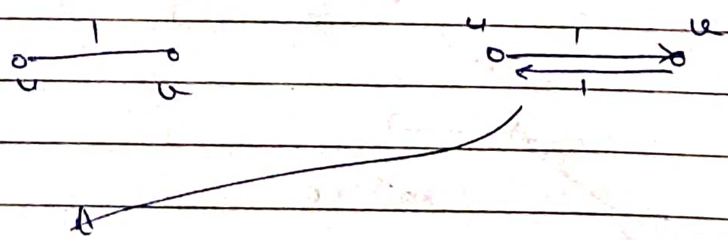
while expanding we land on a particular node again



For this node
incoming edge = 2
outgoing = 1
∴ it must have 1 more outgoing edge to
Conserve flow

So we will remove the cycle and explore the other edge.

(*) Disjoint Path problems (undirected graphs)



But this may lead to problem
as (u, v) and (v, u) are seperate edges in directed graphs and can be used in two diff paths, but in undirected graph it is a single edge

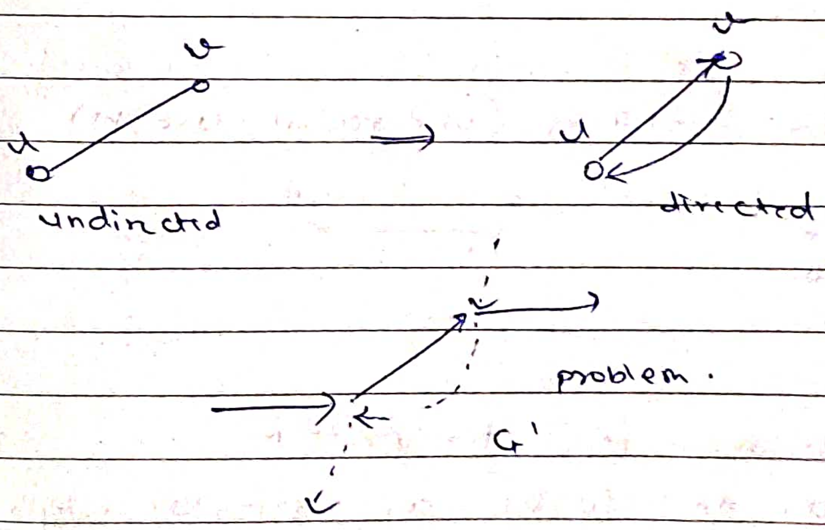
Hence we have to make sure only one of this edge is used.

(7.46) In any flow network, there is a maximum flow f where for all opposite directed edges $e = (v, u)$ and $e' = (u, u)$, either $f(e) = 0$ or $f(e') = 0$. If the capacities of the flow network are integral, then there also is such an integral maximum flow.

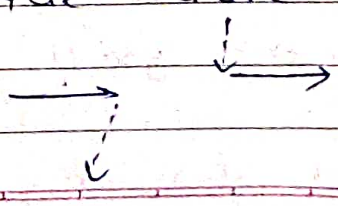
(7.47) There are k edge-disjoint paths in an undirected graph G from S to T if and only if the maximum value of s-t flow in directed version G' of G is atleast k.

Further, Ford Fulkerson can be used to find a maximum set of disjoint s-t paths in an undirected graph G in $O(m \cdot n)$ time.

(7.48) In every undirected graph with nodes s and t, the maximum number of edge-disjoint s-t paths is equal to the minimum number of edges whose removal seperates s and t.

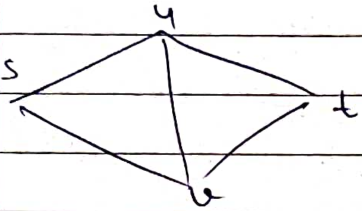


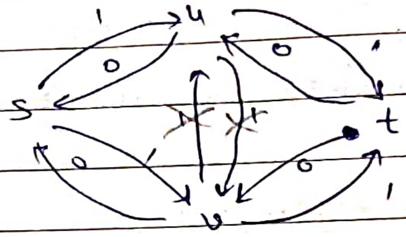
remove cycle before exploring starts



(*) ~~Node disjoint s-t paths~~

example

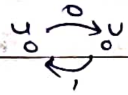




Tip: remove cycles corresponding to a edge

remove

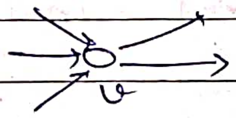
or



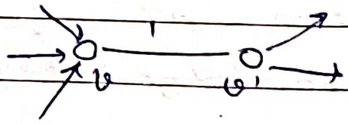
No need to remove

(*) Node disjoint s-t paths.

Given a directed graph G with $s, t \in V$, Find the max number of node disjoint s-t paths in G.



G



G'

Extensions to Max-Flow problem

→ circulation with Demands

→ circulation with Demands and lower bounds.

circulation with Demands.

$G = (V, E)$

d_v → demand associated to any node v

if $d_v > 0$: indicates node v has a demand of d_v for flow; node is a sink

if $d_v < 0$: indicates node v has a supply of d_v; node is a source

if $d_v = 0$: neither source nor sink.

A circulation with demand $\{d_v\}$ is a function f that assigns non-negative real numbers to each edge and satisfies

1) capacity conditions

For each edge $e \in E$ $\quad 0 \le f(e) \le c_e$

2) Demand condition

For each node $v \in V$

$$f^{in}(v) - f^{out}(v) = d_v$$

PAGE No. DATE / /

(7.49) If there exists a Feasible circulation with demands $\{d_v\}$

then $\sum_v d_v = 0$

Proof :

$$\sum_v d_v = \sum_v f^{in}(v) - \sum_v f^{out}(v)$$

For some edge $u \xrightarrow{e} u$

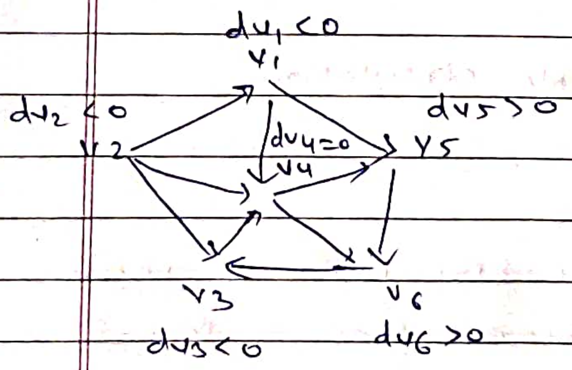
This edge will be $f^{out}(u)$ and $f^{in}(u)$

$\therefore$ each edge appears twice with +ve and -ve

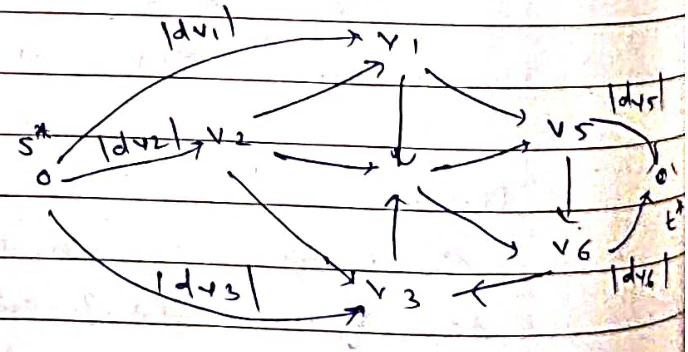
$$\therefore \sum_v d_v = \sum_v f^{in}(v) - \sum_v f^{out}(v) = 0.$$

$$\therefore \sum_{v: d_v > 0} d_v = - \sum_{v: d_v < 0} d_v$$

Demand = Supply.



G



G'

G' in directed graph without demands

we added super source to satisfy demands for all sources

(7.50) There is a feasible circulation with demands $\{d_v\}$ in G if and only if the maximum s^*-t^* flow in G' has value D. If all capacities and demands in G are integers, and there is a feasible circulation, then there is a feasible circulation that is integer-valued.

Run Max Flow on $G' \longrightarrow f$

Say $v(f) = x$

Can $x < D$?

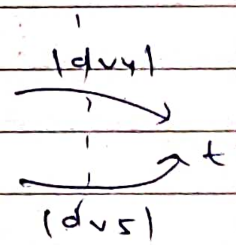
Yes, if no feasible circulation

Can $x > D$?

Not possible

$v(f) \leq c(A, B)$

any cut



$d_{vy} + d_{vs} = D$

$v(f) \leq D.$

Can $x = D$?

Yes, if feasible circulation.

PROOF:

(A) If there is a feasible circulation f with demand value $\{d_v\}$ in G, we can find a max flow in G' of value D.

(B) If there is a max flow in G' of value D, we can find a feasible circulation in G.

PAGE No.
DATE / /

(7.51) The graph G has a feasible circulation with demands $\{d_v\}$ if and only if for all cuts (A, B)

$$\sum_{v \in B} d_v \le c(A, B)$$

Circulation with Demands and Lower Bounds

$G = (V, E)$

$d_v \rightarrow$ demand associated with any node

1) capacity condition

For each $e \in E$ we have $l_e \le f(e) \le c_e$

2) Demand condition

For every $v \in V$ $\quad f^{in}(v) - f^{out}(v) = d_v$

Circulation with demands & lower bounds $\xrightarrow{\text{convert}}$ circulation with demands $\xrightarrow{\text{convert}}$ Max-flow.

Solution in 2 steps

Pass #1 Find f_0 to satisfy all l_e's
↳ removing capacity condition

Pass #2 use remaining capacity of the network to find a feasible circulation f_1 (if it exists)

Combine two flows $(f) = f_0' + f_1$

solution :

1. push flow f_0 through G where $f_0(e) = l_e$

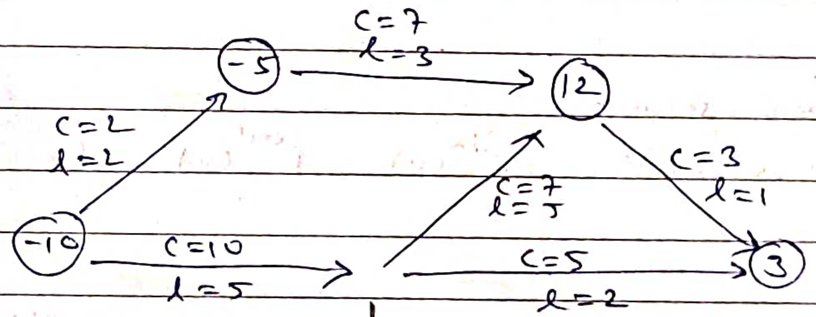
2. Construct G' where $c_e' = c_e - l_e$
$d_v' = d_v - L_v$

3. Find feasible circulation in G'
call this f'

4. If no feasible circulation in G'
∴ no feasible circulation in G

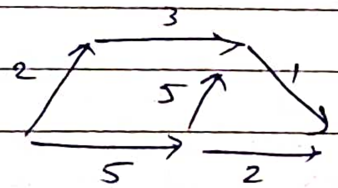
otherwise, feasible circulation in G
$= f_0 + f_1$

eg :

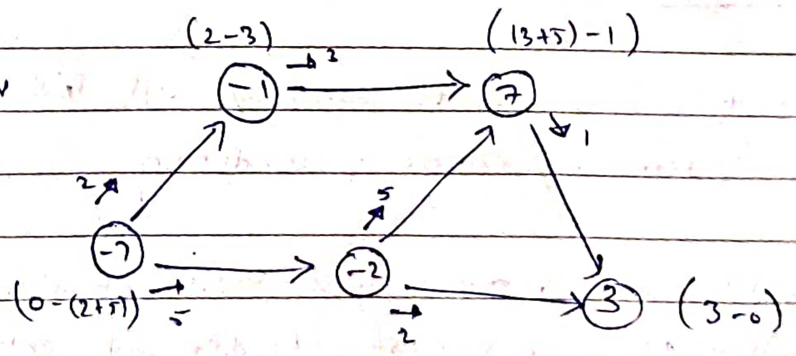


nothing mentioned means d=0

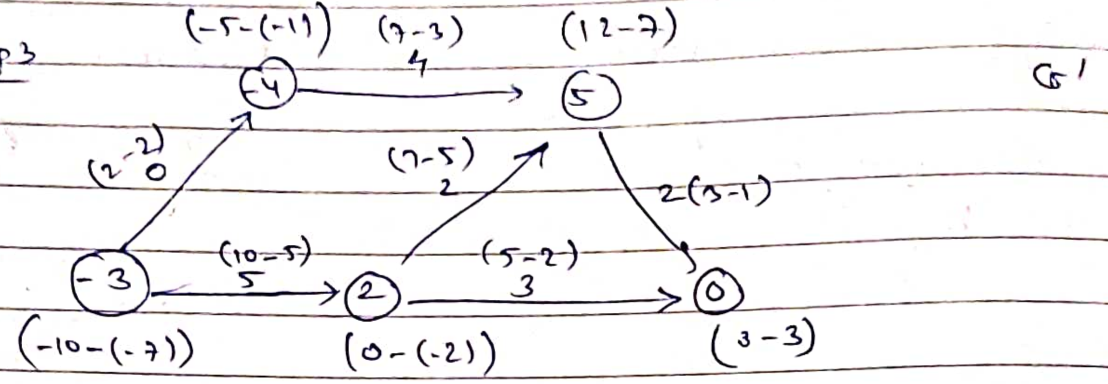
Step1 f_0



Step2 L_v



Step 3



Now this is ~~Max Flow~~ Circulation with Demands

Solve this to find flow f_1

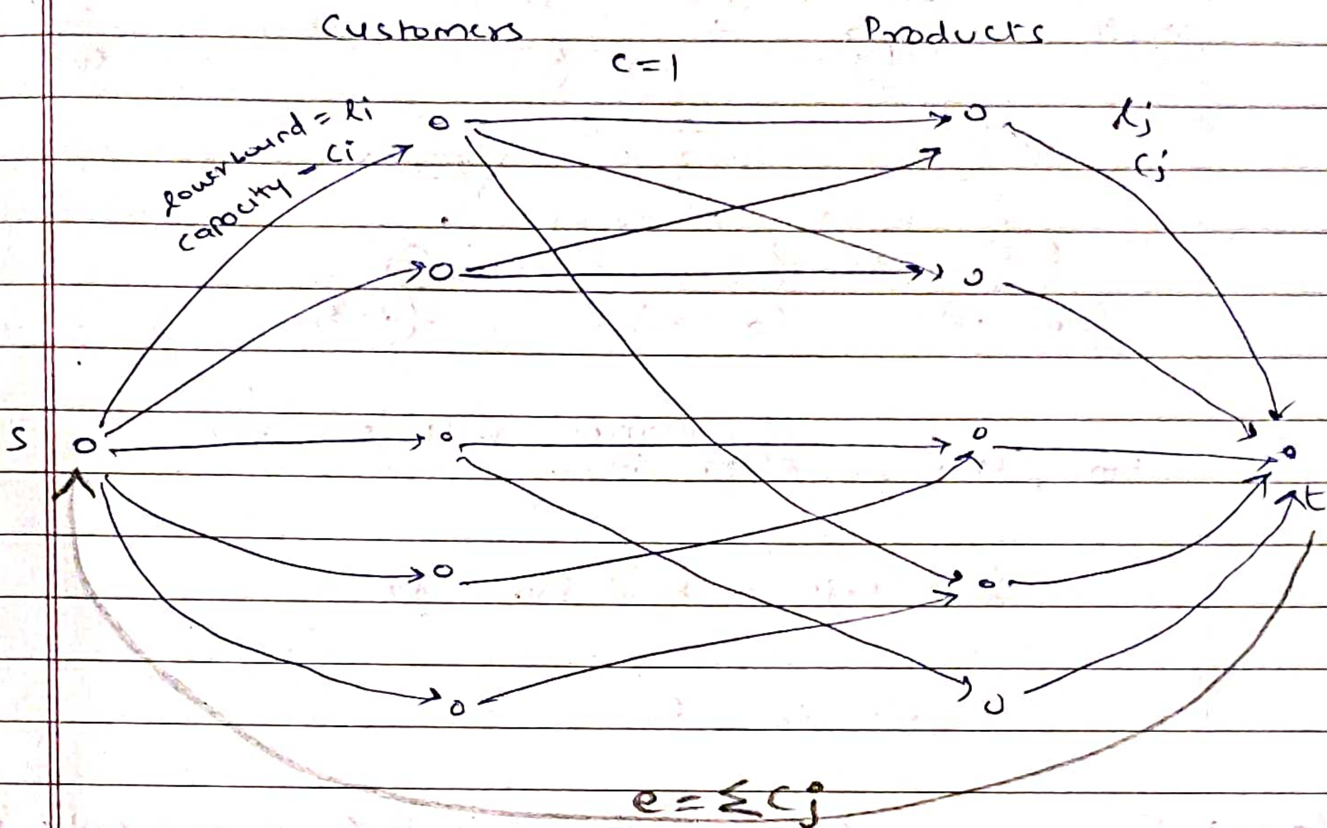
$f = f_0 + f_1$

(7.52) There is a feasible circulation in G if and only if there is a feasible circulation in G'. If all demands, capacities, and lower bounds in G are integers, and there is a feasible circulation, then there is a feasible circulation that is integer-valued.

Survey Design.

Input:
- Info on who purchased which products
- Maximum and Minimum number of questions to send to customer i
- Maximum and Minimum number of questions to ask about product j

PAGE No.	
DATE	/ /



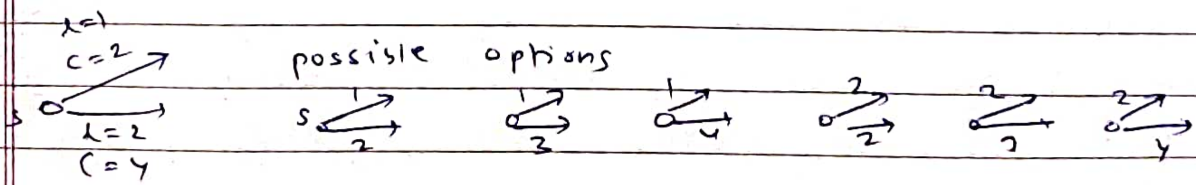
lowerbound = l_i = min Ques asked to customer i
Capacity = c_i = max Ques asked to customer i

$l_j \rightarrow$ min Ques asked about pdt j
$c_j \rightarrow$ max Ques asked about pdt j

This is Circulation with Demand & Lower Bound

But what is demand of source & sink

$d_{source} = \cancel{f^{in}(source)} - f^{out}(source)$
$= - f_{out}(source)$



Hence we do not exactly how much flow is there ~~to~~ out from source

Similarly for sink

$$d_{sink} = f^{in}(sink) - \cancel{f^{out}(sink)}^{0}$$
$$= f^{in}(sink)$$

$\therefore$ To make demands of source & sink $= 0$

we already know
$$-d_{source} = d_{sink}$$

$$d_{sink} = \sum_{\substack{e \text{ into} \\ \text{sink}}} c_e$$

$\therefore$ we add a backward edge from sink to source of capacity $= \sum_{j \text{ into } t} c_j$

(7.53) The graph G just constructed has a feasible circulation if and only if there is a feasible way to design the survey.

AoA | Week 10 Notes continue

(*) Min Flow Problem.

Input: Directed graph $G = (V, E)$

Source $s \in V$

sink $t \in V$

l_e for each edge $e \in E$

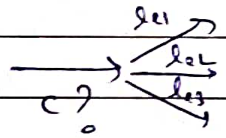
There are no c_e's

Objective: Find a feasible flow of minimum possible value.

SOLUTION :

PASS #1

Satisfy min capacity condition.



$c = \sum l_e$ for all edges

Initially $c = \infty$ replace ∞ with $\sum l_e$

Now find feasible circulation.

Solution

1. Assign "large" capacities to all edges and find a feasible flow f.

excess flow going through every edge

2. Construct G' where all the edges are reversed and the reversed edge e has capacity $= f_e - l_e$

PAGE No.	
DATE	/ /

Find maximum of this excess flow

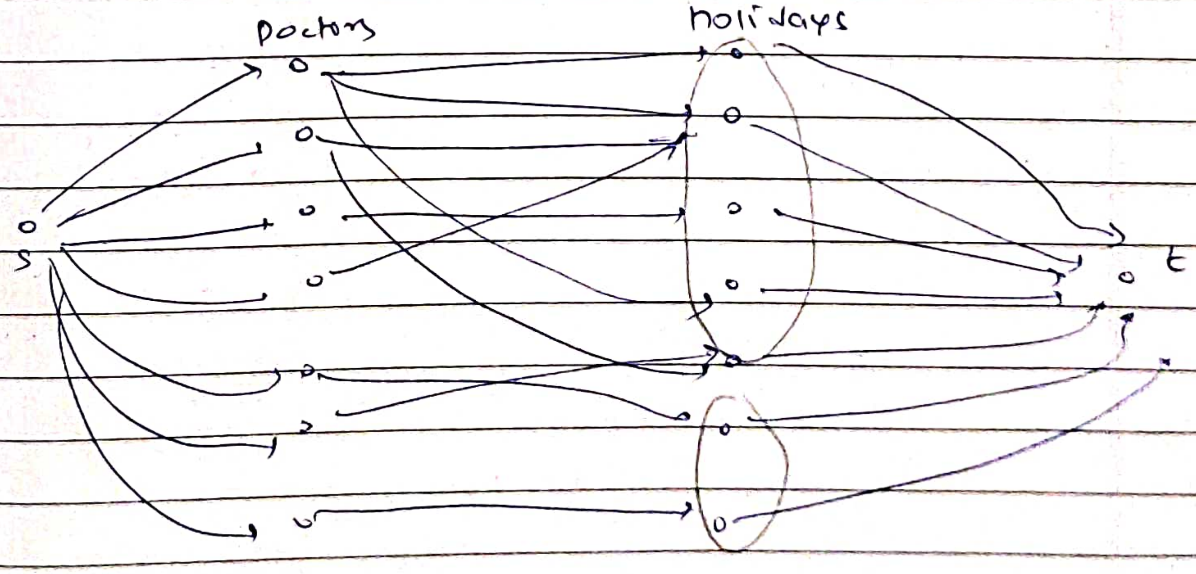
3. Find maximum flow from $\underline{t}$ to $\underline{s}$ in G'

4. Min flow $= f - f'$

minflow − flow from satisfying constraints − Max excess flow.

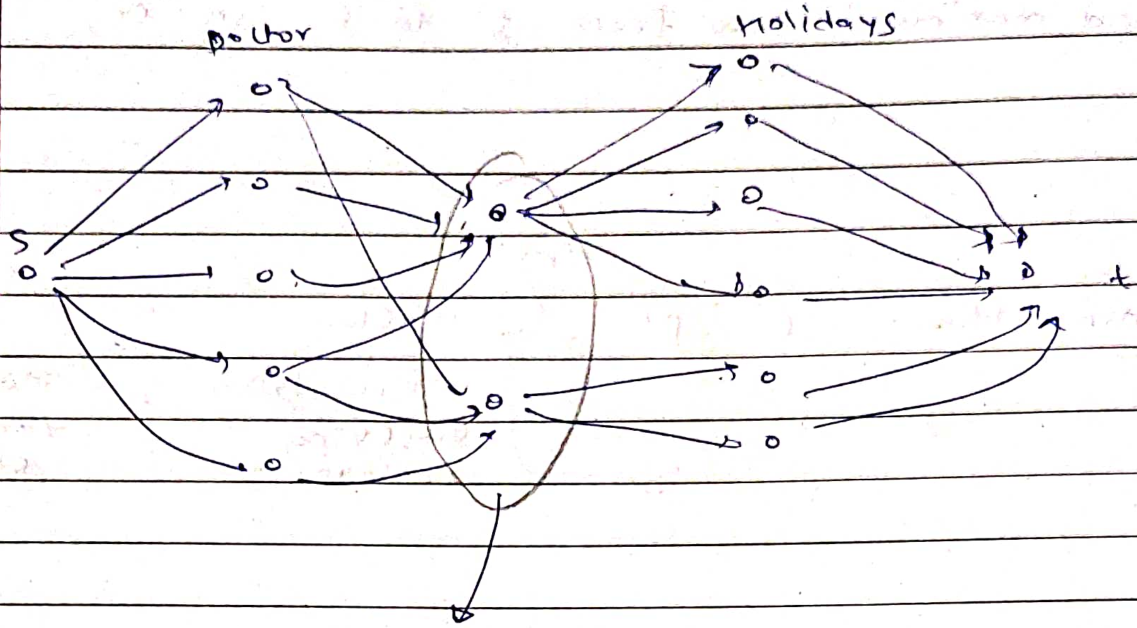
Doctors without weekend problem. (using gadgets)

- Each holiday i consists of h_i consecutive days
- Each doctor specifies which of the days within each holiday (0 or more) they are available to be on call.
- Cannot have a doctor assigned to more than one day in each holiday
- Cannot assign a doctor j to more than d_j days total across all holidays.



No doctor has working day more than 1 in a particular set

PAGE No. DATE / /



These nodes physically do not correspond to anything

They are just to satisfy the constraint that no doctor should have more than 1 working day in a particular holiday set.

Hence they are called gadgets