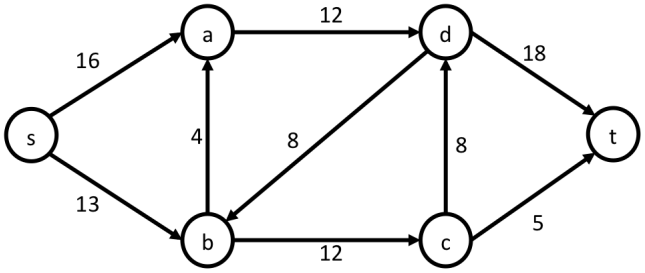
Name: Ananye Agarwal **Entry Number:** 2017CS10326
Name: Kabir Tomer **Entry Number:** 2017CS50410
Name: Rajat Jaiswal **Entry Number:** 2017CS50415

---

1. (20 points) Consider the network shown in the figure. Consider running the Ford-Fulkerson algorithm on this network.



(a) We start with a zero s-t flow f. The algorithm then finds an augmenting path in G_f. Suppose the augmenting path is $s \to b \to c \to t$. Give the flow f' after augmenting flow along this path.

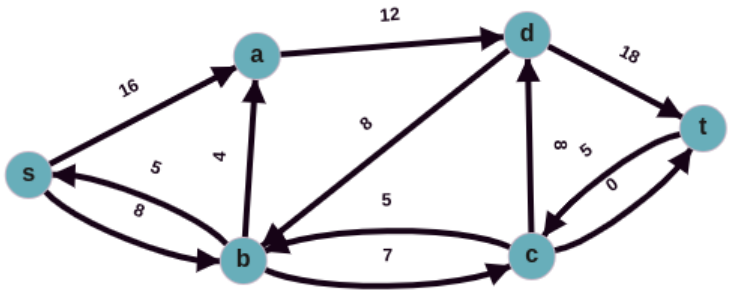
Solution:

$f'(s,a)$	0	$f'(s,b)$	5	$f'(d,b)$	0
$f'(b,a)$	0	$f'(a,d)$	0	$f'(b,c)$	5
$f'(c,d)$	0	$f'(c,t)$	5	$f'(d,t)$	0

Table 1: Flow f' after first iteration.

(b) Show the graph $G_{f'}$. That is, the residual graph with respect to s-t flow f'.

Solution:



Residual graph G_f after first iteration.

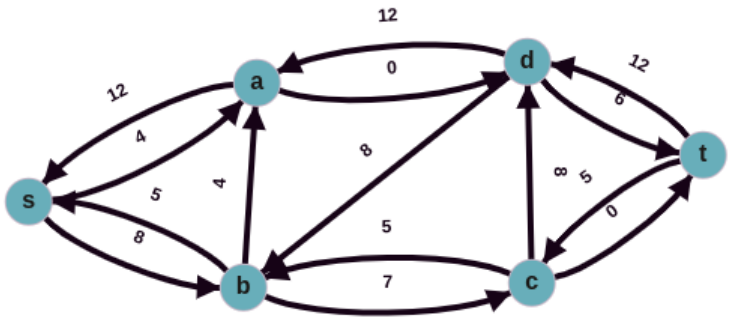
(c) The algorithm then sets f as f' and G_f as $G_{f'}$ and repeats. Suppose the augmenting path chosen in the next iteration of the while loop is $s \to a \to d \to t$. Give f' after augmenting flow along this path and show $G_{f'}$.

Solution:

$f'(s,a)$	12	$f'(s,b)$	5	$f'(d,b)$	0
$f'(b,a)$	0	$f'(a,d)$	12	$f'(b,c)$	5
$f'(c,d)$	0	$f'(c,t)$	5	$f'(d,t)$	12

Table 2: Flow f' after second iteration.

Solution:



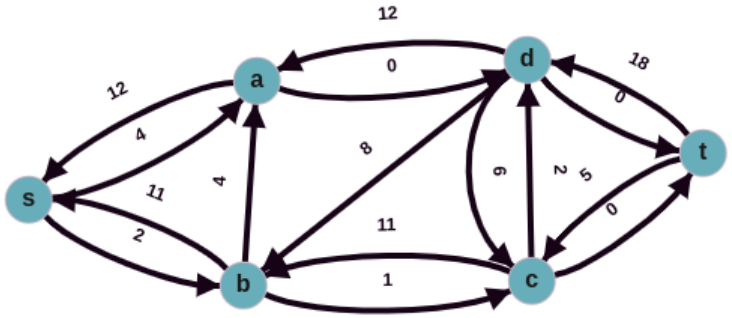
Residual graph G_f after second iteration.

(d) Let f be the flow when the algorithm terminates. Give the flow f and draw the residual graph G_f.

Solution:

$f(s,a)$	12	$f(s,b)$	11	$f(d,b)$	0
$f(b,a)$	0	$f(a,d)$	12	$f(b,c)$	11
$f(c,d)$	6	$f(c,t)$	5	$f(d,t)$	18

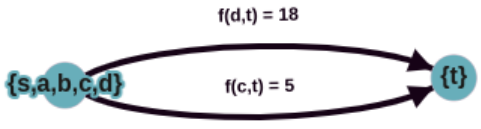
Table 3: Flow f after algorithm terminates.



Residual graph G_f after algorithm terminates.

(e) Give the value of the flow f when the algorithm terminates. Let A^* be the vertices reachable (using edges of positive weight) from s in G_f and let B^* be the remaining vertices. Give A^* and B^*. Give the capacity of the cut (A^*, B^*).

Solution: The flow f is the same as given in `table 3`. The value of this flow $v(f)$ is 23. A^* comes out to be the set, $A^* = \{s, a, b, c, d\}$, and B^* comes out to be the set, $B^* = \{t\}$. The capacity of cut $C(A^*, B^*) = c((d,t)) + c((c,t)) = 18 + 5 = 23$.



$Cut(A^*, B^*)$.

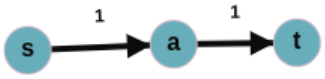
2. Answer the following:

(a) (1 point) State true or false: For every s-t network graph G, there is a unique s-t cut with minimum capacity.

Solution: False.

(b) (4 points) Give reason for your answer to part (a).

Solution: We will give a counter example for the given statement. Consider the given graph G below. It has two possible min-cuts. $CUT(A_1, B_1)$, where $A_1 = \{s, a\}$ and $B_1 = \{t\}$. $CUT(A_2, B_2)$, where $A_2 = \{s\}$ and $B_2 = \{a, t\}$. Both are possible $s-t$ cuts with a capacity of 1, which is minimum possible.



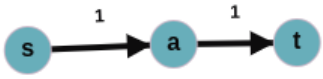
Graph G.

(c) (1 point) State true or false: For every s-t network graph G and any edge e in the graph G, increasing the capacity of e increases the value of maximum flow.

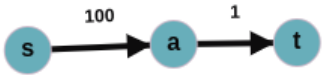
Solution: False.

(d) (4 points) Give reason for your answer to part (c).

Solution: We will give a counter example for the given statement. Consider the given graph G below, and the edge $e = (s, a)$. Increasing the capacity of e to 100 from 1(Graph G'), doesn't increase the value of maximum flow. Value of maximum flow still remains 1.



Graph G.



Graph G'.

(e) (1 point) State true or false: For every s-t network graph G for which an s-t flow with non-zero value exists, there exists an edge e in the graph such that decreasing the capacity of e decreases the value of maximum flow.

Solution: True.

(f) (4 points) Give reason for your answer to part (e).

Solution: The `Max-flow-min-cut` theorem proved in class states that, In every $s-t$ network graph, the maximum value of $s-t$ flow is equal to the minimum capacity of $s-t$ cut. Let $Cut(A, B)_G$ be such a min-cut of G and the value of the max-flow of this graph G be v_1. Let capacity of this min-cut be $C(A, B) = c_1$. Then as per the `Max-flow-min-cut` theorem, $c_1 = v_1$.

Consider a vertex $a \in A$ and $b \in B$. Let the edge $e = (a, b)$. Such an edge has to exist in G, otherwise there would be no connection between A and B and hence no path from s to t, and therefore the max-flow of G will be 0, which is a contradiction. We decrease the capacity of this edge e, $c(e) \to c(e) - \delta, \delta > 0$. Call this new graph, G'. Let the value of the max-flow of G' be v_2. The new capacity of the $Cut(A, B)$ in G' is $C(A, B)_{G'} = c_2 = c_1 - \delta < c_1$.

Claim 2.1: Let f be any $s-t$ flow and $Cut(A, B)$ be any $s-t$ cut. Then $v(f) \leq C(A, B)$.

`Claim 2.1` has already been proven in class. Using this claim in G', we take f as the max-flow of G' and (A, B) as the cut defined above in G', we get, $v(f) = v_2 \leq c_2 = C(A, B)_{G'}$. This implies, $v_2 \leq c_2 < c_1 = v_1$, therefore $v_2 < v_1$. Hence Proved.

3. Suppose you are given a bipartite graph (L, R, E), where L denotes the vertices on the left, R denotes the vertices on the right and E denote the set of edges. Furthermore it is given that degree of every vertex is exactly d (you may assume that $d > 0$). We will construct a flow network G using this bipartite graph in the following manner: G has $|L| + |R| + 2$ vertices. There is a vertex corresponding to every vertex in L and R. There is also a source vertex s and a sink vertex t. There are directed edges with weight 1 from s to all vertices in L and directed edges of weight 1 from all vertices in R to t. For each edge $(u, v) \in E$, there is a directed edge from u to v with weight 1 in G.

(*The figure below shows an example of a bipartite graph and the construction of the network.*)

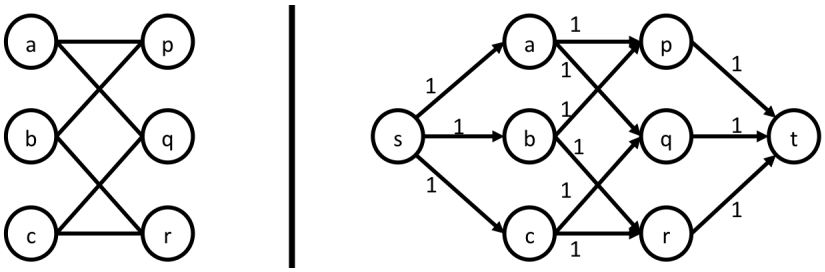


Figure 1: An example bipartite graph (with $d = 2$) and network construction.

(a) (5 points) Argue that for any such bipartite graph where the degree of every vertex is equal to d, $|L|$ is equal to $|R|$.

Solution: Since the graph is a bipartite graph, every edge is between a node from L and a node from R. Since every node has degree d, the number of edges connected to nodes belonging to L is $|L| * d$ because only one of the nodes of an edge can belong to L. Every edge is connected to a node from L so the total number of edges is exactly $|L| * d$. Similarly, the number of edges connected to nodes belonging to R is $|R| * d$, and since every edge is connected to a node from R, there are exactly $|R| * d$ edges. Therefore $|R| * d = |L| * d$, i.e. $|L| = |R|$.

(b) (10 points) Argue that for any given bipartite graph where the degree of every vertex is the same non-zero value d, there is an integer s-t flow (i,e., flow along any edge is an integer) in the corresponding network with value $|L|$.

Claim: For each subset $A \subseteq L, |N(A)| \geq |A|$, where $N(A)$ is the set of neighbours of A in (L, R, E).
Proof: Suppose $|N(A)| < |A|$. The degree of each node in A is d, and there are no edges connecting two nodes in A. Therefore the number of edges between a node in A and a node in $N(A)$ is $|A| * d$ (since all edges go from A to $N(A)$). $|N(A)| < |A|$, therefore by the pigeonhole principle, atleast one node in $N(A)$ must have degree $> d$, which is a contradiction.

By Hall's theorem, there exists a perfect matching, i.e., a matching of size $|L| = |R|$ for (L, R, E). Since this is also the maximum mapping, by the proof in Part (c) this means that the maximum flow for the corresponding network is equal to the size of the mapping, i.e., $|L|$

(c) (5 points) A matching in a bipartite graph $G = (L, R, E)$ is a subset of edges $S \subseteq E$ such that for every vertex $v \in L \cup R$, v is present as an endpoint of at most one edge in S. A maximum matching is a matching of maximum cardinality. Show that the size of the maximum matching in any bipartite graph $G = (L, R, E)$ is the same as the maximum flow value in the corresponding network graph defined as above.

Solution: We show that for every matching there is a flow with value equal to the size of the matching, and vice versa. This would imply that Show that the size of the maximum matching is equal to the maximum flow value.

Suppose there is a matching $S \subseteq E$. We construct the network flow as follows. We first start with flow zero for every edge. For every $(u, v) \in S$ such that $u \in L$ and $v \in R$, set $f(u, v) = 1$, $f(s, u) = 1$, and $f(v, t) = 1$. A node can appear as an endpoint in S only once, therefore at every internal node, either all flows are zero, or if it is present in S, the incoming/outgoing flow from $f(u, v)$ is compensated by the outgoing $f(v, t)$/incoming $f(s, u)$ flow, so conservation of flow holds. Also, the capacity of each edge is 1 and flow is always either zero or one in each edge. Therefore the f constructed is a valid integer s-t flow. The value is the sum of outgoing flows from s, i.e. $1 * |S| = |S|$. Therefore for every matching there is a flow of value equal to its size.

Suppose there is a network flow f. We construct the matching as follows. For every edge (u, v) such that $u \in L$ and $v \in R$ and $f(u, v) = 1$, add it to the matching. A node cannot appear in two edges with nonzero flow, since it has only one incoming edge (if in L) or only one outgoing edge (if in R) which means that if the incoming/outgoing flow is greater than one, it cannot be compensated for by the outgoing/incoming edges, and the sum of flow at the node is non zero. Therefore the constructed matching is valid. The size of the matching is same as the value of the flow (since we can take the cut of $\{s\} \cup L$ and see that the flow is equal to the number of edges added to the matching).

4. (20 points) There are n stationary mobile-phones $c_1, ..., c_n$ and n stationary mobile-phone towers $t_1, ..., t_n$. The distances between mobile-phones and towers are given to you in an $n \times n$ matrix d, where $d[i, j]$ denotes the distance between phone c_i and tower t_j. It is possible for a mobile-phone c_i to connect to a tower t_j if and only if the distance between c_i and t_j is at most D, where D is the connecting radius. Furthermore, at one time, a mobile-phone can connect to at most one tower and a tower can allow at most one connection. Your goal as a Communications Engineer is to figure out whether all mobile-phones are usable simultaneously. That is, whether it is possible for all mobile-phones to connect simultaneously to distinct towers. Answer the following questions.

   (a) Consider a simple example with 5 mobile-phones and 5 towers. Let the connecting radius be $D = 2$ miles. The distance matrix for this example is as given in Figure 2.

d	1	2	3	4	5
1	1	2	3	4	7
2	4	1	1	5	12
3	5	7	2	1	11
4	4	3	6	1	1
5	1	21	8	9	13

Figure 2: Distance matrix d for part (a) of question 4.

Prove or disprove: It is possible for all 5 mobile-phones to simultaneously connect to distinct towers for this example.

Solution (a) True. It is easy to see that this is possible – c_1 connects to t_2, c_2 connects to t_3, c_3 connects to t_4, c_4 connects to t_5 and c_5 connects to t_1.

(b) Design an algorithm that takes inputs n, D, and the distance matrix d, and outputs "yes" if it is possible for all mobile-phones to simultaneously connect to distinct towers (within the connecting radius D) and "no" otherwise. Analyze the running time of the algorithm and give proof of correctness.

(b) Suppose we represent the problem as a bipartite graph $G = (C, T, E)$ where $C = \{c_1, \ldots, c_n\}$ and $T = \{t_1, \ldots, t_n\}$ and c_i, t_j are connected if $d[i, j] \leq D$. Choosing distinct towers for each cellphone to connect to amounts to finding a matching in this graph. Each cellphone can be connected to a distinct tower iff G has a perfect matching. As shown in class, G has a perfect matching if the size of its maximum matching is n. We can use the maximum matching algorithm as discussed in class to determine if a perfect matching exists. We give the pseudocode of the solution to the problem in the function ConnectMobiles.

```
1:  function ConnectMobiles(n, D, d)
2:      Declare C = {c_1, c_2, ..., c_n}
3:      Declare T = {t_1, t_2, ..., l_n}
4:      Declare empty adjacency list E
5:
6:      for i from 1 to n do
7:          for j from 1 to n do
8:              if d[i, j] ≤ D then
9:                  add t_j to the list E[c_i]
10:             end if
11:         end for
12:     end for
13:
14:     M = MaximumMatching(C, T, E)
15:     if |M| = n then
16:         return yes
17:     else
18:         return no
19:     end if
20: end function
```

The MaximumMatching function returns a matching M which is a set of tuples (c_i, t_j) such that c_i is matched to t_j.
Running Time: The lines 2, 3 each take $O(n)$ time. The two for loops 6-12 take $O(n^2)$ time. Since the graph $G = (C, T, E)$ has $2n$ nodes and atmost n^2 edges (each tower may be close to each cell phone in the worst case), the runtime of line 14 is $O(n^3)$. Therefore, the overall runtime of the algorithm is $O(n^3)$.

5. (25 points) Town authorities of a certain town are planning for an impending virus outbreak. They want to plan for panic buying and taking cues from some other towns, they know that one of the items that the town may run out of is toilet paper. They have asked your help to figure out whether the toilet paper demand of all n residents can be met. They provide you with the following information:

   - There are n residents $r_1, ..., r_n$, m stores $s_1, ..., s_m$, and p toilet paper suppliers $x_1, ..., x_p$.
   - The demand of each of the residents in terms of the number of rolls required.
   - The list of stores that each of the residents can visit and purchase rolls from. A store cannot put any restriction on the number of rolls a customer can purchase given that those many rolls are available at the store.
   - The number of rolls that supplier x_j can supply to store s_i for all $i \in \{1, ..., m\}$ and $j \in \{1, ..., p\}$.

The above information is provided in the following data structures:

- A 1-dimensional integer array $D[1...n]$ of size n, where $D[i]$ is the demand of resident r_i.
- A 2-dimensional 0/1 array $V[1...n, 1...m]$ of size $n \times m$, where $V[i,j] = 1$ if resident r_i can visit store s_j and $V[i,j] = 0$ otherwise.
- A 2-dimensional integer array $W[1...m, 1...p]$ of size $m \times p$, where $W[i,j]$ is the number of rolls of toilet paper that the supplier x_j can supply to store s_i.

Design an algorithm to determine if the demand of all residents can be met. That is, given (n, m, p, D, V, W) as input, your algorithm should output "yes" if it is possible for all residents to obtain the required number of rolls and "no" otherwise. Argue correctness and discuss running time.

(*For example, consider that the town has two residents, one store and two suppliers. If* $D = [2,2], V = \left[\begin{smallmatrix}1\\1\end{smallmatrix}\right]$, *and* $W = [2,2]$, *then the demand can be met. However, if* $D = [2,2], V = \left[\begin{smallmatrix}1\\1\end{smallmatrix}\right]$, *and* $W = [2,1]$, *then the demand cannot be met.*)

Solution. Consider the function SupplyTP which checks it is possible to satisfy the demand for toilet paper for each resident. The MaxFlow(V, E, α, β) function returns the maximum flow possible in the weighted directed graph $G = (V, E)$ between source $\alpha \in V$ and sink $\beta \in V$ where E is an adjacency list indexed by the vertex $v \in V$ and $E[v]$ is a list of tuples of the form (v', w) where w is a weight and $v' \in V$ is a vertex.

Running time analysis: Notice the following

- Line 2 takes time $O(n + m + p)$, while line 3 is $O(1)$
- For loop 5-7 takes time $O(p)$
- For loops 9-13 take time $O(pm)$
- For loops 15-21 take time $O(mn)$
- For loop 24-26 takes time $O(n)$
- The summation in line 28 takes time $O(n)$
- The maximum possible flow in the network is bounded by D^* since the sum of incoming edges to the sink β is D^*. The total number of edges in the network is $|E| = p + pm + mn + n = O(pm+mn)$. Therefore, the overall running time of the call to MaxFlow is $O((pm+mn)D^*)$.

The running time is dominated by line 29, therefore the overall runtime is $O((pm + mn)D^*)$.

Proof of Correctness: Notice that it is sufficient to prove the following claim

Claim 5.1: It is possible to satisfy the demand for toilet paper iff the max flow in the graph $G = (V, E)$ (as defined in SupplyTP) is $D^* = D[1] + D[2] + \ldots + D[n]$

We prove both directions of implications of the above claim.

$\boxed{\Rightarrow}$ Suppose it is possible to satisfy the demand for toilet paper. In this scenario, suppose person r_i buys y_{ij} rolls from store s_j. Further, let z_{jk} be the net number of rolls supplied to store s_j from supplier x_k. Suppose that in the end all the rolls are sold (we can imagine that any unsold rolls are returned to the suppliers) In this case,

$$\text{Number of rolls bought by person } i = D[i] = \sum_j y_{ij}$$

Likewise,

$$\text{Number of rolls bought from store } j = \sum_k z_{jk}$$

Notice that we can interpret these numbers as flows along the edges in graph G. In particular, we assign the following flows

- Flow along edge (α, x_i) is $\sum_j z_{ji}$
- Flow along edge (x_k, s_j) is z_{jk}
- Flow along edge (s_j, r_i) is y_{ij}
- Flow along edge (r_i, β) is $\sum_j y_{ij}$

We can check that at each internal node flow is conserved.

- At node x_k, outgoing flow is $\sum_j f((x_k, s_j)) = \sum_j z_{jk}$ which is also the incoming flow
- At node s_j flow is conserved since we assume that all rolls are sold
- At node r_i, incoming flow is $\sum_j f((s_j, r_i)) = \sum_j y_{ij}$ which is also the outgoing flow

Further, at each edge capacity constraints are satisfied.

- Edges of the form (α, x_k), (s_j, r_i) have infinite capacity and therefore the constraints satisfied by default.
- Edges of the form (x_k, s_j) have capacity $W[j, k]$ which is satisfied since the number of rolls supplied from x_k to s_j (y_{kj}) is less than $W[j, k]$ as given in the question.
- Edges of the form (r_i, β) have capacity $D[i]$ which is satisfied since the flow assigned to these edges is precisely $D[i]$.

This implies that this is a valid flow assignment. The flow in this graph can be calculated by summing incoming flows at β. This equals $\sum_i \sum_j y_{ij} = \sum_i D[i] = D^*$ as desired.
$\boxed{\Leftarrow}$ Suppose the maximum flow in the graph is D^*. Since the capacity of each edge (r_i, β) is $D[i]$ this implies that the sum of capacities of incoming edges at β is D^*. Therefore, each edge must be carrying flow equal to its capacity. We can now interpret the flows along the edges (s_j, r_i) as toilet paper rolls supplied from store s_j to resident r_i and the flows along edges (x_k, s_j) as toilet paper rolls supplied from supplier x_k to store s_j. The way the capacities have been assigned to the graph G implies that it is indeed possible to supply the rolls such that the constraints given in the question are met.

```
1:  function SupplyTP(n, m, p, D, V, W)
2:      Declare V = {r_1, ..., r_n} ∪ {s_1, ..., s_m} ∪ {x_1, ..., x_p} ∪ {α, β}
3:      Declare empty adjacency list E
4:
5:      for i from 1 to p do
6:          Add (x_i, ∞) to E[α]
7:      end for
8:
9:      for i from 1 to p do
10:         for j from 1 to m do
11:             Add (s_j, W[j, i]) to E[x_i]
12:         end for
13:     end for
14:
15:     for i from 1 to m do
16:         for j from 1 to n do
17:             if V[j, i] = 1 then
18:                 Add (r_j, ∞) to E[s_i]
19:             end if
20:         end for
21:     end for
22: (contd..)
```

```
23:
24:     for i from 1 to n do
25:         Add (β, D[r_i]) to E[r_i]
26:     end for
27:
28:     total_demand = D[1] + D[2] + ... + D[n]
29:     met_demand = MaxFlow(V, E, α, β)
30:
31:     if met_demand < total_demand then
32:         return no
33:     else
34:         return yes
35:     end if
36: end function
```