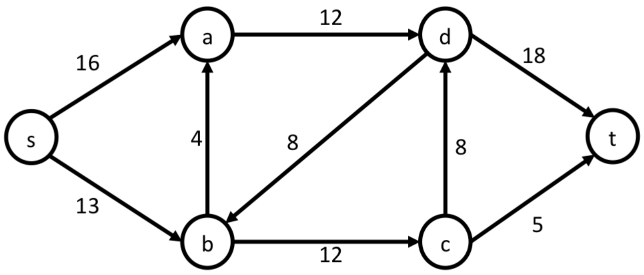
- The instructions are the same as in Homework-0, 1, 2.

There are 5 questions for a total of 100 points.

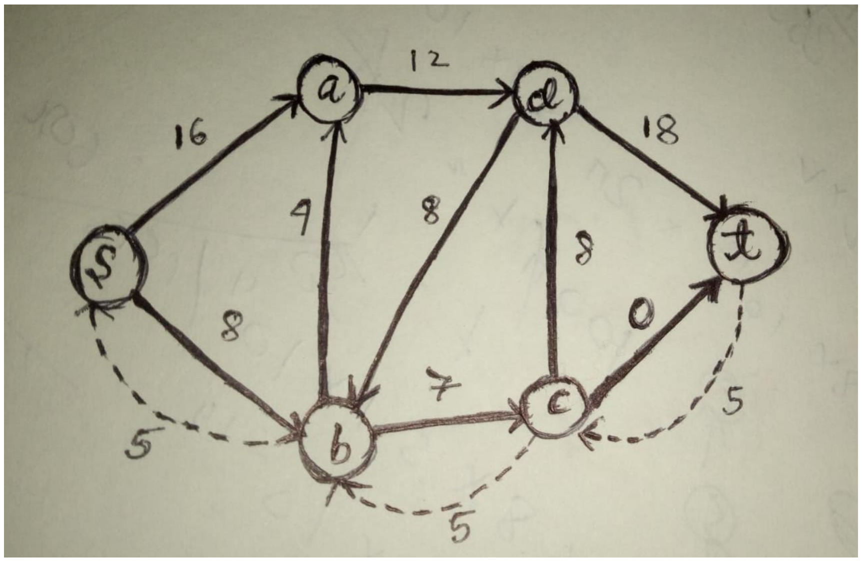
1. (20 points) Consider the network shown in the figure. Consider running the Ford-Fulkerson algorithm on this network.



(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.
Answer: The required flow f', after augmenting the flow along given path for respective edges is as follows::

1. f'(s,b)=5
2. f'(b,c)=5
3. f'(c,t)=5
4. for all other edges, 0

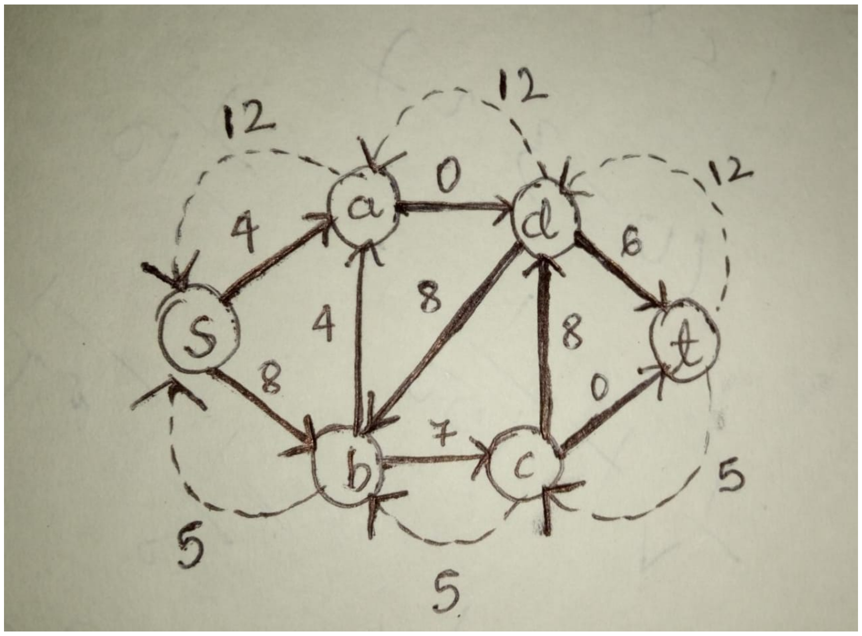
(b) Show the graph $G_{f'}$. That is, the residual graph with respect to s-t flow f'.
Answer: The required graph is as follows:



(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 $G_{f'}$.
Answer: The required flow, f' for different edges is as follows:

1. f'(s,b)=5
2. f'(b,c)=5
3. f'(c,t)=5
4. f'(s,a)=12
5. f'(a,d)=12
6. f'(d,t)=12
7. for all other edges, 0

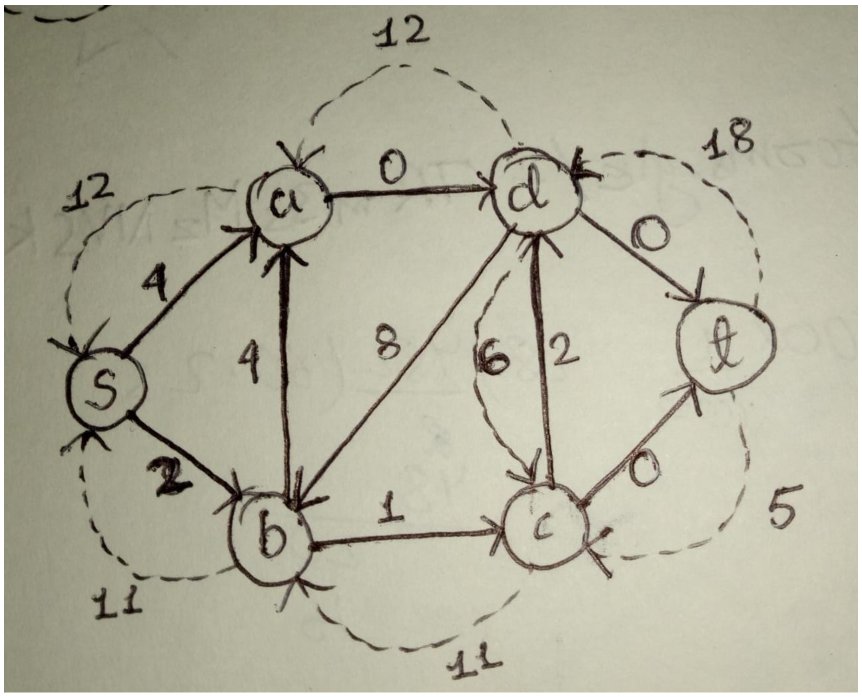
The graph $G_{f'}$ is as follows:



(d) Let f be the flow when the algorithm terminates. Give the flow f and draw the residual graph G_f.
Answer: The flow f, when the algorithm terminates is as follows:

1. f'(s,b)=11
2. f'(b,c)=11
3. f'(c,t)=5
4. f'(s,a)=12
5. f'(a,d)=12
6. f'(d,t)=18
7. f'(c,d)=6
8. for all other edges, 0

The required graph is as follows:



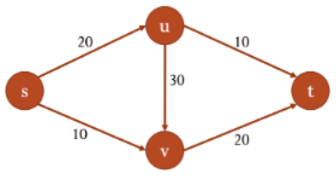
(e) Give the value of the flow f when the algorithm terminates. Let A^* be the vertices reachable (using edges of positive weight) from s in G_f and let B^* be the remaining vertices. Give A^* and B^*. Give the capacity of the cut (A^*, B^*).
Answer: The value of flow f, when the algorithm terminates is 23. The set A^* is $\{s, a, b, c, d\}$ and the set B^* is {t}. The capacity of the cut (A^*, B^*) is 23.
Working :
capacity of cut C(A*,B*) = f^{out}(A)
= 18 (d to t) + 5 (c to t)
= 23

We note that max flow = 23 = minimum capacity of cut
This is in accordance with the max-flow-min-cut theorem.

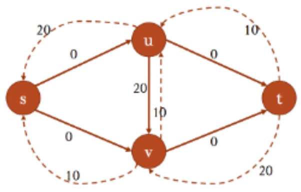
2. Answer the following:

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

(b) (4 points) Give reason for your answer to part (a).
Answer: Here is a counterexample,
Consider the following graph:



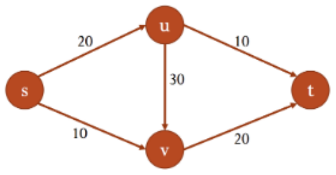
The final residual graph is as follows:



We can see there are two possible cuts with minimum capacity, namely {(s),(u,v,t)} and {(s,u,v),(t)}. Both cuts have capacity = 30. Hence, the statement is false.

(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.
Answer: False

(d) (4 points) Give reason for your answer to part (c).
Answer: Lets take the example of graph below:



We have seen the residual graph of the above graph (in 1b), and now if we increase the weight of edge (u,v), there value of maximum flow will not increase as the values of cuts with minimum capacity is still the same and both quantities ie value maximum flow and value of minimum capacity cut are numerically equal. Hence, the given statement is not true.

(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.
Answer: True

(f) (4 points) Give reason for your answer to part (e).
Answer: We know that for a network graph having non-zero s-t flow, there exists a cut (with at least 1 edge) with minimum capacity, which is the value of maximum flow, which is indeed the sum of edges going from the part of the cut containing s, to the part of the cut containing t. Now if we decrease the weight of any of these edges, the maximum flow will decrease, since the capacity of

minimum capacity cut further decreases.
Hence, there exists an edge in every network graph, such that decreasing the weight of the edge decreases the maximum flow.

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.*)

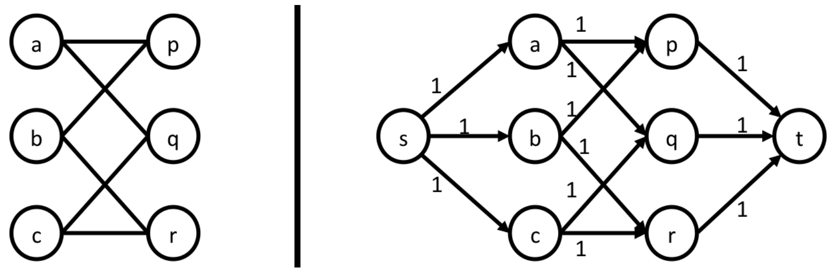


Figure 1: An example bipartite graph (with $d = 2$) and network construction.

(a) (5 points) Argue that for any such bipartite graph where the degree of every vertex is equal to d, $|L|$ is equal to $|R|$.
Answer:
It is given that all the vertices in $L \cup R$ have degree d. This implies that each vertex is connected to d edges. We know that for any edge $e = (u, v) \in E$, one vertex is in L and the other vertex is in R, this follows from the definition of bipartite graph.
Claim 1. The number of edges with one vertex being in set L is $d|L|$.

$$\begin{aligned} The\ number\ of\ edges from\ one\ vertex &= d \\ \implies The\ number\ of\ edges\ from\ all\ vertices\ in\ L &= d * (number\ of\ vertices\ in\ L) \\ &= d|L| \end{aligned}$$

Similarly,
Claim 2. The number of edges with one vertex being in set R is $d|R|$.
From the above 2 claims and from the fact that all the edges have one vertex in L and one in R, the above 2 values are equal to $|E|$.
$\implies d|L| = d|R|$
$\implies |L| = |R|$
Hence proved.

(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 $|L|$.
Answer:
Using Hall's Theorem, which states that Given any bipartite graph G = (X, Y , E), there is a perfect matching in G if and only if for every subset $A \subseteq X$, we have $|A| \leq |N(A)|$, we can prove that there exists a perfect matching in G.
Consider any subset A of G(L,R,E),i.e ∀ a ∈ A, a ∈ L.
This implies that there are no edges between any 2 vertices in A. Therefore, all edges from A have the opposite vertex lying outside of A, in N(A). For the set E' containing all edges (x,y) where $y \in N(A)$, we know that $\forall a \in A$, $\exists$ *edge* (a, y) s.t $(a, y) \in E'$ since all of As neighbours lie in

N(A). There can exist x $\notin$ A s.t (x,y) $\in E'$, let the set of all these vertices be A'. So we the set $A \cup A'$ that maps edges to set $N(A)$ forms a sub bipartite graph. Since all vertices have degree d, using result of Q3a, we say that $|A \cup A'| = |N(A)|$. Since $A \cap A' = \Phi$ and $|A'| \geq 0$, this implies

$$|A| + |A'| = |N(A)|$$
$$\implies |A| \leq |N(A)|$$

Therefore, according to Hall's Theorem, there exists a perfect matching in the bipartite graph (L,R,E).
Next we make the following claim:-
Claim 1. Suppose the bipartite graph has a matching of size k. Then there is an integer flow of value k in the network graph.
If the bipartite graph has a matching of size k, that implies there exist k distinct pairs of vertices (x,y) where $x \in L$ and $y \in R$, each with an edge between them from x to y in the network graph. So, for every vertex pair (x,y) we can consider the flow from $s- > x- > y- > t$ with flow value 1. The k such flows will not have a common edge as the pairs are distinct by definition, and each vertex in L has a unique edge from source s to it and each vertex in R has a unique edge from it to sink t. Therefore, the total value of the integer-flow will be equal to the number of vertices that are connected to the sink with non-zero flow edges (i.e the number of distinct pairs) which is k,as the value of each flow is 1.
Therefore, a matching of size k can be represented as an integer-flow of value k.
From the first proof, we can see that since a perfect matching exists,there exists a matching of size $|L|$. Using the above claim 1, this implies that an integer-flow of size $|L|$ exists.
Hence proved.

(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.

Answer:
To prove the above statement we will first prove the following 2 claims.
Claim 1. Suppose there is an integer flow of value k in the network graph. Then the bipartite graph has a matching of size k.
Proof.
In the given graph, if there is an integer flow of value k in the network graph, that implies that there there are k distinct paths from s to t, since each edge from s to L has a capacity 1.Therefore, there are k vertices in L, each having a distinct path to t, as part of the integer-flow, to carry the flow of value 1 entering that vertex forward to t.
Claim 1.1 A vertex in L will pass on the flow only to one vertex in R.
Since it is an integer flow and all the edges directed out of L to R have capacity 1, the flow of value 1 that enters a vertex in L can pass through only one of the outgoing edges, because if it split the individual flow values in those edges will be fractional.
Claim 1.2 A vertex in R will receive flow from only one of the incoming edges from L.
WLOG, let us assume the flow entering vertex a was passed to vertex q. From vertex q in R , by flow conservation the only path outward is to sink t. Note that since the capacity of the edge (q,t) is 1, only one of the edges entering vertex q can pass the flow. This implies that, if a is sending the flow of value 1 to q, then no other vertex can send flow through q since the edge (q,t) has reached it's capacity.

From the above 2 claims, it implies that there is a unique mapping from every vertex in L(receiving flow from s) to a vertex in R. Therefore, there we can define k unique mappings from L to R, given an integer flow of value k.
Claim 2. Suppose the bipartite graph has a matching of size k. Then there is an integer flow of value k in the network graph.
This Claim has been proven in Claim 1 in part 3b.
Using the above 2 claims, we can say that the size of the maximum matching in any bipartite graph G is the same as the maximum flow value in the corresponding network graph.

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.
   Answer: Yes, it possible for the distinct phones to connect to distinct towers. We can prove it by providing a matching that follows givven constraints. The require matching, x[i] which denotes the tower for ith cellphone is as follows:
   x[1] = 2, x[2] = 3, x[3] = 4, x[4] = 5, x[5] = 1
   Hence proved.

   (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.
   Answer:
   Main Idea: We will convert the given matrix d[i,j] into a bipartite graph G(X,Y,E), where e contains an edge between ith node of X and jth node of Y only if the value d[i,j] is less than or equal to D. And then we will apply the max matching function. If the number of nodes equal n(total nodes in X), then we output yes, otherwise no.

The algorithm implementing the above idea is as follows:

```
isPossible(d,D):
        - initialize a bipartite graph G(X,Y,E) with x containing the cells c1....cn, and Y containing
the towers t1...tn, and edge list as empty for now.
        - for i in range(0,n):
                - for j in range(0,n):
                        - if d[i,j] <= D:
                                - append edge (ci,tj) to the graph G
        - maxConnections = Max-Matching(G)
        - if maxConnections == number of nodes in X:
                - return "yes"
        - return "no"
```

Proof of Correctness: First of all, by converting the given matrix into a bipartite graph, we get rid of all the ways of reaching towers with distance more than D. So, our graph only contains edges which are reachable and where connections are plausible.
There is no distinction between the edges that are less than or equal to D, and hence we assumed all the edges to have weight 1. Now, we apply the max- matching function to the graph, which gives us the maximum matching possible for the given bipartite graph(we directly used the algorithm from class and hence it gives the right number).
If the maximum matching possible is equal to the number of cells given, it means that such a matching exists so that all cells are connected to different towers following given constraints. Hence we output "yes", otherwise "no".

Running Time: The formation of graph from the matrix requires an iteration over the whole matrix, which takes O(n^2) time. The max-matching process is just Ford-Fulkerson, and hence takes O(n(n+m)), where n is the number of nodes in X, and m is the number of edges. Hence, the net running time is O(n(n+m)).

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 $j \in \{1, ..., p\}$.

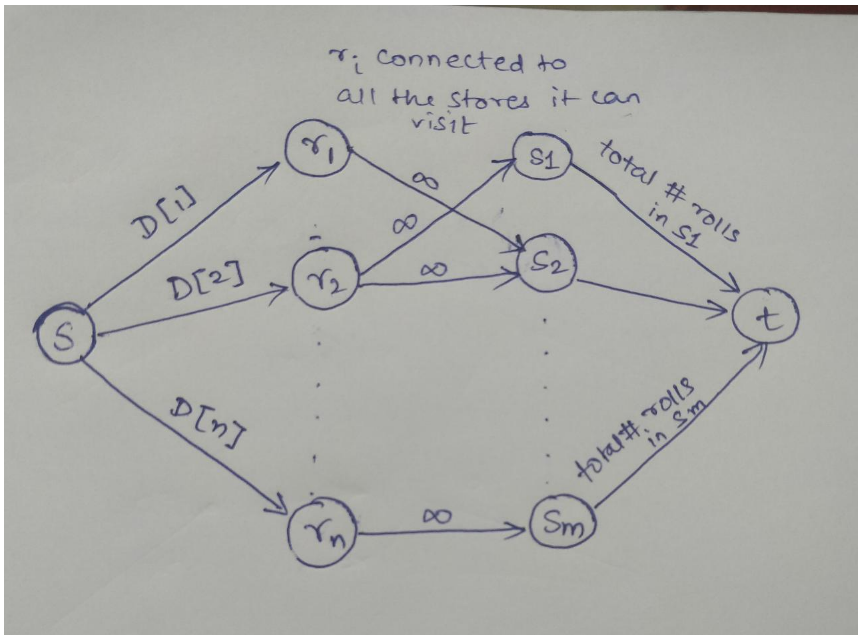
The above information is provided in the following data structures:

- 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.*)

Answer:



We can construct a network graph for the given problem as shown in the above image. Consider a bipartite graph G = (R,S,E) where R contains one vertex for each resident, S contains one vertex for each store,and an edge $e = (r_i, s_j) \in E$ if r_i can visit store s_j, i.e if V[i,j] = 1. In the network graph, we have a source node s from which there is an edge to all vertices of R, each having a capacity D[i]. The edges from S to the sink node t have capacity =total number of toilet paper rolls that the store has gotten from suppliers = $\sum_{j=1}^{p} W[i, j]$ for every store s_i.

Algorithm.

```
procedure FindEnoughRolls(n,m,p,D,V,W):
    // to construct the network graph
    G = empty graph
    Add source vertex s to G
    Add sink vertex t to G
    total-toilet-rolls-all = 0 // stores sum of all toilet roll demands of all residents
    for i from 1 to n:
        Add vertex $r_i$
        Add edge $(s, r_i)$ with capacity D[i]
        total-toilet-rolls-all += D[i]
    for j from 1 to m:
        Add vertex $s_j$
        for k from 1 to n:
            if(V[k,j]==1):
                Add edge $(r_k, s_j)$ with capacity $\infty$
        total-rolls = 0
        for l from 1 to p:
            total-rolls += W[j,l]
        Add edge $(s_j, t)$ with capacity = total-rolls
    max-toilet-papers-sold = Max-Flow-Algorithm(G)
    if(max-toilet-papers-sold == total-toilet-rolls-all):
        return "yes"
    else:
        return "no"
```

Proof of Correctness:
The path from s to t passing through r_i represents the total number of toilet rolls that r_i purchased. It will not exceed the demand of resident i since edge (s, r_i) has capacity = D[i]. The flow passing through vertex s_j represents the total number of rolls that store j has sold. The incoming edges(or flows) are from residents who can visit that store and hence the incoming flow is valid. The total number of rolls sold is capped at the number of rolls it got from all the suppliers(=capacity of edge (s_j, t). Therefore, the value of the s-t integer-flow of this graph is maximised and hence the total number of toilet paper rolls bought by the residents is optimised. If the maximum flow equals the total demand that implies that each of the residents got their demand of toilet rolls and hence the demand is met.
Running time analysis.
The total time to construct the graph is $O(n + mn + mp)$. The time complexity of max-flow algorithm which uses Ford-Fulkerson Algorithm is $O(C * |E|)$ where C is the maximum flow value and $|E|$ is the number of edges. Therefore, $C = \sum_{i=1}^{n} D[i]$.For our graph, $|V| = n+m+2$ and $|E| = O(mn+n+m) = O(mn)$. Therefore, the time complexity of max-flow algorithm is $O(C * |E|) = O(Cmn)$. Therefore the total time complexity is $O(Cmn + mn + n + mp) = O(Cmn + mp)$.