Problem 1

Here is the original graph:

15

4

3

10

10

5

12

7

In order to find the maximum flow, we have to conduct the Ford-Fulkerson algorithm. In order to conduct the algorithm, we also have to draw the residual graph, a graph that consists of edges with opposite directions combined with the existing graph that has currently the flow (weight) of 0. The weights of residual edges and the existing edges combined should have the weight of the original existing edge. For example, the combined weight of residual edge from 1 to s and original edge s to 1 should add up to the weight of original edge, which is 15. The graph below shows the graph with residual edges.

0

0

0

15

4

3

10

10

5

12

7

0

0

0

0

0

First, we know that that there exist simple paths from s to 1 to 2 to t and s to 3 to 4 to t. When we go through the path s to 1 to 2 to t, we know that the maximum amount of flow that can be sent is the minimum of weights from (s to 1, 1 to 2, 2 to t). In this case, the minimum of (15,12,7) is 7. Notice that if we go over 7, there might exist room from s to 1 and 1 to 2, but there exists an overflow from 2 to t. Similarly, from node s to 3 to 4 to t, we find the minimum to be 4 from (4,10,10). The graph below (the next page) is an update.

7

7

7

8

0

3

6

6

5

5

0

0

0

4

4

4

Now that we have sent weights 4 from s to 3 to 4 to t and weights 7 from s to 1 to 2 to t, we also have to send those weights backward from t to 4 to 3 to s and t to 2 to 1 to s, using the residual backward edges we created. Also, since the maximum capacity was sent from 2 to t of 7, we notice that we cannot send anymore flow through the edge from 2 to t. Similarly, we cannot send anymore flow through the edge from s to 3, since maximum flow 4 was already taken. However, we can still find augmenting paths to send further flow form source node to node t. We can move from node s to 1 to 2 to 3 to 4 to t. Notice that the capacities have decreased by amount that initial flow was sent through the edges. The method for amount of flow that can be sent is the same: find the minimum through the weighted edges: (s to 1, 1 to 2, 2 to 3, 3 to 4, 4 to t), which is min(8,4,3,6,6). The minimum is 3, so we can send 3 more flow from s to t using those paths. The graph below is an update.

10

10

7

5

0

0

3

3

5

2

0

3

0

4

7

7

The weights have been updated, with the residual paths. Notice that we also cannot send anymore flows from node 2 to 3, due to flow being 0. The only remaining path is to send from node s to 1 to 4 to t, using the residual path, but since no flow was passed from node 4 to 1 (the existing path in original question), the edge from 1 to 4 has the capacity of 0, which means that there are no more augmenting paths, allowing us to end the Ford-Fulkerson algorithm, giving the maximum capacity of 7+4+3 = 14.

The maximum amount of flow that can be sent in this question is 14.

Problem 2

If there are 20 workers and 40 set of tasks where each worker gets two tasks, this is similar to the marriage problem between n men and 2n women, where each man has the opportunity to marry 2 women, with no overlapping of women (with no concept of divorce where a man marries the same woman twice).

We can solve this problem by drawing a bipartite graph. Let Set A = 20 workers and Set B = 40 set of tasks. Draw |A| nodes on one side and |B| nodes on the other side (20 nodes on one side and 40 nodes on the other side). Then, draw a source node (node s) on the left of the 20 nodes and the destination node (node t) on the right side of the 40 nodes so that this problem can be turned into the flow network graph.

According to the marriage theorem, a bipartite graph has a perfect matching if and only if |A| = |B| and for every S ⊆ A we have |N(S)| ≥ |S|. Since each worker is assigned two jobs, we can allow to have 20 \* 2 nodes in set A and have a weighted edge of 1 going from the source node to the nodes in set A, fulfilling the |A| = |B|. This means that we are dividing one worker into two nodes since each worker can do two tasks. In addition, since each worker can do 6 jobs, we can interpret that each worker in the divided set of workers in set A can do 3 jobs where for the same worker a1 and a2, the set of tasks they can do are different. For example, if a1 and a2 are same workers and if a1 can do tasks b1, b2, and b3, a2 cannot do the tasks b1, b2, and b3 because they are the same worker; a2 must be assigned the set of tasks that are not b1, b2, and b3 (b4 to b40). After dividing this question, each worker in set A have three neighbor nodes in set B, which means that for every S ⊆ A we have |N(S)| ≥ |S| (every worker has 3 neighbor nodes). Therefore, according to the marriage theorem, we can always find a perfect matching.

Problem 3

Algorithm:

According to the question, each product pi in the set P has set of properties x and there are customers C that can review the product, depending on their properties.

This can be changed into the marriage problem (or perfect matching problem) in the following way:

1. If |C| < |P|, return “Impossible” If not, move on:
2. Draw |C| nodes in one side. (c1,c2,…,cn)
3. Draw |X| nodes on one side of the |C| nodes created in part 2 (x1,x2,…xk)
4. Draw |P| nodes on the other side of |X| nodes (p1,p2,…,pz)
5. Draw a source node (node s) at one end.
6. Connect the source node with nodes of customer |C| in a way that the edge points towards the customer nodes from the source node and give all of the edges the weight of 1.
7. For each customer, if the customer has the eligibility to review a property, connect them with an edge having a weight of 1000 (very large number) in a way that the directed edges point towards the |X| nodes (property nodes)
8. For each item, if the item has following properties, connect them with an edge having a weight of 100000 (very large number that is greater than |C| since it if all customer can review the same item, it should reach the item, meaning that capacity should be big enough) in a way that the directed edges point towards the |P| nodes (item nodes)
9. Draw a destination node (node t) at the other end.
10. Connect the destination node with the item nodes in a way that the directed edge is towards the destination node with weight of 1. At this point, the graph would look like:

Connect X1..Xn with P1…Pn nodes with weight of 100000 according to what property the item contains.

Connect c1..cn with x1…xn nodes with weight of 1000 according to what property each customer can review

1

1

1

1

1

…

…

…

1

1. Run the Ford-Fulkerson algorithm for the graph (the graph seems like tripartitie graph)
2. If the value returned from the running the Ford-Fulkerson algorithm equals the number of items (|P|), match each customer to their item chosen.
3. If not (value returned from the running the Ford-Fulkerson algorithm does not equal the number of items |P|), return “Impossible”

Proof:

The requirement for this question is that each customer reviews at most one of the item for whatever property in the set |X|. This means that it is not necessary for customers to all find a match (there can be some customers that have not reviewed any item). It also means that if the number of items is greater than the number of customers, the algorithm should return “Impossible,” due to every item not being reviewed (if |P| > |C|, then at least one customer in C should review more than one products, which cannot occur in this case).

By drawing the customer nodes, property nodes, product nodes and connecting them through whether customer can review certain property and the properties contained by an item through directed edges, the setup was designed to run the Ford-Fulkerson algorithm that returns the maximum amount of flow through the graph.

Since each customer can review at most one item, the flow from the source node to customer had the weight of 1 and since we wanted to check whether all items were reviewed, the flow from the items to the destination node had the weight of 1. In the middle (connect between |C| nodes, |X| nodes, |P| nodes, the weight was a very large number because it had to guaranteed that all nodes in the middle could reach the |P| nodes, not depending on whether one, two, or more customer was able to review items.

The number returned from the Ford-Fulkerson algorithm cannot be greater than |P| because each flow that |P| nodes send to the destination node has the weight of 1 so |P| \* 1 = |P|. If the number returned from the Ford-Fulkerson algorithm is less than |P|, this means that all items in the set P were not reviewed, which would mean that the system should return “Impossible.” If the number returned from the Ford-Fulkerson algorithm is equal to |P|, this means that all items in the set P were able to be reviewed by at least one customer. In this case, we can find a customer match for the items P, which I did above.

Runtime:

According to <https://stackoverflow.com/questions/33565995/time-complexity-of-the-ford-fulkerson-method-in-a-flow-network-with-unit-capacit>, the runtime of Ford-Fulkerson algorithm is O(m \* f) where f is the value of maximal flow and m is the number of edges. The total number of edges in this problem is |C| + |P| + total customer’s eligibility to review a property + total property of items. Therefore, the runtime is O((|C| + |P| + total customer’s eligibility to review a property + total property of items) \* f) where f is the value of maximal flow.