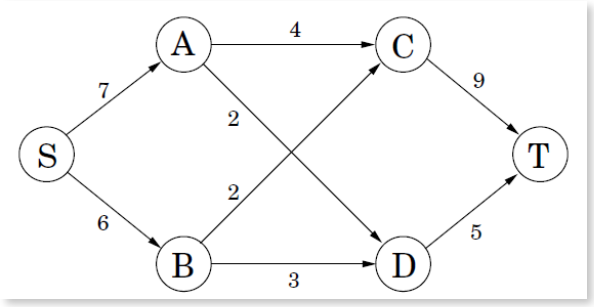
Isaac Delgado
4189182267

You are to solve problems 2, 3, and 4 using the following steps:

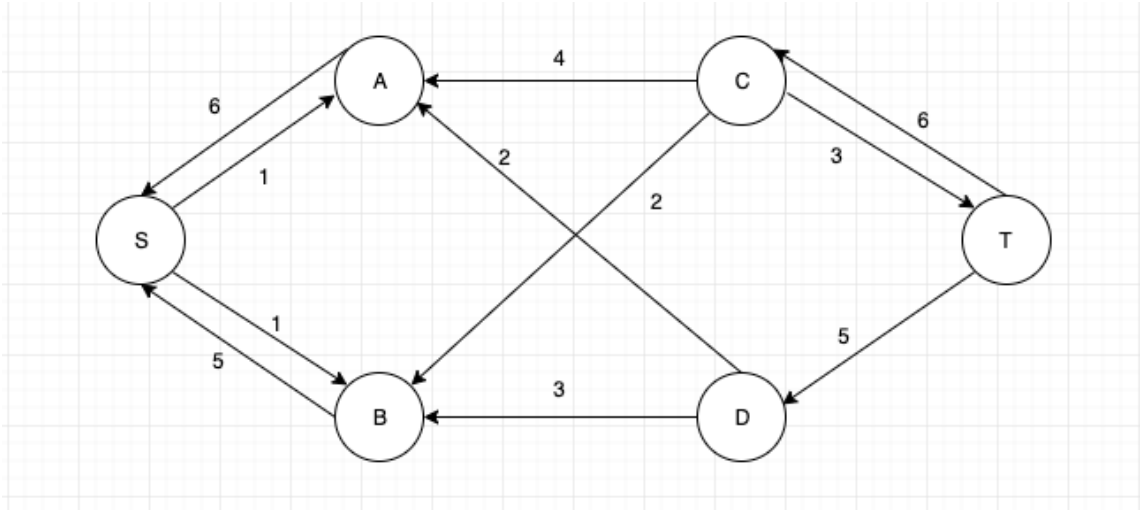
I. Describe how to construct a flow network
II. Make a claim. Something like "this problem has a feasible solution if and only if the max flow is…"
III. Prove the above claim in both directions

1. You are given the following graph *G*. Each edge is labeled with the capacity of that edge.



a) Find a max-flow in *G* using the Ford-Fulkerson algorithm. Draw the residual graph G_f corresponding to the max flow. You do not need to show all intermediate steps.

G_f =

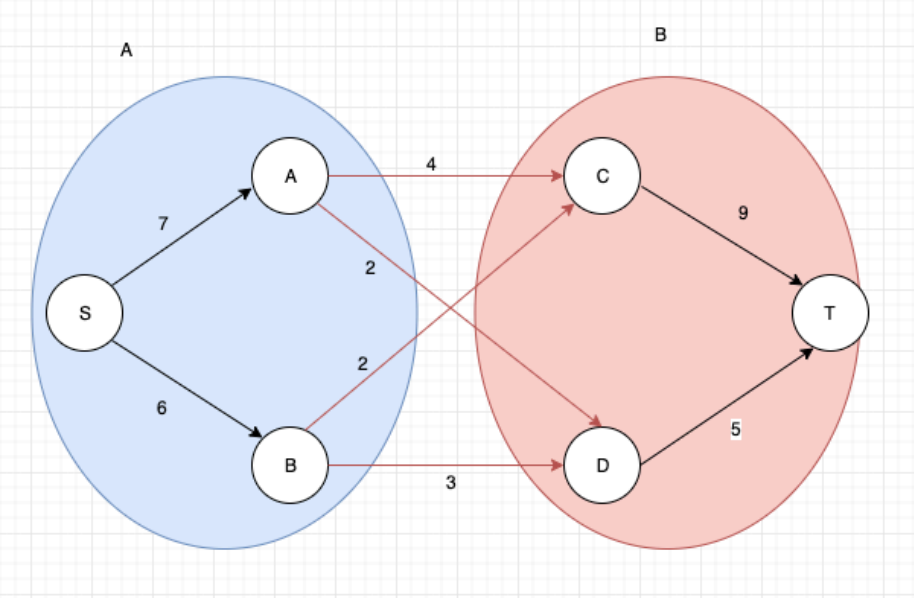


Max |f| = 4 + 2 + 2 + 3 = 11

Isaac Delgado
4189182267

b) Find the max-flow value and a min-cut.

Max-Flow = 11
Min-Cut = (Set A = {S, A, B}, B = {T, C, D}, Cap(A, B) = 4 + 2 + 2 +3 = 11



c) Prove or disprove that increasing the capacity of an edge that belongs to a min cut will always result in increasing the maximum flow.

Direct Proof:

By Theorem 1 (The Ford-Fulkerson alg. Outputs the max-|f|), any edge from set A to set B partition must be saturated and any edge from B to A must have flow value of 0. Thus flow value equals to capacity of (A, B) partition (cap(A, B)).

$$|f| = \sum_{u \in A, v \in B} f(u,v) - \sum_{u \in A, v \in B} f(v,u) = \sum_{u \in A, v \in B} f(u,v) - 0$$

The flow must be ≤ to the capacity of an edge that is is flowing through. (0 ≤ f(u, v) ≤ c(u, v))
Thus,

$$\sum_{u \in A, v \in B} f(u,v) \leq \sum_{u \in A, v \in B} cap(u,v)$$

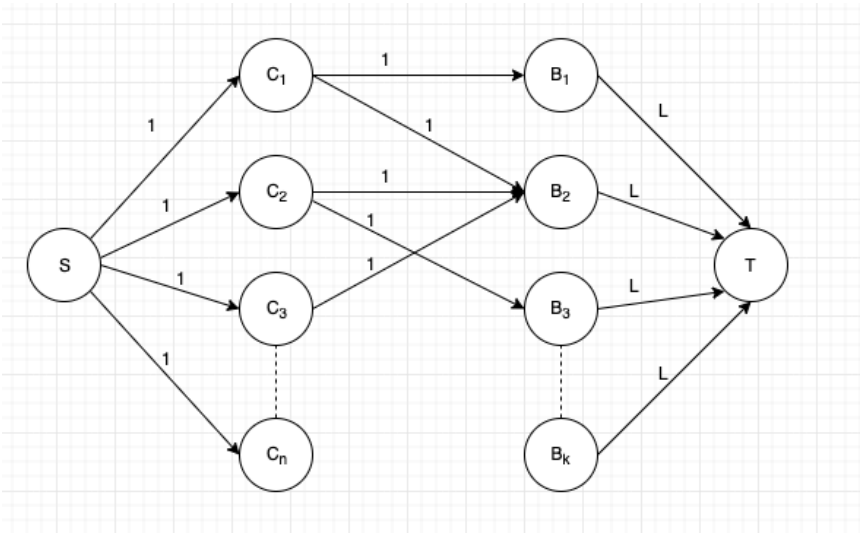
Hence the **max** flow is equal to the **full** capacity. (max(f(u, v)) = cap(A, B))

When we increase an edge from our min-cut we are increasing the value of cap(A, B). Then it must be the case that Max Flow is increased as well.

Isaac Delgado
4189182267

2. Consider a set of mobile computing clients in a certain town who each need to be connected to one of several possible *base stations*. We'll suppose there are *n* clients, with the position of each client specified by its (x, y) coordinates in the plane. There are also *k* base stations; the position of each of these is specified by (x, y) coordinates as well. For each client, we wish to connect it to exactly one of the base stations. Our choice of connections is constrained in the following ways. There is a *range parameter R* which means that a client can only be connected to a base station that is within distance *R*. There is also a *load parameter L* which means that no more than *L* clients can be connected to any single base station. Given the positions of a set of clients and a set of base stations, as well as the range and load parameters, decide whether every client can be connected simultaneously to a base station.

I. We can construct the network flow by the following;
   1. We will make a directed weighted graph and make all Mobile Clients and Base Stations individual vertices
   2. We will add 2 additional vertices, a source *s* and a sink *t*.
   3. Next we add edges connecting *s* to every client vertex and add a capacity of 1 to each of these edges
   4. We will add connected edges from base stations to the sink *t*. The capacity of these edges will be the load parameter for the base station thus they are all cap(b_i, *t*) = L
   5. Lastly we will connect all clients to base station vertices that are within R distance.



II. Claim;
   All clients can be connected simultaneously if and only if the Max Flow = n

Isaac Delgado
4189182267

III. Prove

(->) Given the scenario where all mobile clients are connected to a base station given the constraints then every edge leaving the source to a particular client must be saturated. There are n clients and thus n connecting edged from source to clients with weights 1, so the max flow must be equal to n

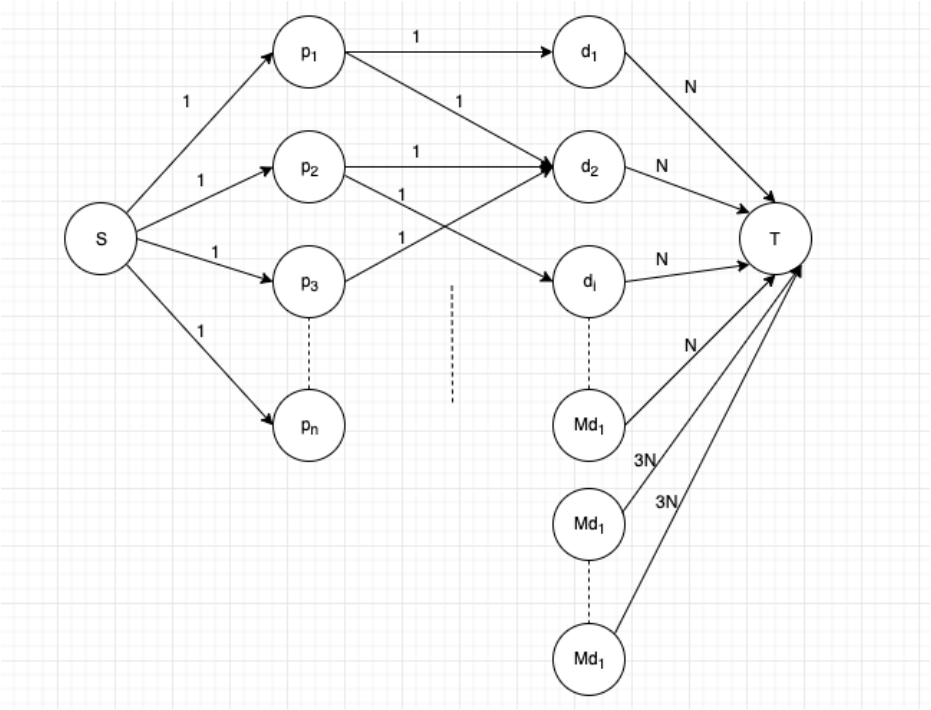
(<-) Given the the max flow, we can simply choose all saturated edges between client and base station nodes. A client can be connected to multiple bas stations but since there is only one edge connected the client to the source only on of the edges to a bas station will be saturated, we will consequently count n saturated edges if we have a max flow of n

3. Consider a drip irrigation system, which is an irrigation method that saves water and fertilizer by allowing water to drip slowly to the roots of plants. Assume that there are n plants. Suppose that the locations of all drippers are given to us in terms of their coordinates (x^d, y^d). Also, we are given locations of plants specified by their coordinates (x^p, y^p). A dripper can only provide water to plants within distance l. A single dripper can provide water to no more than n plants. However, we recently got some funding to upgrade our system with which we bought k monster drippers, which can provide water supply to three times the number of plants compared to standard drippers. So, we now have i standard drippers and k monster drippers. Given the locations of the plants and drippers, as well as the parameters l and n, decide whether every plant can be watered simultaneously by a dripper, subject to the above mentioned constraints. Justify carefully that your algorithm is correct and can be obtained in polynomial time.

I. We can construct the network flow by the following;

1. We will make a directed weighted graph and make Plants and Drippers (both standard and monster drippers) individual vertices.
2. We will add 2 additional vertices a source s and a sink t
3. Next we create edges connecting s to every plant vertex. The capacity for every edges going from source s to a plant vertex will be 1.
4. We next add connecting edges from every dripper to the sink t. The capacities of these edges will be how many plants the connected dripper can water, thus standard dripper = $cap(d_i, t) = n$, monster dripper = $cap(Md_i, t) = 3n$.
5. Lastly we will connect all plant vertices to all droppers that are within L distance. These edges will have a capacity of 1.

Isaac Delgado
4189182267



II. Claim;

We can water all plants simultaneously if and only if the Max Flow = n

III. Prove

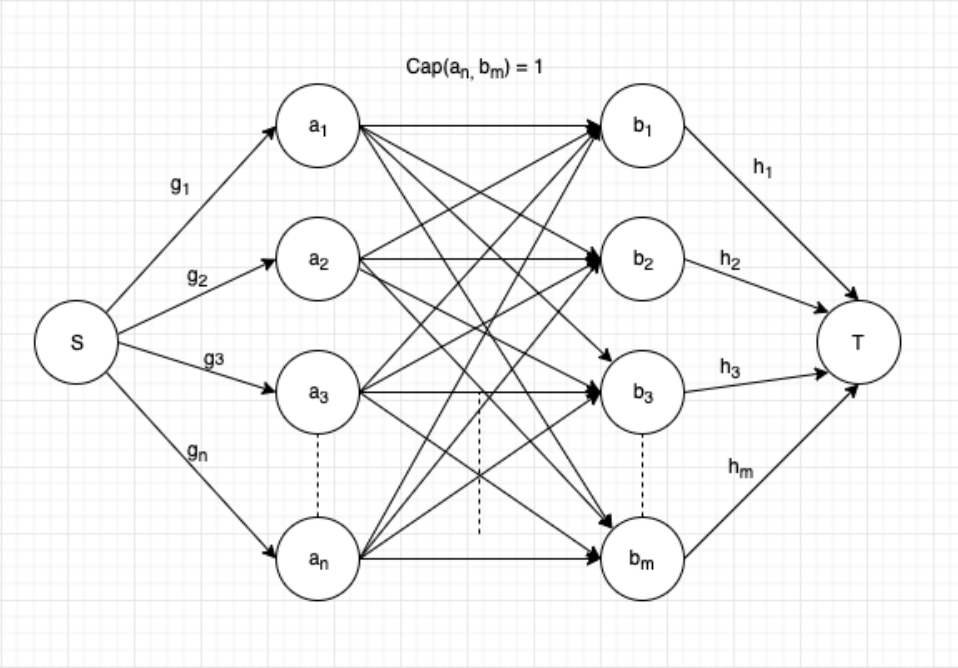
(->) Given the scenario where all plants are being watered at the same time, then as before all edges from the source to each plant must be saturated. Since there is exactly one edge connecting each plant from the source, and there are n plants then the max flow must be n

(<-) Given the the max flow, we can identify all the saturated edges between plant and dripper vertices. Similar to the problem before, a plant can be connected to multiple dippers but there is only a single edge connecting the source to a particular plant and thus there will only be one saturated edge between a plant and a dipper. So we can find the saturated edges to find the plant dipper matching.

Isaac Delgado
4189182267

4. At a dinner party, there are n families $a_1, a_2, ..., a_n$ and m tables $b_1, b_2, ..., b_m$. The i-th family a_i has g_i members and the j-th table b_j has h_j seats. Everyone is interested in making new friends and the dinner party planner wants to seat people such that no two members of the same family are seated at the same table. Design an algorithm that decides if there exists a seating assignment such that everyone is seated and no two members of the same family are seated at the same table. What would be a seating arrangement?

I. We can construct the network flow by the following;
   1. We will make a directed weighted graph and make Families and Tables individual vertices.
   2. We will add 2 additional vertices a source s and a sink t
   3. Next we create edges connecting s to every family vertex. The capacity for every edge going from source s to a family vertex will be the number of family members their connected family vertex has, cap(s ,a_i) = g_i.
   4. We next add connecting edges from every Table to the sink t. The capacities of these edges will be how many chairs the connected table can host, cap(b_i, t) = h_i
   5. Lastly we will connect every family vertex to every table vertex with a capacity of 1.



II. Claim;

We can seat all families where no two members of the same family are seated at the same table if an only if Max Flow = $\sum_{i=1}^{n} gi = g_1 + g_2 + g_3 + \ldots + g_n$

Isaac Delgado
4189182267

III. Prove

(->) Given the scenario where all family members are seated given the constraints defining in our problem then it must be the case that our edges leaving the source *s* are all saturated which will give us the max flow. Since every edge leaving the source is the number of family members for a given family then Mac flow is the summation of number of family members for all families.

(<-) Given the the max flow, we can deduce the seating arrangements by going through the partition between family and table vertices and adding up all saturated edges between them. This will give us the valid seating arrangement.