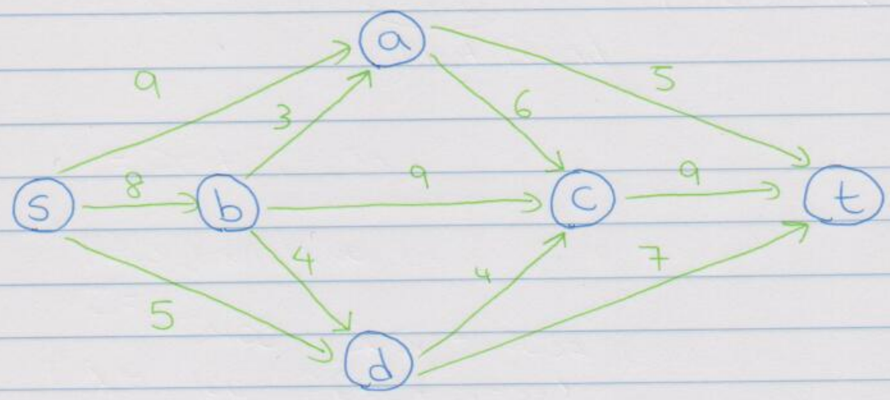
Ford - Fulkerson Examples

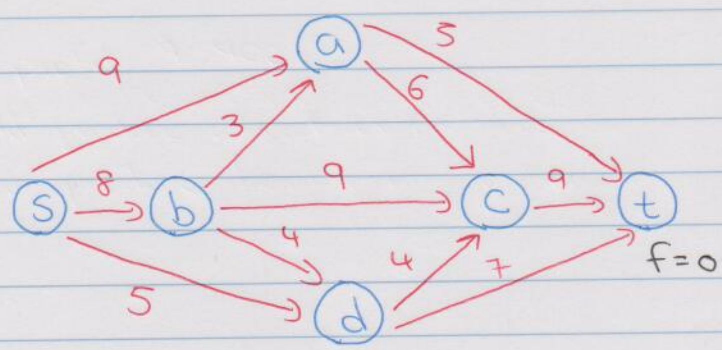
1. Consider the graph below



a) Use the F-F algo to compute the max flow.

Soln:

1. Get G_F, the residual graph

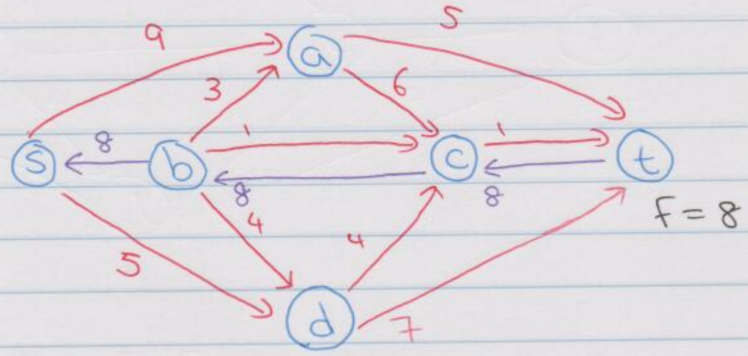


Red edges denote forward edges.

Purple edges denote reverse edges.

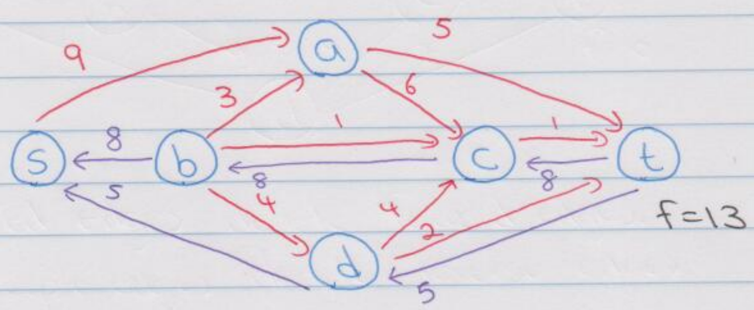
2. Choose a s-t path.
I'll choose $(s,b) \to (b,c) \to (c,t)$.
The bottleneck is 8.

New graph:



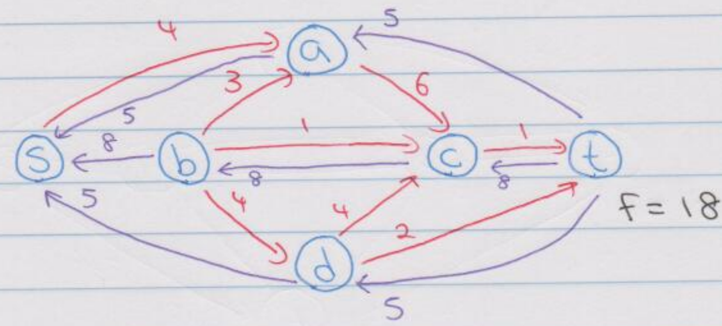
3. Choose a s-t path.
I'll choose $(s,d) \to (d,t)$
The bottleneck is 5.

New graph:



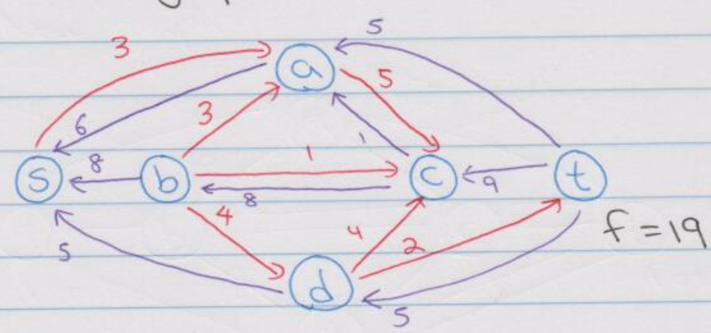
4. Choose a new s-t path.
I'll choose (s,a) → (a,t)
The bottleneck is 5.

New graph



5. Choose a new s-t path.
I'll choose (s,a) → (a,c) → (c,t)
The bottleneck is 1.

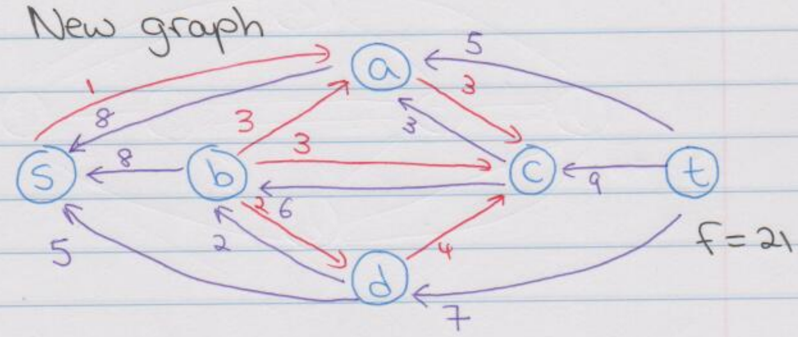
New graph:



6. Choose a new s-t path
I'll choose (s, a) → (a, c) → (c, b) → (b, d) → (d, t)

Note: Reverse edges are allowed.
The bottleneck is 2.

New graph



There are no more s-t paths anymore.
Hence, the max flow is 21.

b) Consider the cut $X_0 = \{s, b, c, d\}$ and $X_1 = \{a, t\}$.
Identify all forward and backward edges and the capacity of the cut.

Soln:
Forward edges: (s, a), (b, a), (c, t), and (d, t)
Backward edges: (a, c)
Note: We define the cut in the original graph.
Capacity: $9+3+9+7$
$= 28$

c) Find the min-cut in the network generated in part a.

Soln:
The set of nodes reachable from S is $\{s, a, b, c, d\}$.
Hence, the cut is $\{\{s, a, b, c, d\}, \{t\}\}$.
The min-cut's capacity is $5+9+7=21$.

2. (T/F) In any network G with int edge capacities, there always exists an edge e s.t. inc the capacity of e inc the max flow value in G.

Soln:
False. Max-flow = min ($\sum_{e \in E} cap(e)$ where e is an edge leaving S,

$\sum_{e \in E} cap(e)$ where e is an edge going into t

)

If the sums are equal, then inc the cap of any one edge won't do anything.

3. Suppose there are m courses, $C_1, \ldots, C_m$, and n profs: $P_1, \ldots, P_n$. For each $j \in \{1, \ldots, m\}$, course C_j has S_j sections. For each $i \in \{1, \ldots, n\}$, prof P_i has a teaching load of l_i and likes to teach a subset of courses $A_i \subseteq \{C_1, \ldots, C_m\}$.

Design a network flow problem to generate an assignment of profs to courses satisfying the following constraints or reports no such assignment exists.

Constraints:

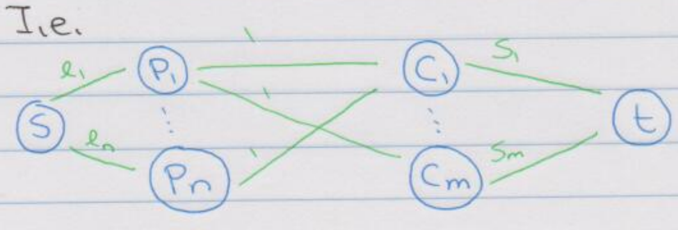
1. Each prof P_i must be assigned to exactly l_i courses.
2. Each course C_j must be assigned to exactly S_j profs.
3. No prof should teach a course they don't like.
4. No prof should teach multiple sections of the same course.

a) Describe your full algorithm.

Soln:

- Create a start node, s, and an end node t.
- Make each prof a node.
- Make each course a node.
- Connect S with each prof node s.t. the capacity is l_i.
- Connect each course node with t s.t. the capacity is S_j.
- Connect a prof node with a course node only if the course is liked by the prof. Set the capacity of each edge to 1.

I.e.



Algo:
1. Create the graph as described.
2. Run the F-F algo.
3. Check to see if there are any edges going from a course node to t in the residual graph.
   If there are, then report no such assignment can be made.
   Otherwise, return the assignment.

b) Prove that the algo is correct.

Soln:

Flow → Assignment

Consider a fully saturated flow, meaning that each edge (s, P_i) has a flow of l_i and each edge (C_i, t) has a flow of S_i.
Furthermore, we know that the capacity of any prof to course edge is 1.
Hence, each prof will teach exactly 1 section of a course, will teach only courses they like, have them teach exactly l_i courses and have exactly S_i profs teaching course i.

Assignment → Flow

Consider any valid assignment. This means that each prof teaches exactly l_i courses, teaches only courses they like, doesn't teach more than 1 section of any course and each course has exactly S_i profs.
Hence, we get a saturated flow.