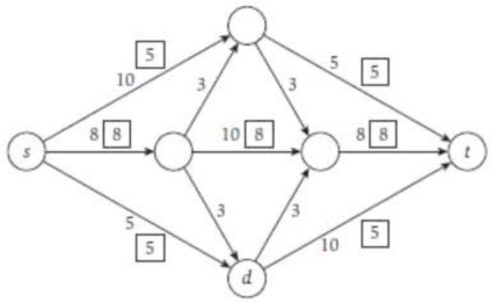
CS 577 # Assignment 9 – Network Flow Fall 2022

Answer the questions in the boxes provided on the question sheets. If you run out of room for an answer, add a page to the end of the document.

Name: Carson Miller Wisc id: 9081473028

Network Flow

1. *Kleinberg, Jon. Algorithm Design (p. 415, q. 3a)* The figure below shows a flow network on which an $s-t$ flow has been computed. The capacity of each edge appears as a label next to the edge, and the flow is shown in boxes next to each edge. An edge with no box has no flow being sent down it.

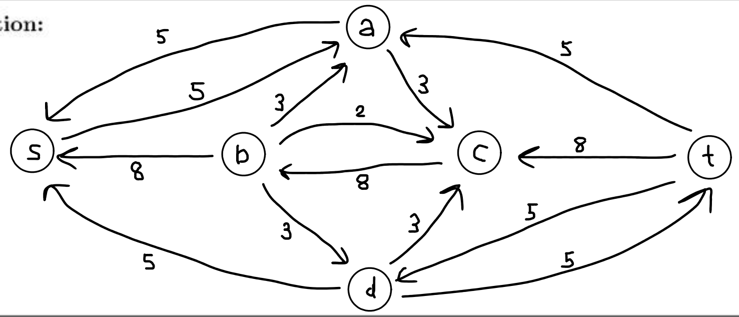


(a) What is the value of this flow?

Solution: 18

(b) Please draw the **residual graph** associated with this flow.

Solution:



(c) Is this a maximum $s-t$ flow in this graph? If not, describe an augmenting path that would increase the total flow.

Solution: No, an augmenting path s -> a -> c -> b -> d -> t would increase the total flow

2. *Kleinberg, Jon. Algorithm Design (p. 419, q. 10)* Suppose you are given a directed graph $G = (V, E)$. This graph has a positive integer capacity c_e on each edge, a source $s \in V$, a sink $t \in V$. You are also given a maximum $s - t$ flow through G: f. You know that this flow is *acyclic* (no cycles with positive flow all the way around the cycle), and every flow $f_e \in f$ has an integer value.

   Now suppose we pick an edge e^* and reduce its capacity by 1 unit. Show how to find a maximum flow in the resulting graph G^* in time $O(m + n)$, where $n = |V|$ and $m = |E|$.

Solution:

We accomplished this in lecture in Theorem 9. You can run BFS or DFS from s on the resulting graph G* in time O(m + n) to find A*. You can then find B* in O(n) time as it is V \ A*. Knowing A* and B* will give us the min-cut, which is equivalent to the max-flow.

3. *Kleinberg, Jon. Algorithm Design (p. 420, q. 11)* A friend of yours has written a very fast piece of code to calculate the maximum flow based on repeatedly finding augmenting paths. However, you realize that it's not always finding the maximum flow. Your friend never wrote the part of the algorithm that uses backward edges! So their program finds only augmenting paths that include all forward edges, and halts when no more such augmenting paths remain. (Note: We haven't specified *how* the algorithm selects forward-only augmenting paths.)

   When confronted, your friend claims that their algorithm may not produce the maximum flow every time, but it is guaranteed to produce flow which is within a factor of b of maximum. That is, there is some constant b such that no matter what input you come up with, their algorithm will produce flow at least $1/b$ times the maximum possible on that input.

   Is your friend right? Provide a proof supporting your choice.

Solution:

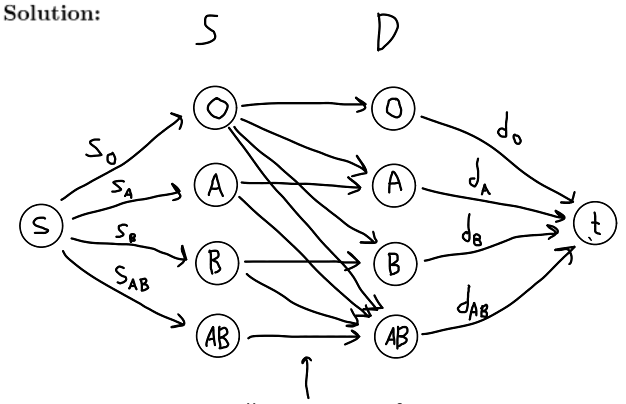
Yes, my friend is right. Forward-only augmenting paths will yield some flow if any flow is to be had. Thus, the resulting max-flow from only using forward paths will always be at least some fraction 1/b of the maximum possible input. If forward-only paths result in a max-flow of 0, then adding in the ability to use backwards paths will not increase that max-flow.

4. *Kleinberg, Jon. Algorithm Design (p. 418, q. 8)* Consider this problem faced by a hospital that is trying to evaluate whether its blood supply is sufficient:

   In a (simplified) model, the patients each have blood of one of four types: A, B, AB, or O. Blood type A has the A antigen, type B has the B antigen, AB has both, and O has neither. Patients with blood type A can receive either A or O blood. Likewise patients with type B can receive either B or O type blood. Patients with type O can only receive type O blood, and patients with type AB can receive any of the four types.

   (a) Let integers s_O, s_A, s_B, s_{AB} denote the hospital's blood supply on hand, and let integers d_A, d_B, d_O, d_{AB} denote their projected demand for the coming week. Give a polynomial time algorithm to evaluate whether the blood supply is enough to cover the projected need.

Solution:



All capacities infinite

This problem can be represented by the network flow problem on the left. The answer to the question comes from whether or not the flow of edges entering the sink (t) are equal to the capacities of those edges. If they are, then the supply is enough to cover the need. If not, then there is not enough supply to cover the need.

   (b) Network flow is one of the most powerful and versatile tools in the algorithms toolbox, but it can be difficult to explain to people who don't know algorithms. Consider the following instance. Show that the supply is **insufficient** in this case, and provide an explanation for this fact that would be understandable to a non-computer scientist. (For example: to a hospital administrator.) Your explanation should not involve the words *flow*, *cut*, or *graph*.

blood type	supply	demand
O	50	45
A	36	42
B	11	8
AB	8	3

Solution:
We should start with filling the most difficult demand, blood type O. Patients of blood type O can only receive blood of type O, so fill that demand first. Now we have 41 supply for A (remaining O plus 36 A) and 16 supply for B (remaining O plus 11 B). Because the demand for A is greater than the supply, we can deem our supply insufficient in this case. The only way to increase the supply to meet the demand for A would be to not meet the demand for O and save some of that supply, which would also lead to the conclusion that the supply is insufficient in this case.

5. Implement the Ford-Fulkerson method for finding maximum flow in graphs with only integer edge capacities, in either C, C++, C#, Java, or Python. Be efficient and implement it in $O(mF)$ time, where m is the number of edges in the graph and F is the value of the maximum flow in the graph. We suggest using BFS or DFS to find augmenting paths. (You may be able to do better than this.)

   The input will start with a positive integer, giving the number of instances that follow. For each instance, there will be two positive integers, indicating the number of nodes $n = |V|$ in the graph and the number of edges $|E|$ in the graph. Following this, there will be $|E|$ additional lines describing the edges. Each edge line consists of a number indicating the source node, a number indicating the destination node, and a capacity. The nodes are not listed separately, but are numbered $\{1 \ldots n\}$.

   Your program should compute the maximum flow value from node 1 to node n in each given graph.

   A sample input is the following:

```
2
3 2
2 3 4
1 2 5
6 9
1 2 9
1 3 4
2 4 1
2 5 6
3 4 4
3 5 5
4 6 8
5 6 5
5 6 3
```

   The sample input has two instances. For each instance, your program should output the maximum flow on a separate line. Each output line should be terminated by a newline. The correct output for the sample input would be:

```
4
11
```