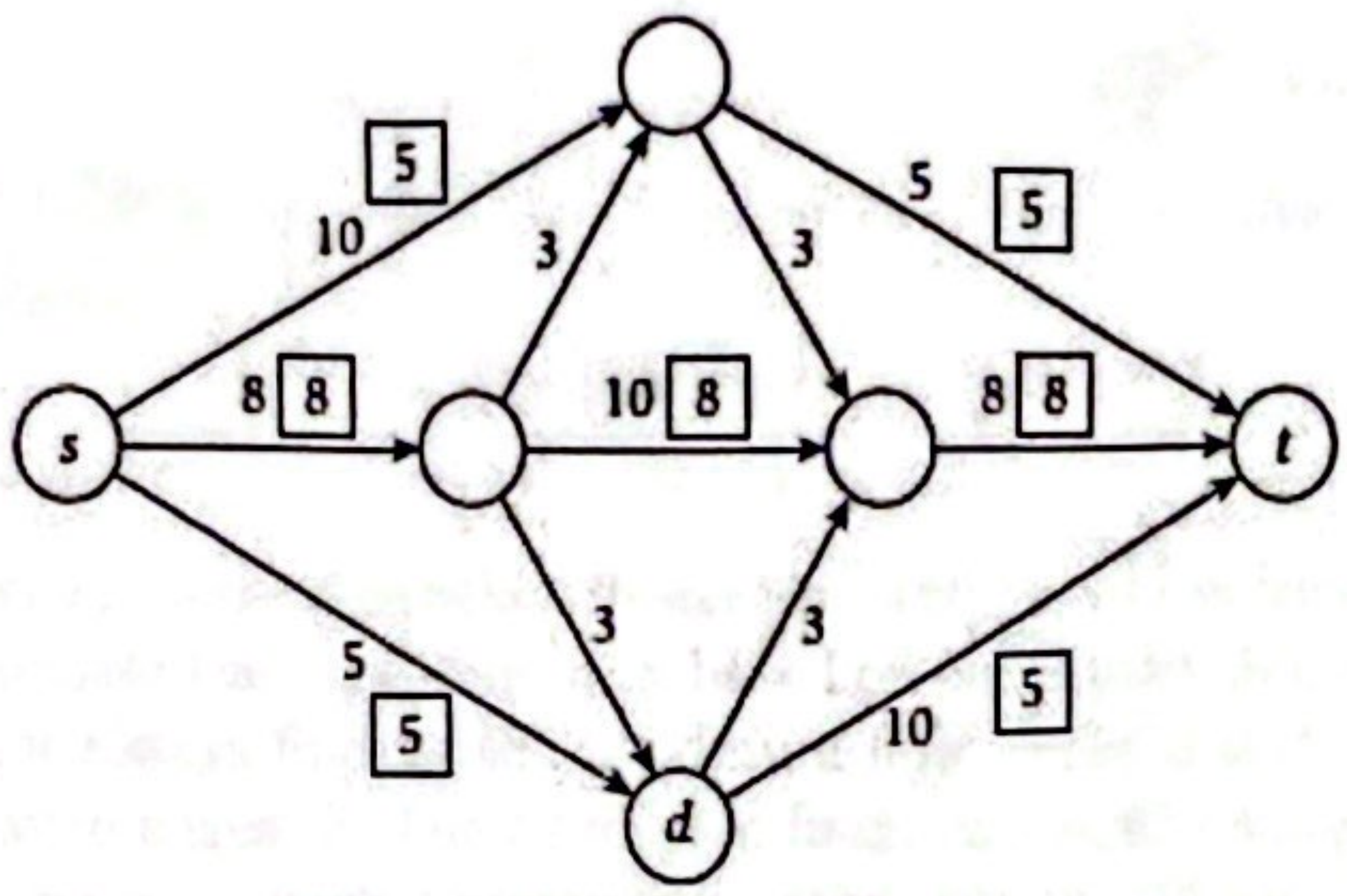
CS 577 **Assignment 9 – Network Flow** Fall 2023

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: Ethan Yan Wisc id: 9084649137

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.

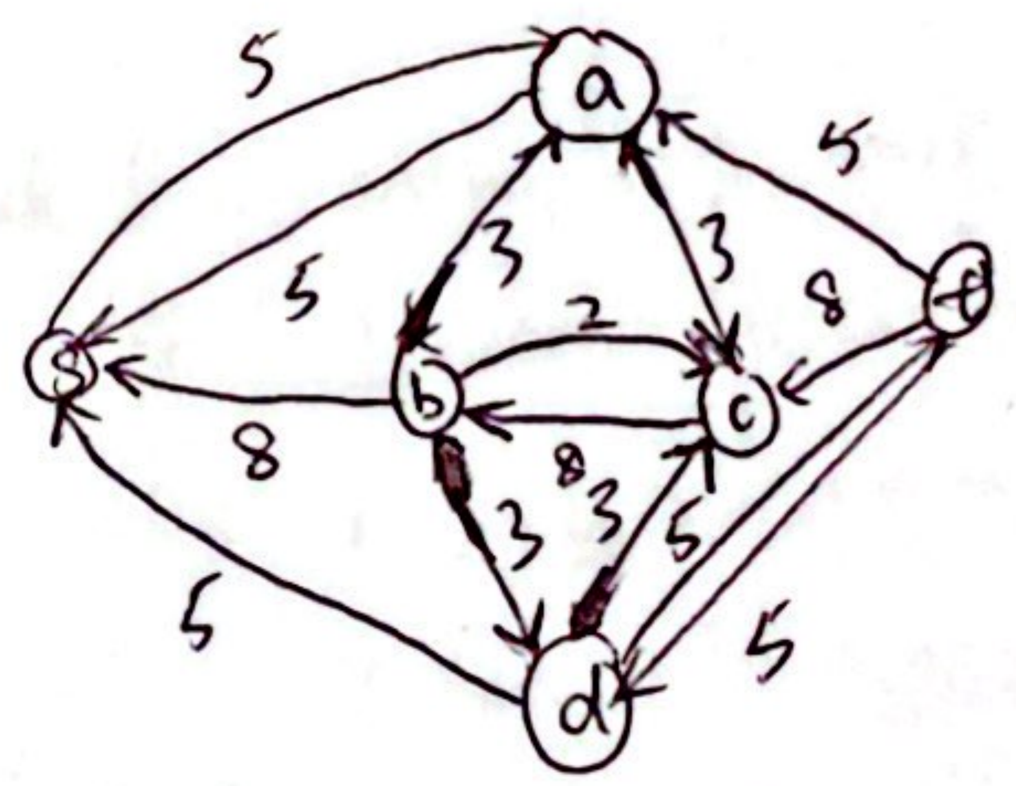


(a) What is the value of this flow?

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: s, a, c, b, d, t , flow is 3.

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 $m = |E|$.

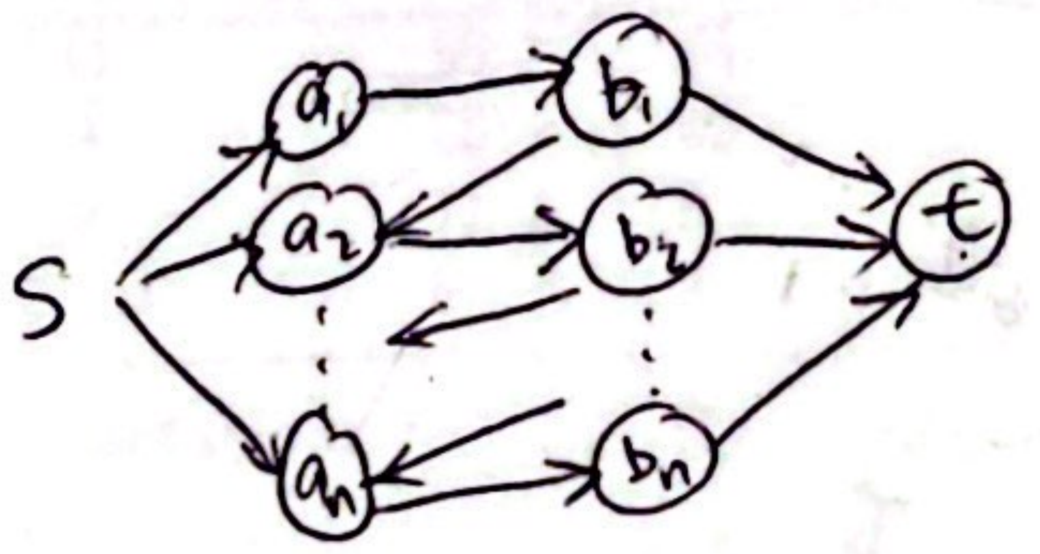
Solution: First generate residual graph of G_f, then since all edges are positive integers then we know that it would not be negative after reduction of 1. ~~R~~ Generating residual graph is $O(m+n)$ as well.
We can then perform BFS on the ~~graph~~ residual graph $O(m+n)$, whilst updating flow along the path $O(n)$ before building a new G^* $O(m)$. Thus overall runtime is $O(m+n)$.
BFS first on node going back to s, using backward edges, then BFS on s forward to t.

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: No. Counter example:



all edges has capacity 1.

Each Augmenting Path would have flow of 1, however, ideally the max flow is n, by making use all all n edges from s to $a_1, a_2, \dots, a_n$ to $b_1, b_2, \dots, b_n$ to t.
If longest augmenting path used, i.e. $s, a_1, b_1, a_2, b_2, \dots, a_n, b_n, t$, then max flow is 1 in this case & not n, thus not optimal.

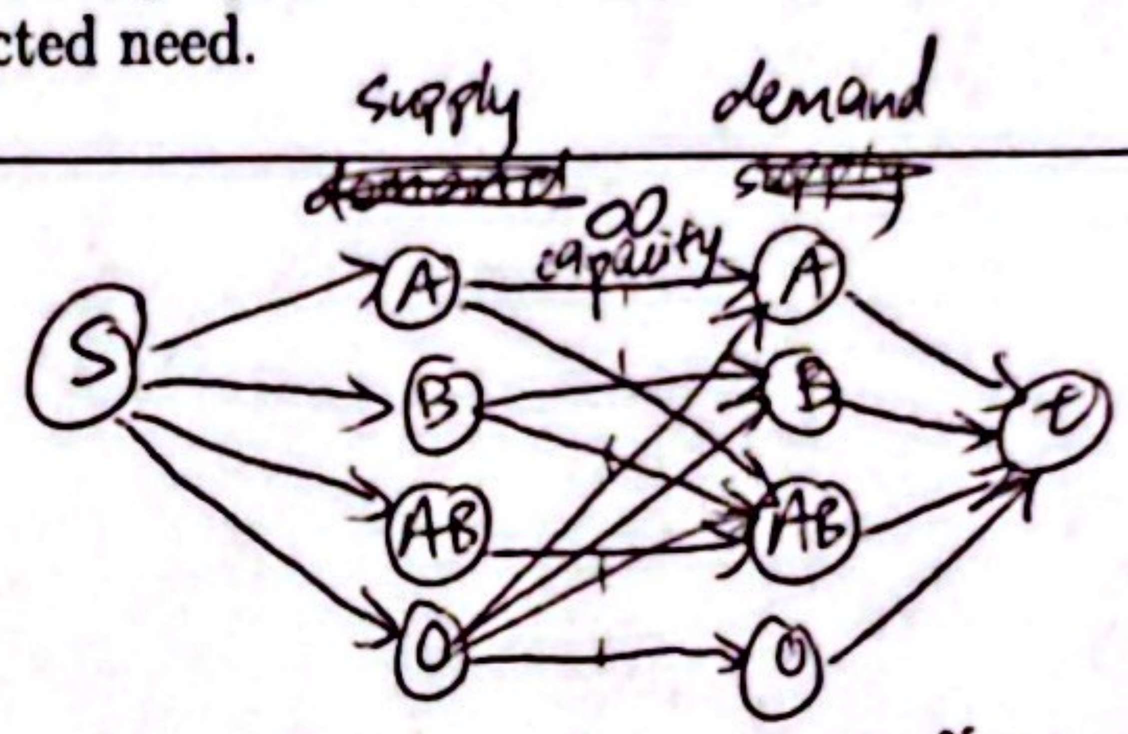
4. *Kleinberg, Jon. Algorithm Design (p. 418, q. 8)* **Consider this problem faced by a hospital that is trying to evaluate whether its blood supply is sufficient:**

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.

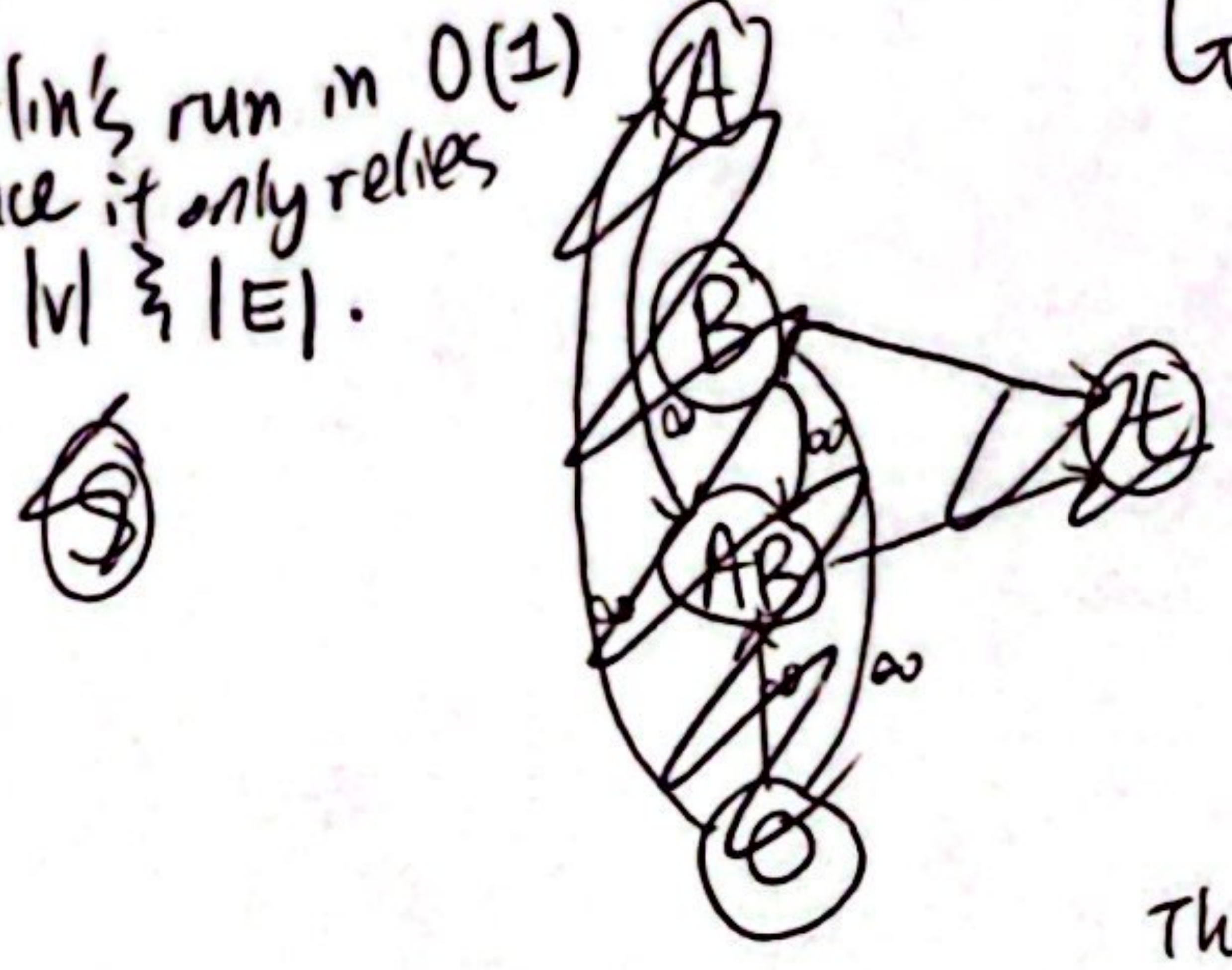
(a) **Let integers s_O, s_A, s_B, s_{AB} denote the hospital's blood supply on hand, and let integers d_A, d_B, d_O, d_{AB} denote their projected demand for the coming week. Give a polynomial time algorithm to evaluate whether the blood supply is enough to cover the projected need.**

Solution:

G:



Orlin's run in O(1) since it only relies on |V| & |E|.



supply is sufficient if flow from demand nodes to t is ~~&~~ equal to max flow through G

There is fixed # of nodes & edges so, $|V|=O(1)$ and $|E|=O(1)$

Ford-Fulkerson runs in $O(|E||f^*|)$ and since $|E|=O(1)$ & $O(|f^*|)$ is also constant. we know in this case, max flow is equal to min cut $= d_A+d_B+d_O+d_{AB}$ so we know algorithm runs in ~~polynomial~~ linear time with respect to sum of demand & sum of supply w similar logic

(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: Type A ~~to~~ needs 6 more to meet demand, A can alternatively receive type O. Type O can only spare 5 to still meet demand. Nothing can make up for blood O can receive since type O can only receive type O. so A is still lacking 1, thus supply is insufficient. Both B & AB sufficient but cannot supply to O or A.