Homework 4 Solutions IEORE4004 (Optimization M&M)
Instructor: Christian Kroer

Problem 1

1.
Consider the following edge-list for the problem:
$(s, v_2), (s, v_1), (v_2, v_1), (v_1, v_2), (v_2, v_3), (v_3, t), (v_2, t), (v_1, t)$
For this set of edges you get the following table of distances and predecessors after 1 iteration:

Node	d	p
s	0	-
v_1	~~∞~~ ~~4~~ 0	~~s~~ v_2
v_2	~~∞~~ 2	s
v_3	~~∞~~ 4	v_2
t	~~∞~~ 0	v_3

On doing another iteration it can be checked that these values remain unchanged and the algorithm terminates. Therefore, the shortest path from s to t is $s \to v_2 \to v_3 \to t$ with distance 0.

2.
The final table of d and p contains all information of the shortest path from s to any node in a graph (the path and its distance). The shortest path from s to t that we obtain from the final tables will be $s \to v_2 \to v_3 \to t$. This path contains v_2. Therefore, the shortest path from v_2 to t must be $v_2 \to v_3 \to t$ with distance $d(t) - d(v_2) = -2$. This is because, if there existed a shorter path from v_2 to t, then path from s to t represented in the final table could not be the shortest path, as we could replace the path between v_2 and t with the shortest path from v_2 to t and get a path with smaller distance.

The shortest path from s to v_3 is $s \to v_2 \to v_3$. This path does not contain v_1. Thus, we have no information on the shortest path between v_1 and v_3.

Problem 2

Project Network:

$$\begin{tikzcd}
 & & & C \arrow[drr, "3"] & & \\
s \arrow[r, "5"] & A \arrow[r, "8"] & B \arrow[ur, "10"] \arrow[r, "5"] \arrow[dr, "4"'] & D & F \arrow[r, "3"'] & G \\
 & & & E \arrow[ur, "6"] & &
\end{tikzcd}$$

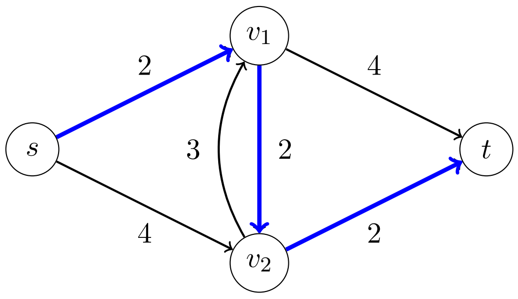
Critical path:
To get the critical path, we use the label correction method with negative weights. We consider the edge-list: $(s, A), (A, B), (B, C), (B, D), (B, E), (E, F), (C, G), (F, G)$

Node	d	p
s	0	-
A	5	s
B	13	A
C	23	B
D	18	B
E	17	B
F	23	E
G	26	C

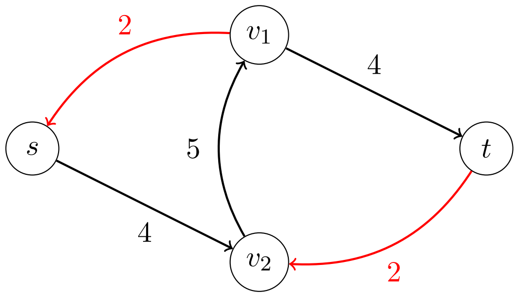
We get the critical path from s to G to be $s \to A \to B \to C \to G$ using this table, with total days taken being 26.

Problem 3

1. First we consider the augmenting path $s \to v_1 \to v_2 \to t$ from s to t.

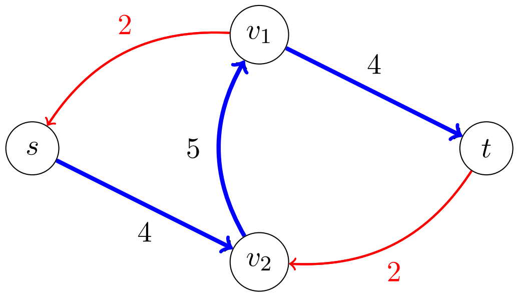


   The minimum capacity on the arcs is 2, so we send a flow of 2 units along this path. And then we create the residual network:

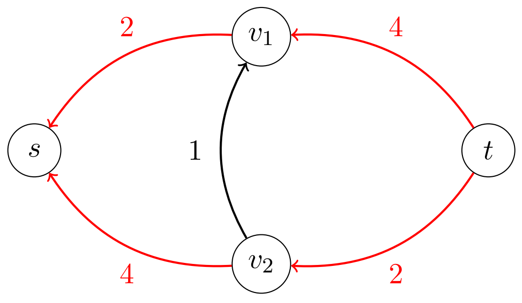


   Note that there already exists an edge from v_2 to v_1 so there is no need to add another residual arc from v_2 to v_1, instead we can update the capacity of the existing arc itself.

2. In the residual network, we consider the augmenting path $s \to v_2 \to v_1 \to t$ from s to t.



The minimum capacity on the arcs is 4, so we send a flow of 4 units along this path and create the residual network:

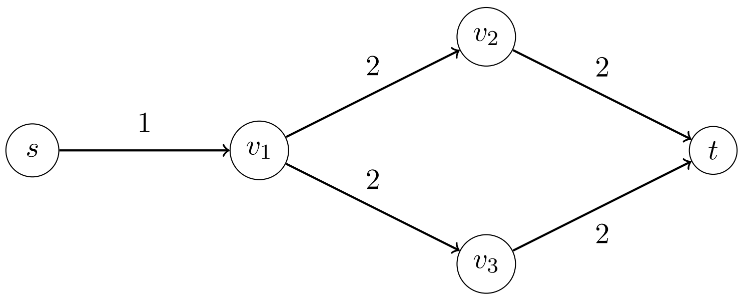


3. No more augmenting paths exist from s to t. Therefore, we terminate the algorithm, with Max-Flow $= 2 + 4 = 6$.

4. For Min-Cut, no nodes are reachable from s in the final residual network, so consider the cut $\{s\}$. This is a cut with capacity 6, hence, it is a Min-Cut.

Problem 4

Consider the simple counter-example:



The maximum value for an $s - t$ flow in this example is 1. However two flows are possible: (1) 1 unit of flow along $s \to v_1 \to v_2 \to t$ and, (2) 1 unit of flow along $s \to v_1 \to v_3 \to t$. However, the $s - t$ cut of minimum capacity is unique, and that is $\{s\}$ with capacity 1.

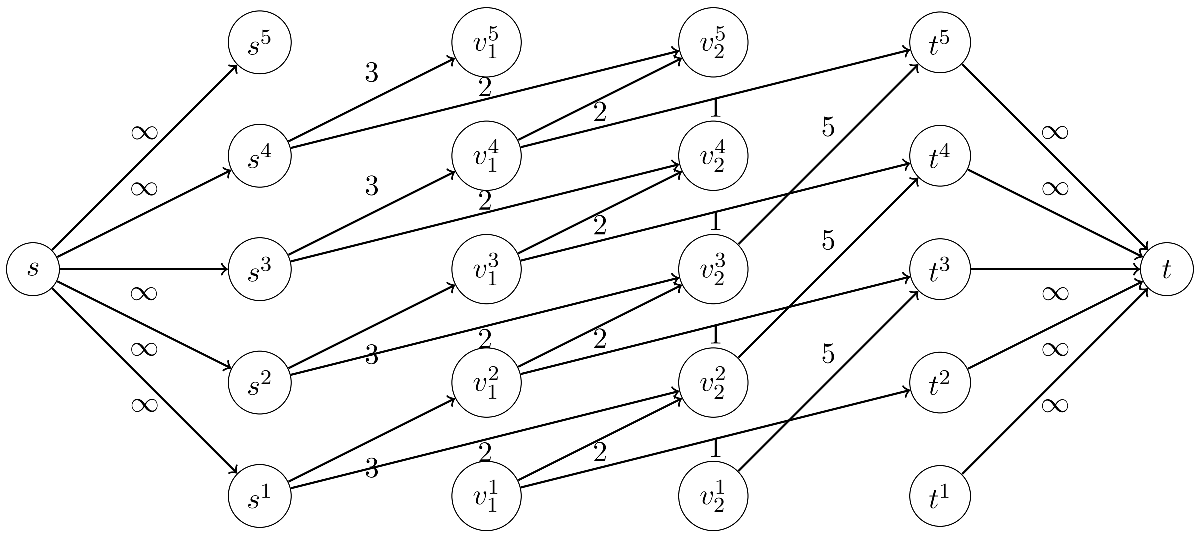
Therefore, it is possible to have multiple maximum $s - t$ flows but a unique minimum $s - t$ cut. Therefore, we cannot deduce that a graph has more than one $s - t$ cut of minimum capacity if it has more than one $s - t$ flow of maximum value.

Problem 5

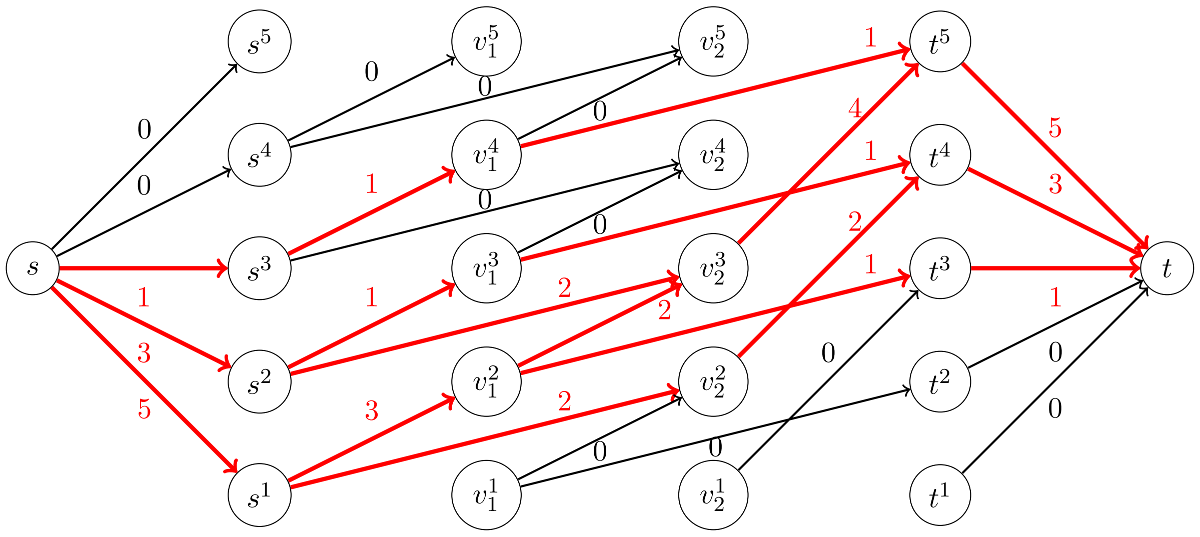
Consider a path from s to t with minimum capacity $\epsilon_0 > 0$. Suppose we only augment by $\epsilon/2$ in that iteration. Then in the next iteration, when we look at the residual network, the same path will still remain an augmenting path from s to t, but the new minimum capacity ϵ_1 becomes $\epsilon_0/2$, i.e. $\epsilon_1 = \epsilon_0/2$. In this iteration, we will augment the path by $\epsilon_1/2 = \epsilon_0/4$, and once again, the same augmenting path will still exist in the residual network with minimum capacity $\epsilon_2 = \epsilon_0/4$. Thus, we can see that after n iterations, for any $n > 0$, this augmenting path will still always remain in the residual network with minimum capacity $\epsilon_n = \epsilon_0/2^n$. Therefore, this algorithm will never terminate.

Problem 6

1. The following is the time-expanded network fro the problem, with capacities.



2. The following is the maximum flow solution you get using Ford and Fulkerson's algorithm. The maximum flow is 9 units.



3. At time t=3.7, 2 units dispatched at t=2 and 4 units dispatched at t=3 are traversing arc (v_2, t). Therefore, a total of 6 units of flow is traversing (v_2, t) at time t=3.7 in our solution.

Problem 7

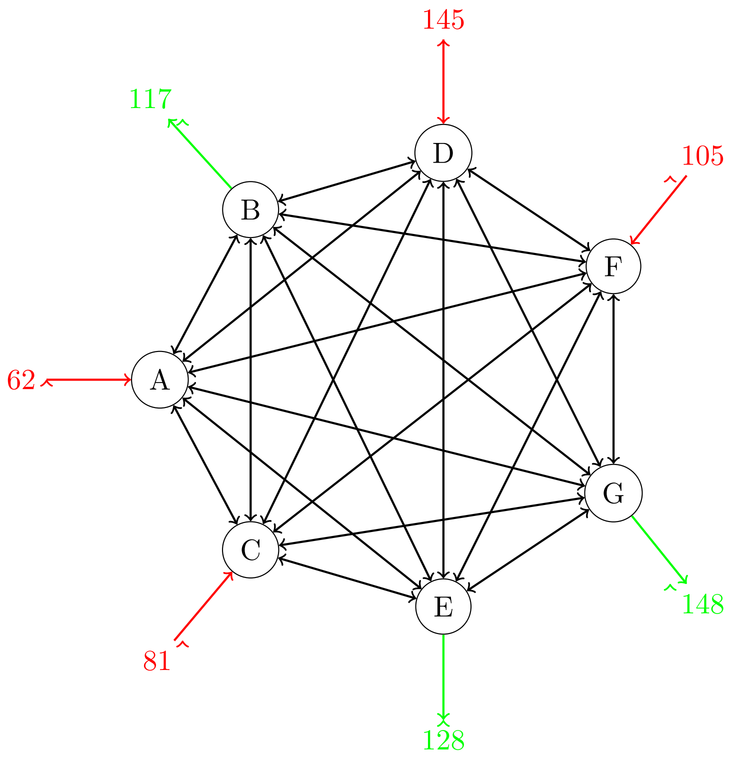
1. Motivation:
   A naive solution to this problem would be each bank i paying off whatever it owes to every bank j directly. But we can do better than that, because in the end, all that would

really matter to a bank is that it gets back the total amount owed to it. This allows us to model the problem as a min-cost problem. To find how much each bank should gain or lose, we can first complete the table given in the question as follows and then add up the values along the rows to find the total amount each bank should lose/gain (in thousands of USD):

Bank	A	B	C	D	E	F	G	Total
A	0	40	30	-120	12	60	40	62
B	-40	0	30	-70	15	-40	-12	-117
C	-30	-30	0	90	11	-20	60	81
D	120	70	-90	0	40	-15	20	145
E	-12	-15	-11	-40	0	-20	-30	-128
F	-60	40	20	15	20	0	70	105
G	-40	12	-60	-20	30	-70	0	-148

Note that the sum of the entries in the last column is 0 as expected, so it is possible to formulate this as a min-cost flow problem.

We design a network with each bank being a node. Every bank can pay any other bank; for every pair of banks i and j, there should an arc from i to j, as well as an arc from j to i. If 1000 USD is sent from one bank to another, the transmission cost will be 0.003 USD $\times$ 1000 = 3 USD. Thus, the cost of each arc will be 3 USD. This gives us the following graph:



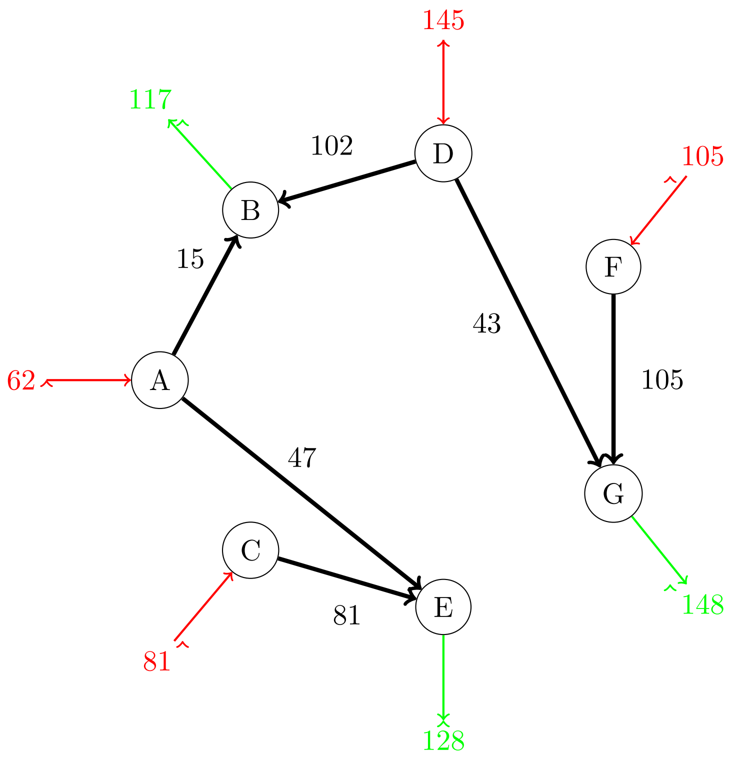
Each arc in the graph has a cost of 3. Supplies are shown in red and demands are shown in green. Total supply and demand are both 393,000 USD.

2. LP:
A standard LP formulation of the min-cost flow problem given in part 1 works. Some of you might have used the following formulation with free variables. **Please note that the objective function with free variables is a summation over absolute values. As a result, the objective function is no longer linear, and the optimization program is not a linear program.** One out of five points will be deducted if you submitted a formulation like the one below.

Let x_{AB} be the flow from bank A to bank B, this flow can be positive or negative, so we treat it as a free variable. Similarly we create variables x_{ij} for every pair of banks, $x_{ij} > 0$ denotes that bank i pays bank j whereas $x_{ij} < 0$ denotes that bank j pays bank i.

$$
\begin{array}{ll}
\min & 3\sum_{i,j} |x_{ij}| \\
\text{s.t.} & \\
& \begin{array}{lllllllll}
x_{AB} & +x_{AC} & +x_{AD} & +x_{AE} & +x_{AF} & +x_{AG} & = & 62 \\
-x_{AB} & +x_{BC} & +x_{BD} & +x_{BE} & +x_{BF} & +x_{BG} & = & -117 \\
-x_{AC} & -x_{BC} & +x_{CD} & +x_{CE} & +x_{CF} & +x_{CG} & = & 81 \\
-x_{AD} & -x_{BD} & -x_{CD} & +x_{DE} & +x_{DF} & +x_{DG} & = & 145 \\
-x_{AE} & -x_{BE} & -x_{CE} & -x_{DE} & +x_{EF} & +x_{EG} & = & -128 \\
-x_{AF} & -x_{BF} & -x_{CF} & -x_{DF} & -x_{EF} & +x_{FG} & = & 105 \\
-x_{AG} & -x_{BG} & -x_{CG} & -x_{DG} & -x_{EG} & -x_{FG} & = & -148
\end{array}
\end{array}
$$

An optimal solution is bank C pays E 81 thousands, A pays E 47 thousands, A pays B 15 thousands, D pays B 102 thousands, D pays G 43 thousands, F pays G 105 thousands, and the optimal value is 1179 USD. The following is a graphical representation of the optimal solution:


145
117
102
D
105
B
15
F
43
62
A
105
47
G
C
148
E
81
81
128