CSCI 570 Summer 2023

Exam II Review Slides

Dynamic Programming

Refer to the previous review, and problems from HW & Exam I for solutions and feedback

Network Flow

Xiyang Zhang

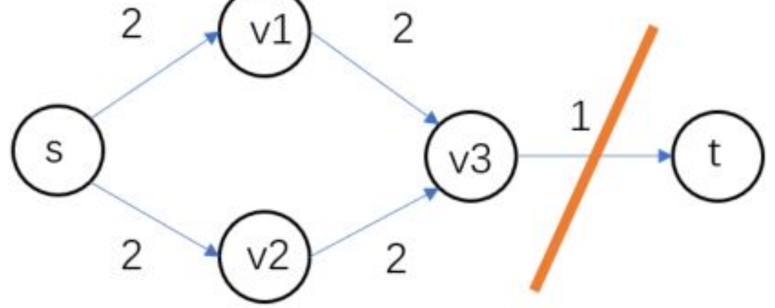
Warmup - True or False?

1. If every edge capacity in a flow network is an integer multiple of 42, then the value of the maximum flow will also be a multiple of 42.

   True!

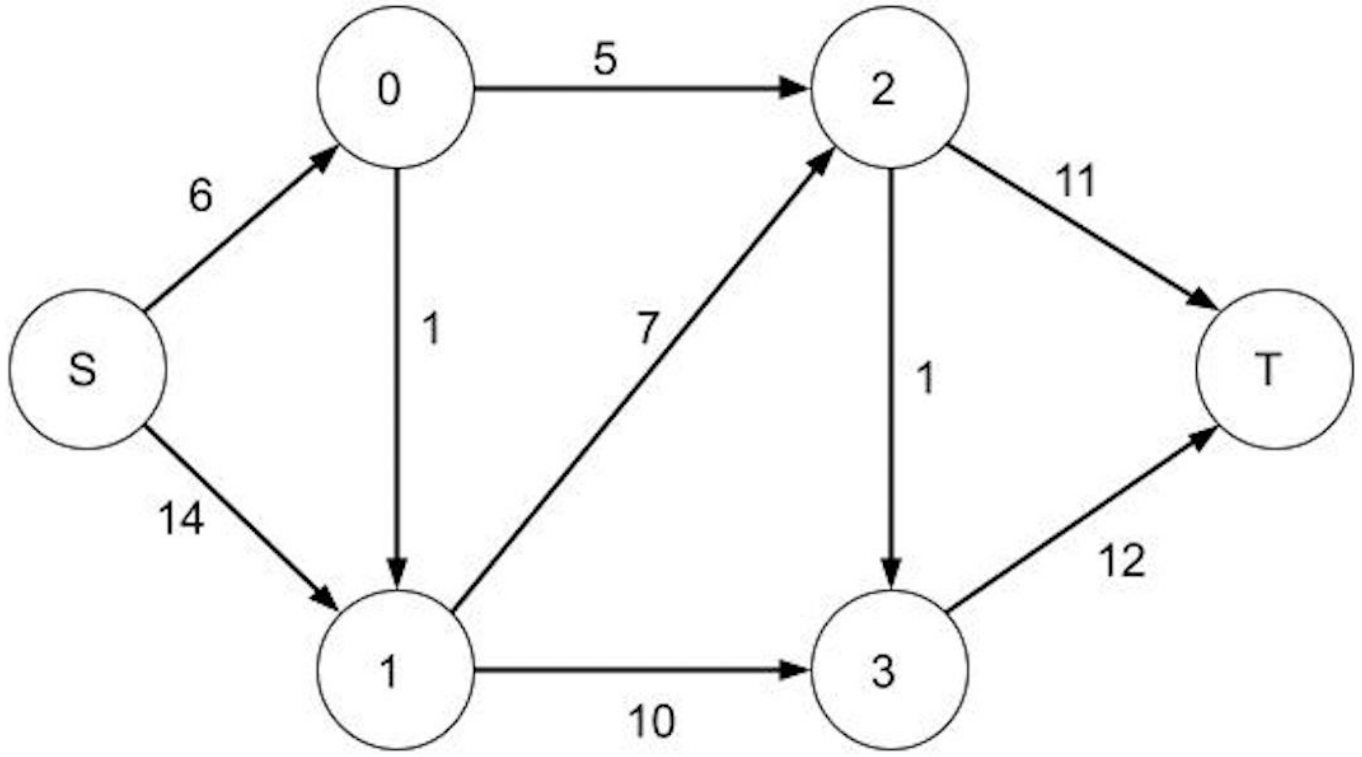
2. Let (S, V−S) be a minimum (s,t)-cut in the flow network G. Let (u, v) be an edge that crosses the cut in the forward direction, i.e., u ∈ S and v ∈ V−S. If we increase the capacity of (u, v) by any integer k > 1, then this edge will no longer be crossing a min-cut.

   False!



Question 1 - Scaled Ford-Fulkerson

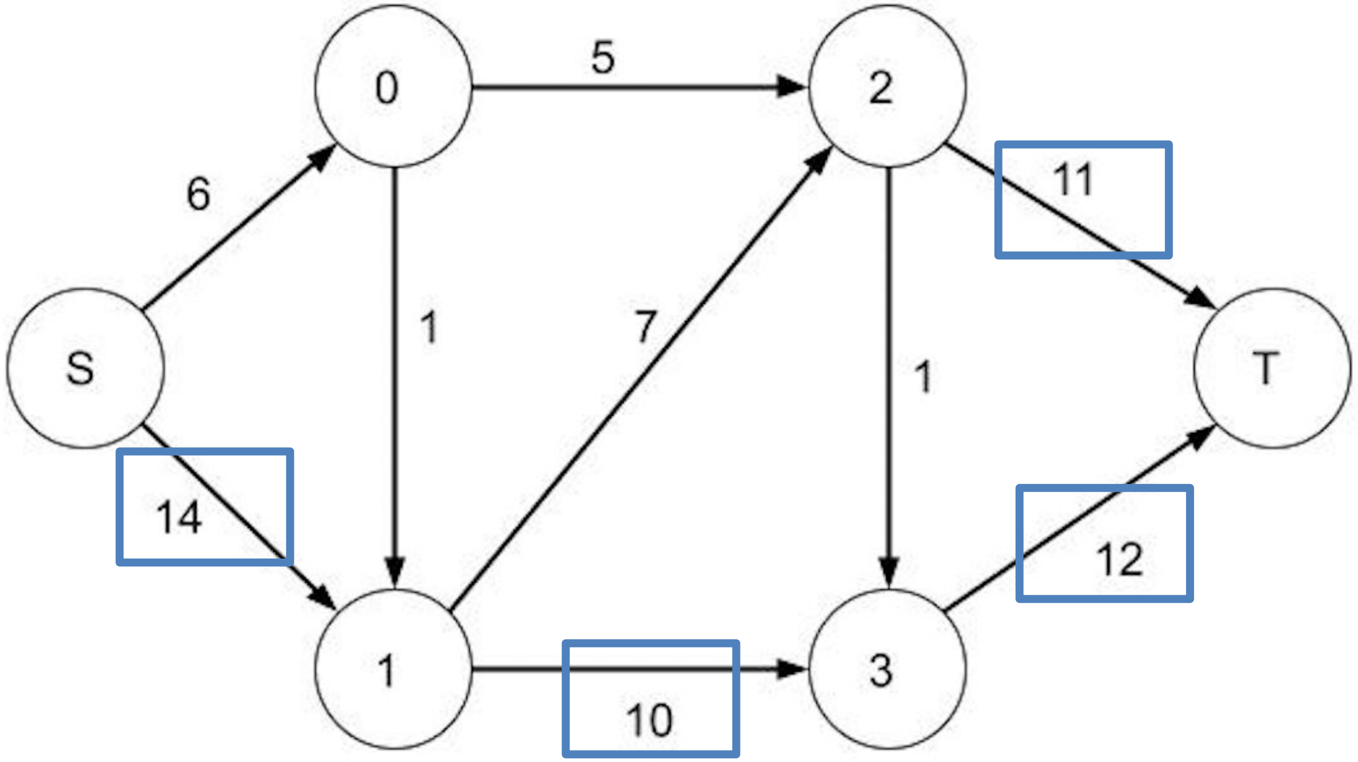
Using capacity scaling, calculate the max-flow of the following network:



Solution

max capacity out of S = 14

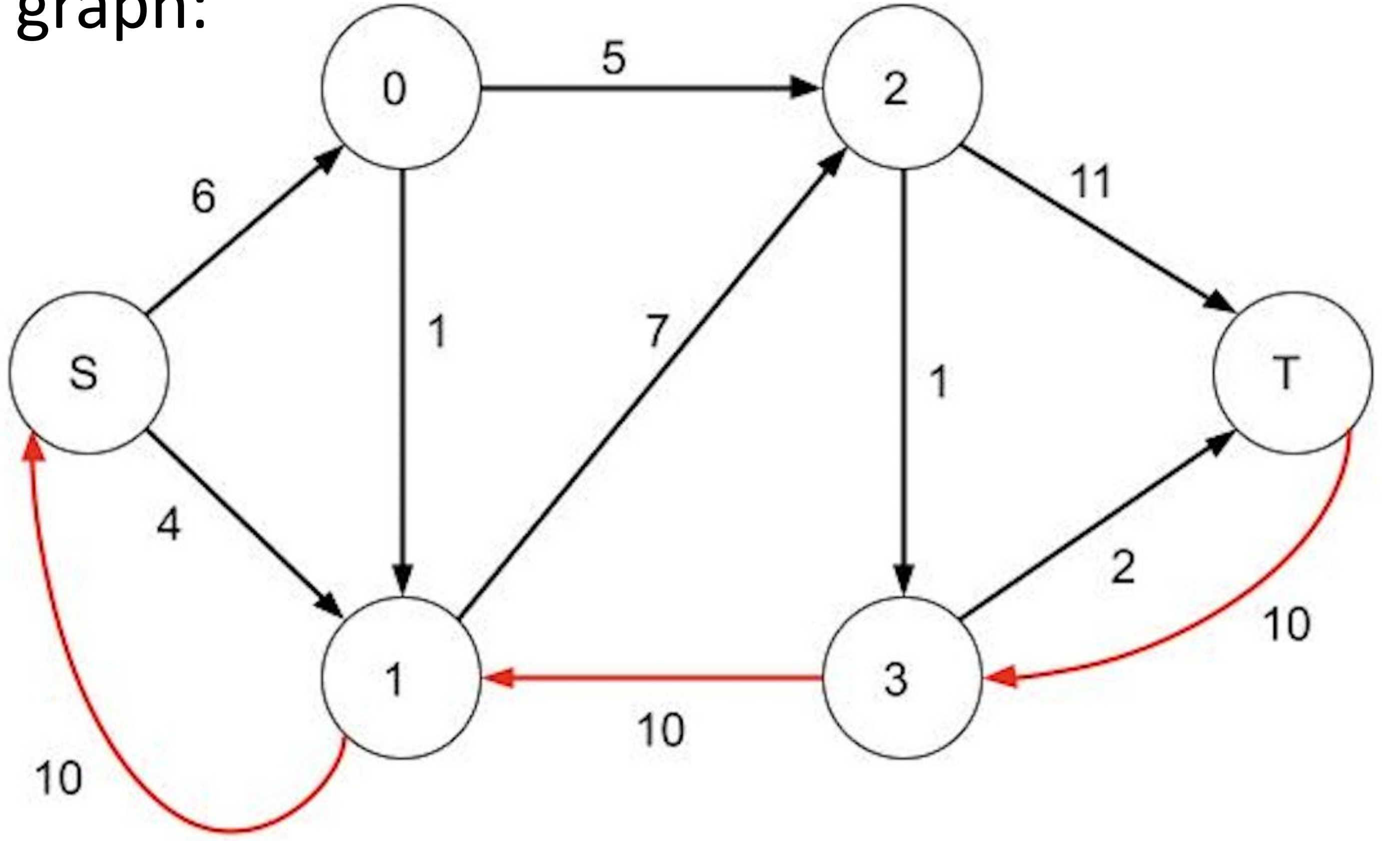
At Δ = 8 phase, available edges:



Solution cont'd

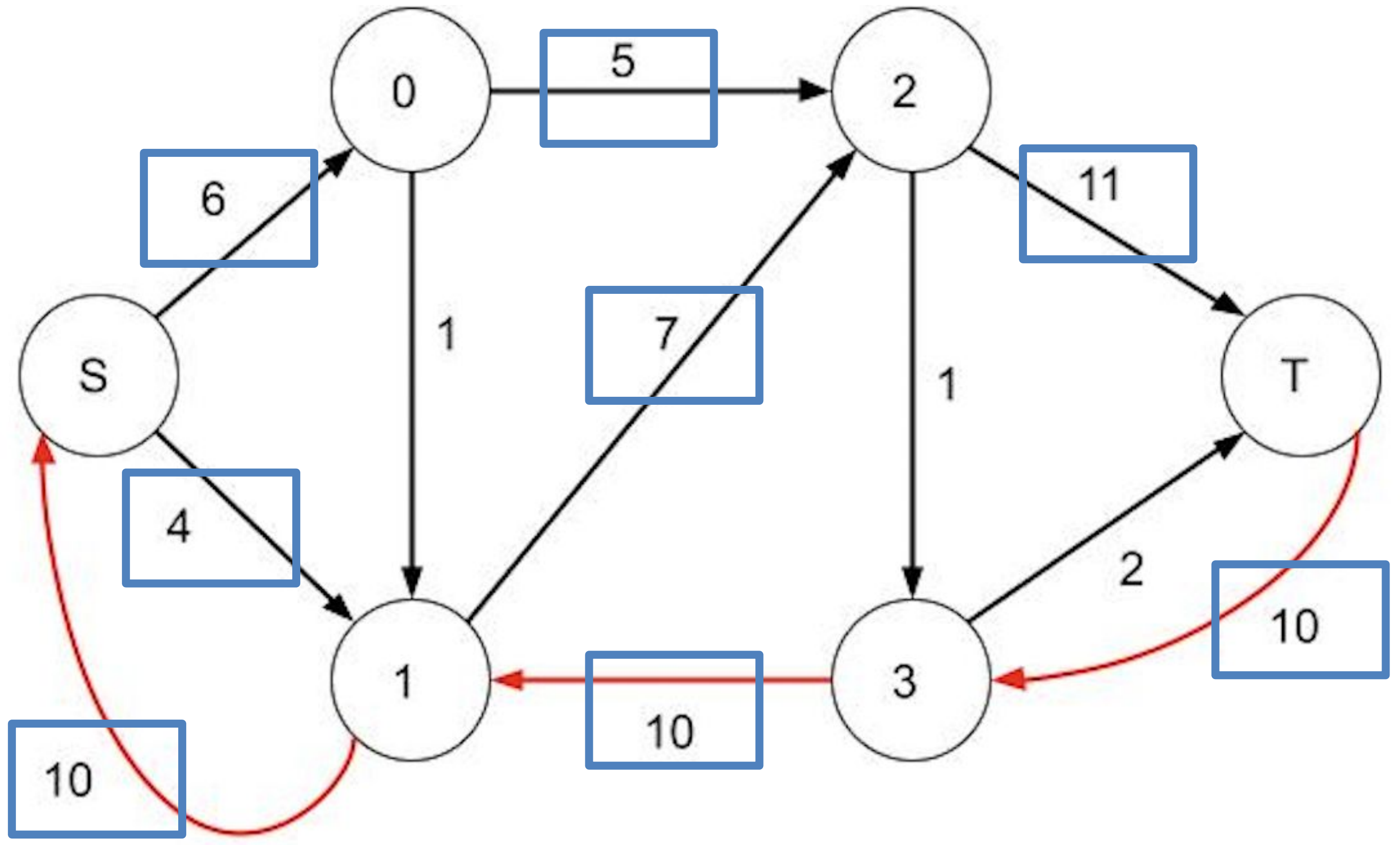
At Δ = 8 phase, we have 1 augmenting path (**S-1-3-T**, bottleneck=10).

Updated residual graph:



Solution cont'd

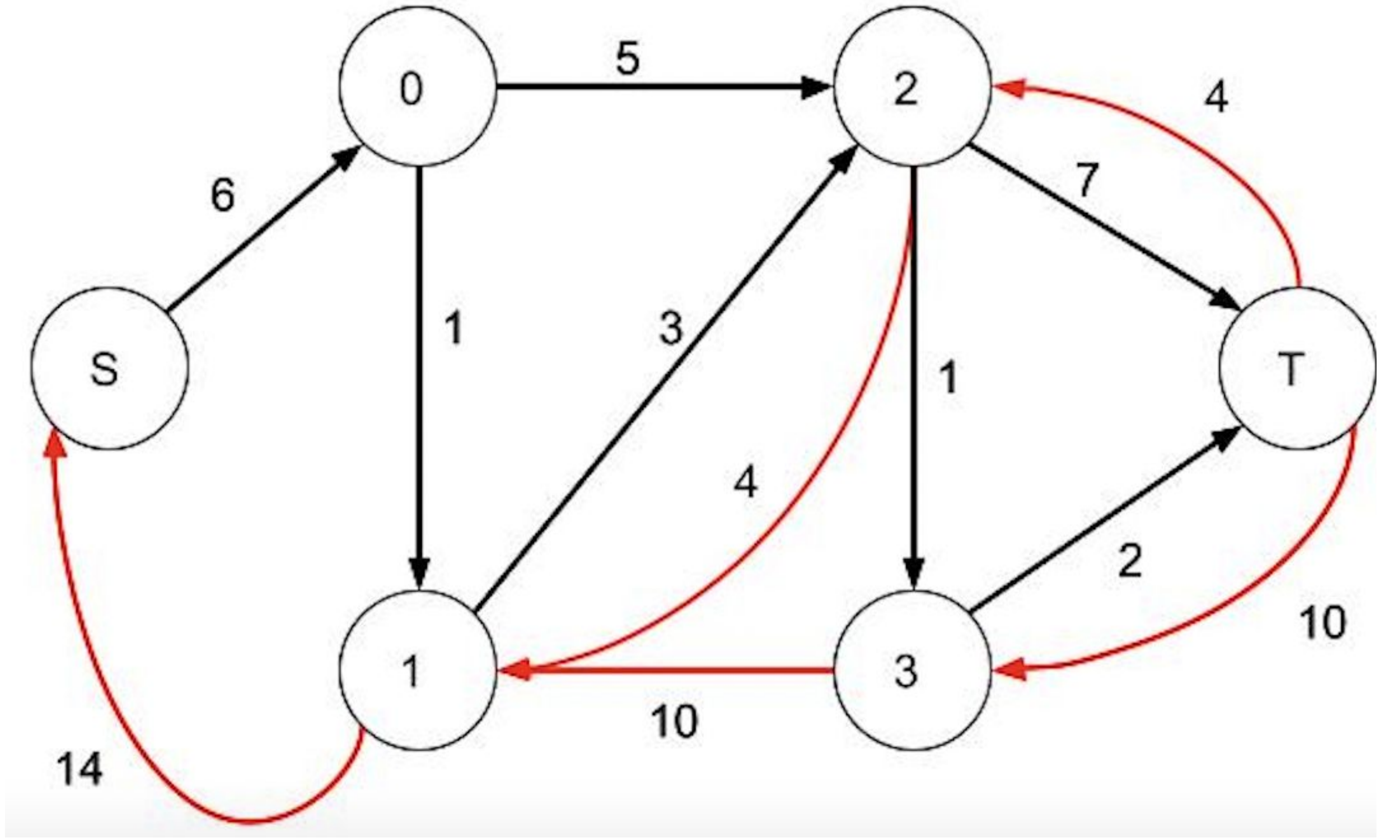
At Δ = 4 phase, available edges:



Solution cont'd

At Δ = 4 phase, we have 2 Augmenting Paths

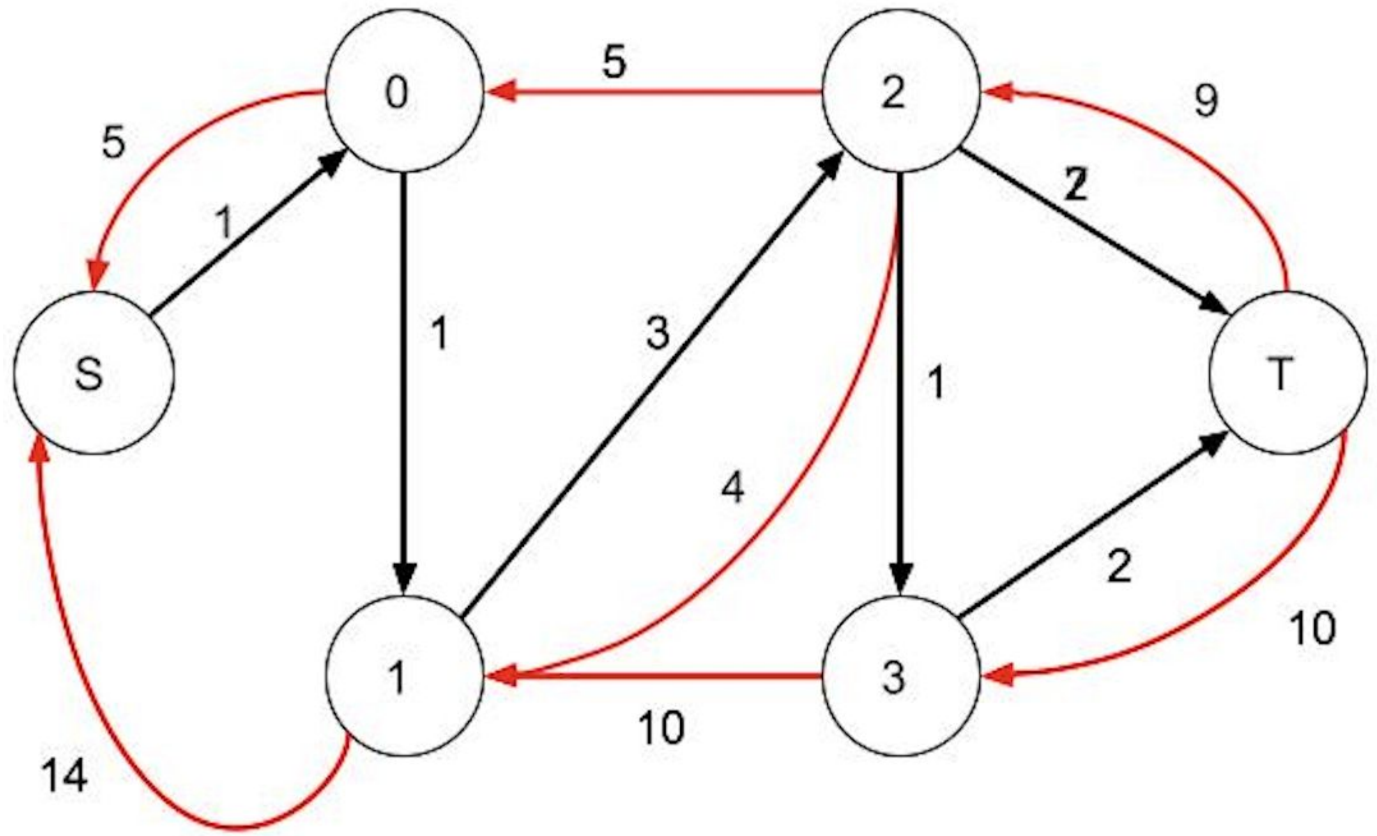
Residual graph after 1st path (**S-1-2-T**, bottleneck=4):



Solution cont’d

At Δ = 4 phase, we have 2 Augmenting Paths

Residual graph after 2nd path (**S-0-2-T**, bottleneck=5):

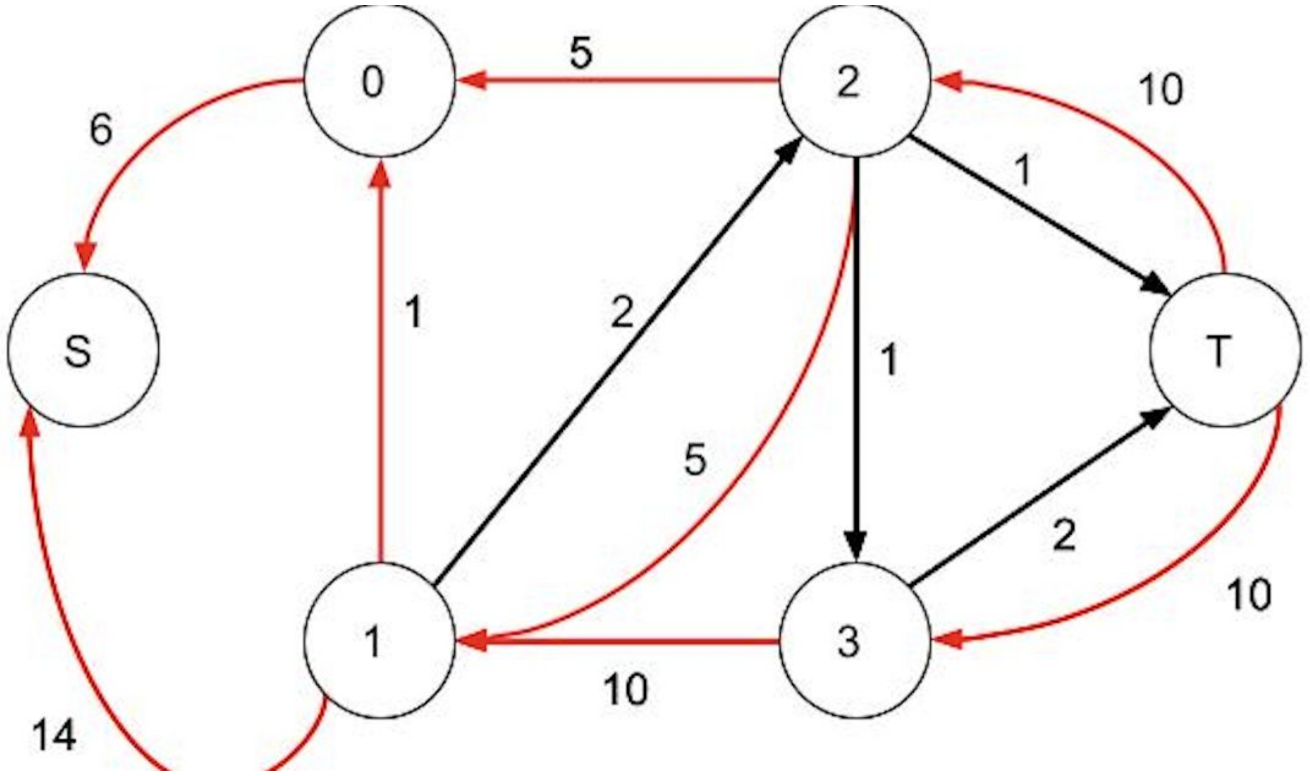


Solution cont'd

At Δ = 2 phase, No possible Augmenting Path

At Δ = 1 phase, 1 Augmenting path possible (**S-0-1-2-T**, bottleneck=1)

Final residual graph:



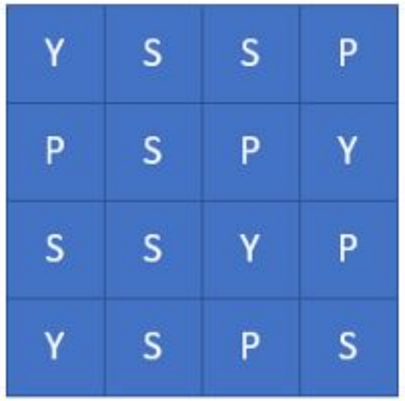
Algorithm Terminates, Max-Flow = 20

Question 2 - SPY Detection

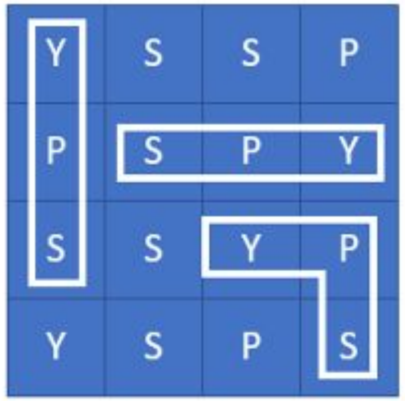
Counter Espionage Academy instructors have designed the following problem to see how well trainees can detect SPY's in an n×n grid of letters S, P, and Y. Trainees are instructed to detect as many **disjoint copies** of the word SPY as possible in the given grid. To form the word SPY in the grid they can start at any S, move to a neighboring P, and then move to a neighboring Y (only north, east, south or west). Give an efficient network flow-based algorithm to find the largest number of disjoint SPY's.

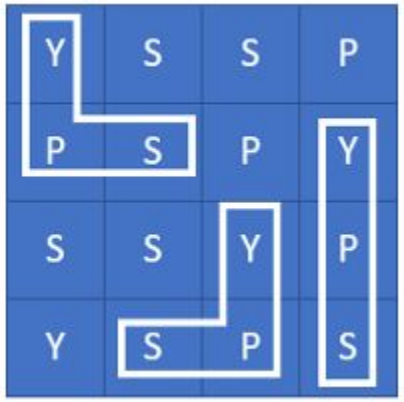
*Note: We are only looking for the largest **number** of SPY's, not the actual location of the words. Proof of correctness is left as an exercise.*

Example:



Possible optimal sols:





Solution: SPY Detection

We construct a Flow Network as follows, with **all edges having capacity 1**:

1. Create one layer of nodes for all the S's in the grid. Connect this layer to a source node s, directing from s.

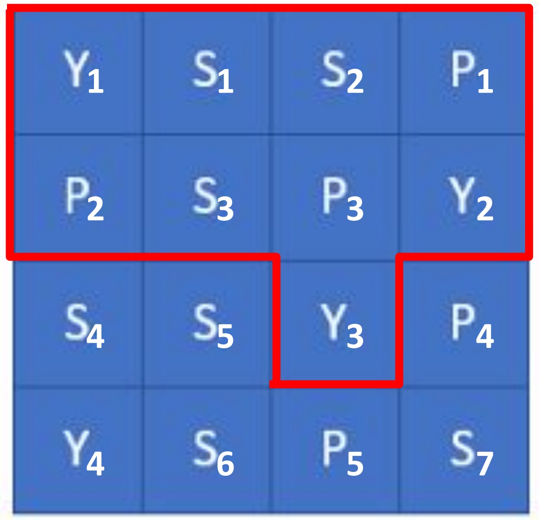
2. Create **two layers of nodes** for all the P's in the grid. Connect the first layer to the S's based on whether they are adjacent in the grid, directing from the S's layer. Also, connect the first layer to the second layer by connecting the nodes that correspond to the same location.
*(Representing **each P as an edge with capacity 1**, similar to how we solved the node-disjoint paths problem.)*

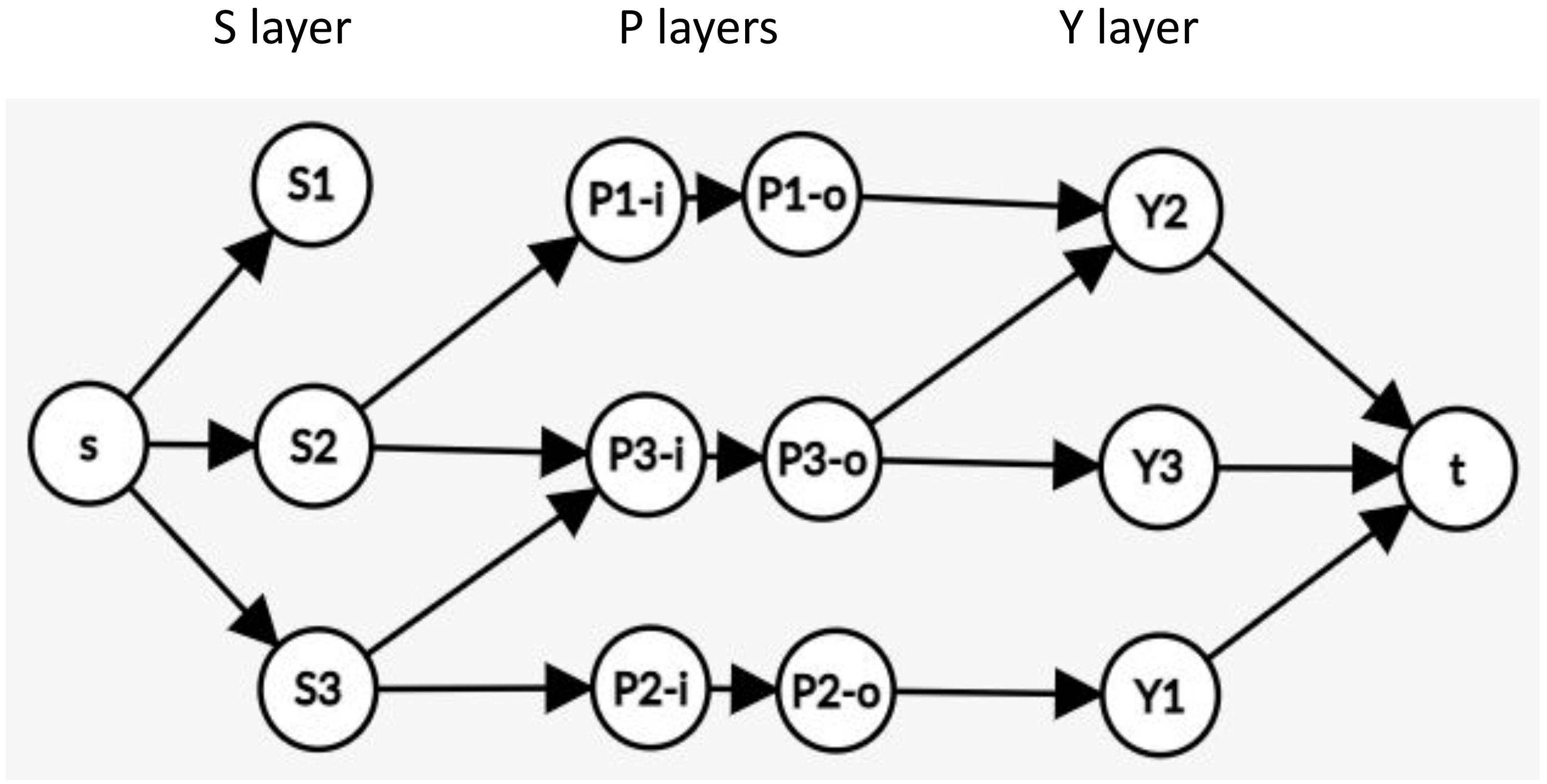
3. Create a layer of nodes for all the Y's in the grid. Connect these to a sink node t, directing from the Y's layer. Also, connect them to the second layer of P's based on whether the P and Y are adjacent in the grid, directing edges from the P's layer.

*(Note that we don't need to duplicate nodes for S or Y, since there are edges of capacity 1 going into S's and edges of capacity 1 leaving Y's **which will force these letters to be picked only once**.)*

Now the value of Max-Flow in this Flow Network will give us the maximum number of disjoint SPY's.

Solution: SPY Detection (example)





Claim to prove: There is a flow of value k in the constructed network *if and only if* there are k disjoint copies of SPYs in the grid.

Question 3 - Edge Connectivity

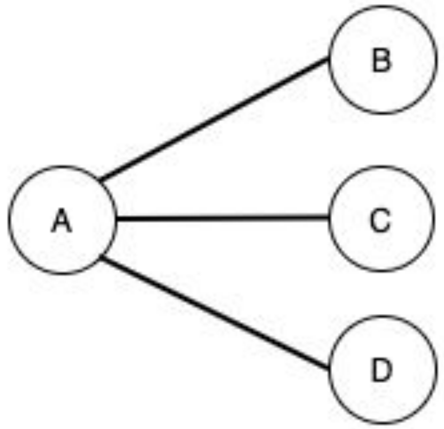
The edge connectivity *k* of an undirected graph G = (V, E) is the minimum number of edges that must be removed to disconnect G (i.e., to split G into 2 components). For example, the edge connectivity of a tree is 1.

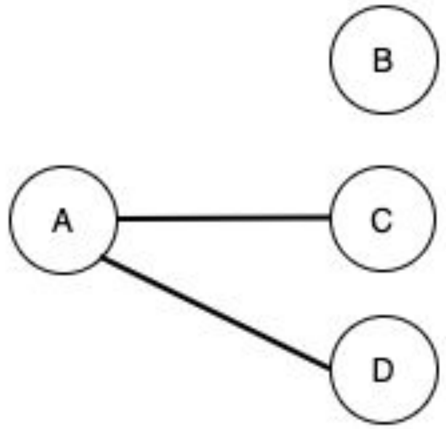
(a) Show how the edge connectivity *k* of an undirected graph G with n vertices and m edges can be determined efficiently by running a max-flow algorithm.

(b) Describe how many max-flow computations will be required to solve edge connectivity.

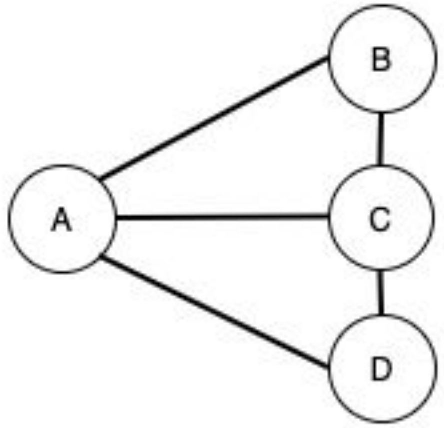
Example: Edge Connectivity

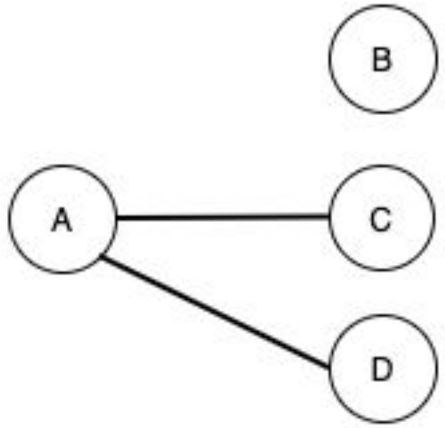
k = 1





k = 2



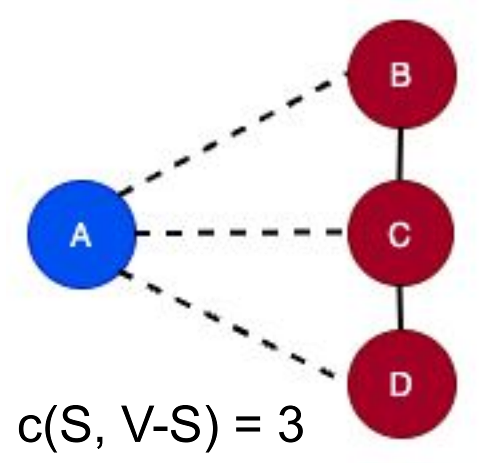


Solution: Edge Connectivity

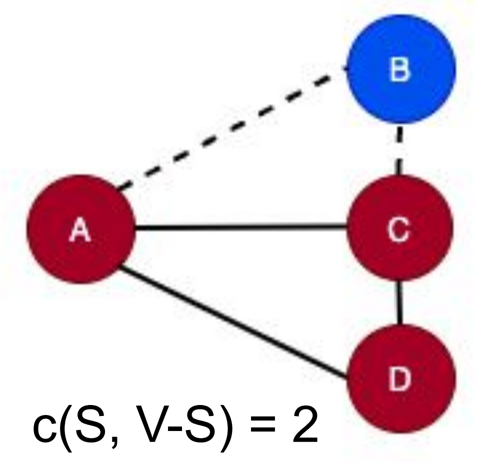
Let V be the set of n vertices.

For a cut (S, V-S), let its capacity c(S, V-S) denote the number of edges crossing the cut.

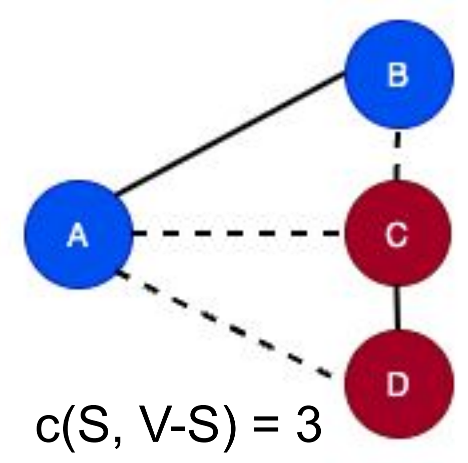
By definition, the edge connectivity $k = \min_{S \subset V} \{ c(S, V\text{-}S) \}$.
(Meaning that the minimum number of edges required to disconnect the graph is equal to the minimum cut capacity)



c(S, V-S) = 3



c(S, V-S) = 2



c(S, V-S) = 3

k = min { 3, 2, 3, ... }

= 2

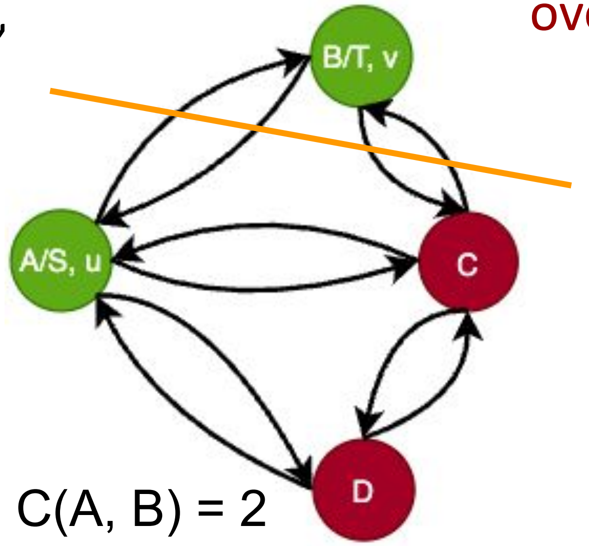
Solution: Edge Connectivity (cont'd)

Construct a **directed** graph G* from G by replacing each edge (u, v) in G by two directed edge (u, v) and (v, u) in G* with capacity 1.

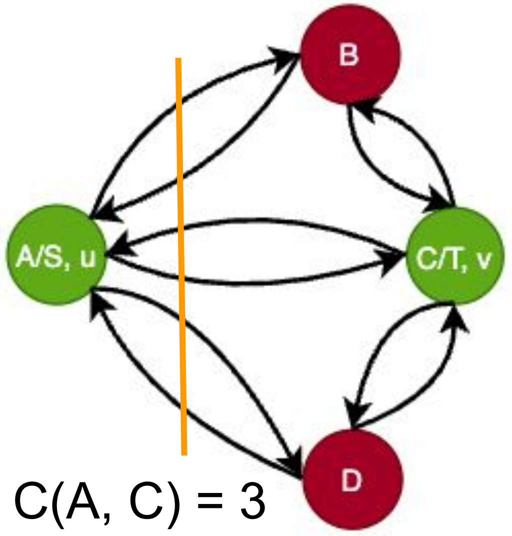
Now in G*, **fix a vertex** s ∈ V as the source. For every cut (S, V-S) in G*, there is a vertex t ∈ V such that s and t are on different sides of the cut, so we can compare the s-t cut value for **every t** as the sink.

Let C(s, t) denote the value of the min s-t cut in G*.

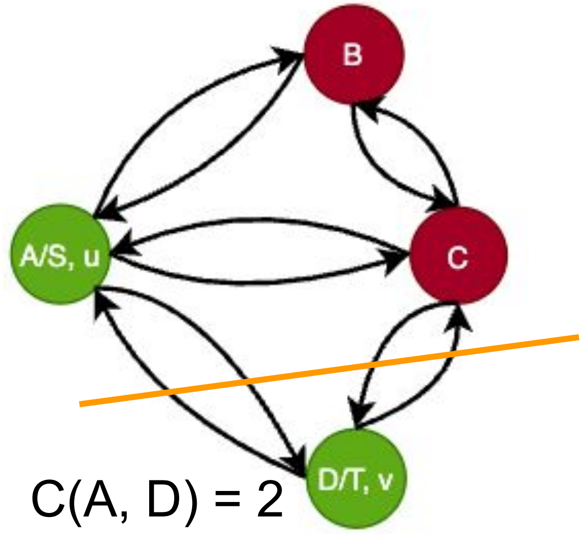
Thus, ove



C(A, B) = 2



C(A, C) = 3



C(A, D) = 2

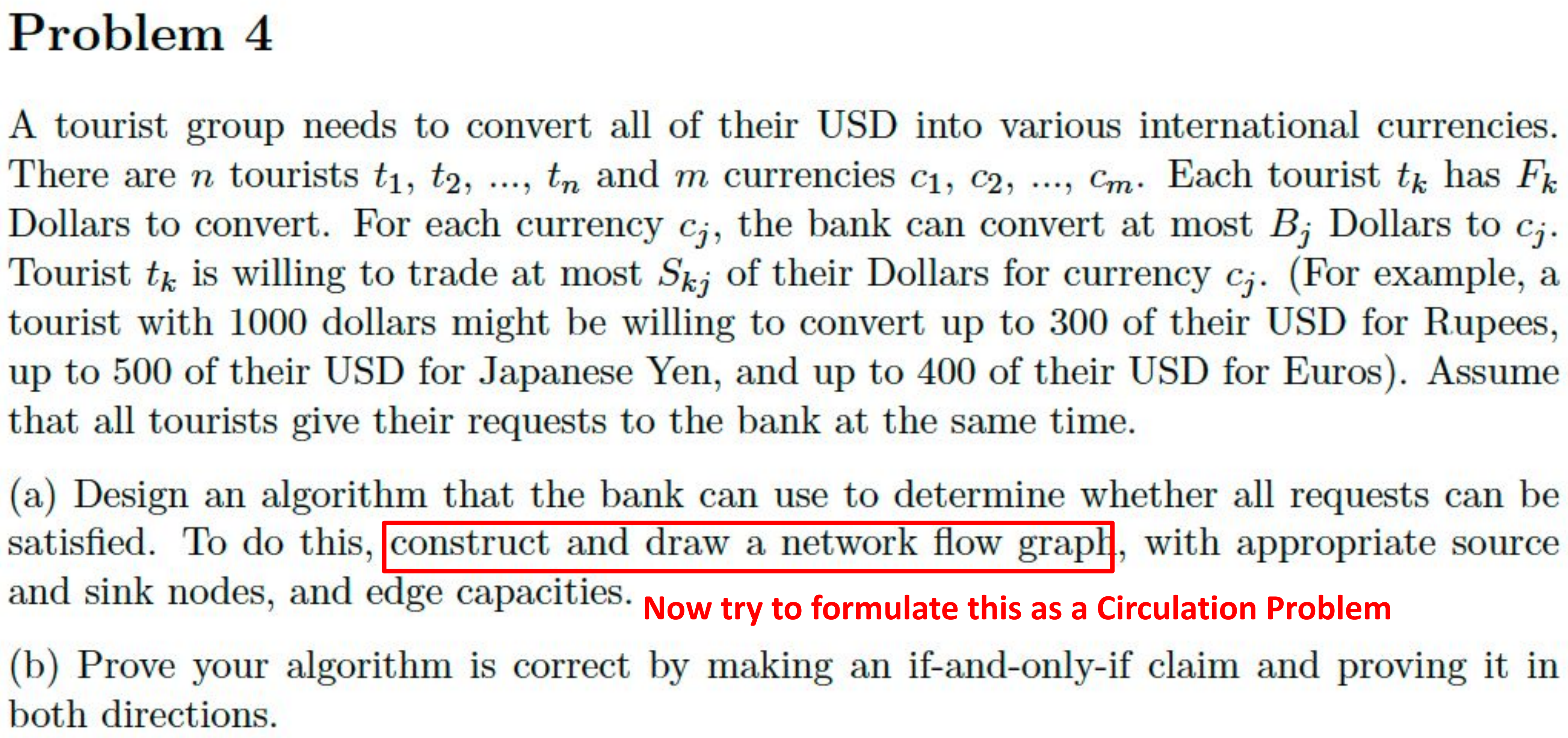
k = min{2, 3, 2}
= 2

Solution: Edge Connectivity (cont'd)

To compute the value of min s-t cut in G*, simply calculate the max-flow setting s as the source and t as the sink in G*. This follows from the max-flow min-cut theorem.

(b) There are a total of n vertices, and we fix a vertex s. Hence, we need to run max-flow for all the remaining vertices (except s) as the sink and then take the min. Therefore, there will be (n-1) max-flow computations.

Bonus: Homework Revisited



Problem 4

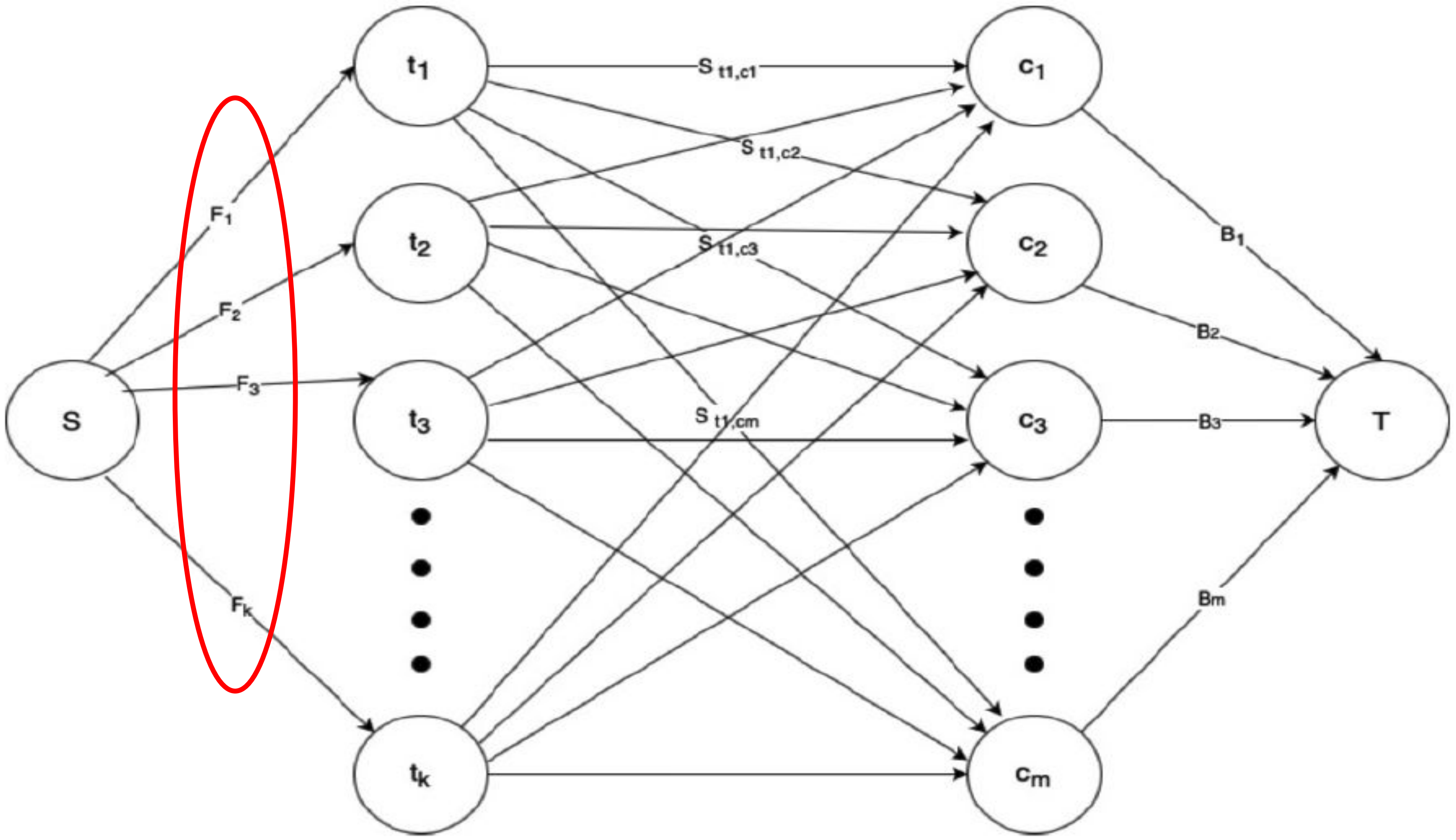
A tourist group needs to convert all of their USD into various international currencies. There are n tourists t_1, t_2, ..., t_n and m currencies c_1, c_2, ..., c_m. Each tourist t_k has F_k Dollars to convert. For each currency c_j, the bank can convert at most B_j Dollars to c_j. Tourist t_k is willing to trade at most S_{kj} of their Dollars for currency c_j. (For example, a tourist with 1000 dollars might be willing to convert up to 300 of their USD for Rupees, up to 500 of their USD for Japanese Yen, and up to 400 of their USD for Euros). Assume that all tourists give their requests to the bank at the same time.

(a) Design an algorithm that the bank can use to determine whether all requests can be satisfied. To do this, construct and draw a network flow graph, with appropriate source and sink nodes, and edge capacities.

Now try to formulate this as a Circulation Problem

(b) Prove your algorithm is correct by making an if-and-only-if claim and proving it in both directions.

(a) How can we turn this flow network into a circulation network?



(b)

Claim to prove: The problem has a solution (i.e., all the tourists are able to exchange their specified USD while following all the constraints), if and only if *there is a feasible circulation in the constructed network*.

Proof: refer to the hw solution.

Approximation Algo / LP

Mengxiao Zhang

USC Viterbi
School of Engineering

4.1 Set Cover

- Valid instances : Universe U, $|U| = n$. Family of sets $F = \{S_1, \ldots, S_m\}$, $S_i \subseteq U$ for all i.
- Feasible solutions : A set $I \subseteq [m]$ such that $\bigcup_{i \in I} S_i = U$.
- Objective function : Minimizing $|I|$.

4.2 Weighted Set Cover

- Valid instances : Universe U, $|U| = n$. Family of sets $F = \{S_1, \ldots, S_m\}$, $S_i \subseteq U$ for all i. Each set S_i has a cost c_i.
- Feasible solutions : A set $I \subseteq [m]$ such that $\bigcup_{i \in I} S_i = U$.
- Objective function : Minimizing $\sum_{i \in I} c_i$.

Equivalent ILP formulation

$$
\begin{aligned}
\text{minimize:} \quad & \sum_{S \in F} c(S) \cdot x_S & & \\
\text{subject to:} \quad & \sum_{\{S : e \in S\}} x_S \geq \; 1 & & \text{for each element } e \in U \\
& x_S \in \; \{0, 1\} & & \text{for each set } S \in F
\end{aligned}
$$