Bo Yang (Aaron) USCID: 7526922531

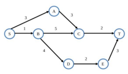
USC CSCI 570 Homework4 Date: 2021-03-17

Note. You are to solve problems 2, 3 and 4 by using the following steps:

1. Describe how to construct a flow network.
2. Make a claim. Something like "this problem has a feasible solution if and only if the max flow is ..."
3. Prove the above claim in both directions

1. Running the Algorithm (15 points)

You are given the following graph G. Each edge is labeled with the capacity of that edge.

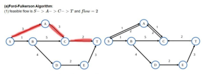


- a. Find a max-flow in G using the Ford-Fulkerson algorithm. Draw the residual graph G_f corresponding to the max flow. You do not need to show all intermediate steps. (10 pts)
- b. Find the max-flow value and a min-cut. (5 pts)

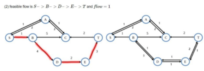
Solution:

>

(a)Ford-Fulkerson Algorithm:
(1) feasible flow is $S->A->C->T$ and $flow = 2$

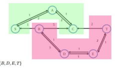


(2) feasible flow is $S->B->D->E->T$ and $flow = 1$



(3) The Max Flow is $2+1=3$

(b)Max Flow & Min Cut:
The Max Flow is value is 3
And a feasible minimual cut as follows:



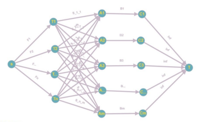
$S=\{S,A,C\}, S'=\{B,D,E,T\}$

2. Trading Currencies (20 points)

A group of traders are leaving Switzerland, and need to convert their Francs into various international currencies. There are n traders $t_1, t_2, \ldots t_n$ and m currencies $c_1, c_2, \ldots c_m$. Trader t_k has F_k Francs to convert. For each currency c_j, the bank can convert at most B_j Franks to c_j. Trader t_k is willing to trade as much as S_{kj} of his Francs for currency c_j .(For example, a trader with 1000 Francs might be willing to convert up to 200 of his Francs for USD, up to 500 of his Francs for Japanese's Yen, and up to 200 of his Francs for Euros). Assuming that all traders give their requests to the bank at the same time, design an algorithm that the bank can use to satisfy the requests (if it is possible).

Solution:

>



(a)Definition:
We construct the network as follows:
First, we create a source node S and denote every trader and currencies as a node $T_k, C_j, k \in [n], j \in [m]$ respectively. As the question said, "Trader t_k is willing to trade as much as S_{kj} of his Francs for currency C_j ." So, we create a set of intermediate nodes called "accepted money", also namely $A_j j \in [m]$. And connect all intermediate nodes A and trader T with a directed edge and weight is S_{kj}. Next, continue to add n directed edges from S to each trader node $T_k, k \in [n]$, and the capacity is $F_i, i \in [n]$.Since the bank only allowed to "convert at most B_j Franks to C_j. ", so we connect all node in A to all currencies nodes C with an edge whose weight is $B_j, j \in [m]$. and finally, we add a target node T and connect all currencies node ($C_J, j \in [m]$) to it with a directed edge of capacity ∞. My algorithm is to first compute the max-flow value of this generated graph by using Ford Fulkerson algorithm. According to our construction, the value of the max-flow is no more than $\sum_{i=0}^{n} F_i$.

(b)Claim:

There exist a feasible plan if and only if the above network has a max-flow of value $\sum_{i=0}^{n} F_i$.

(c)Proof:
(flow value $\rightarrow$ feasible plan)
If there exists a max-flow value $\sum_{i=0}^{n} F_i$, we can construct n disjoint paths according this flow. Specifically, for each trader, there must will have F_i France money flow from S, and finally it will reach the sink T, which means it reaches the boundary point. Also this will be the most feasible plan as well.

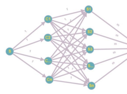
(feasible plan $\rightarrow$ flow value)
If there exists a feasible plan like this, every trader send F_i France money to the bank and finish currency conversion, then the max flow value should be $\sum_{i=0}^{n} F_i$.

3. Getting Vaccinated (20 points)

After getting vaccinated, students will be able to return to in-person classes next semester. There will be n students, $s_1, s_2, \ldots s_n$, return to in-person classes, and there will be k in-person classes. Each in-person class consists of several students and a student can be enrolled in more than one in-person class. We need to select one student from each in-person class and the maximum times a student is selected should be less than m. Design an algorithm that decides if such a selection exists, or not.

Solution:

>



(a)Definition:
We construct the network as follows:
First, we create a source node S and denote every class and student as a node $C_i, S_j, i \in [k], j \in [n]$ respectively.
And then add k directed edges from S to each class node $C_i, i \in [k]$, and the capacity is 1.Since we only allowed to "select one student from each in-person class", so we connect all class nodes to all student nodes with an edge whose weight is 1. and finally, we add a target node T and connect all students node ($S_i, i \in [n]$) to it with a directed edge of capacity m.
My algorithm is to first run the Ford Fulkerson algorithm. And then find a min-cut. All saturated edges we found between class nodes and student nodes could be seen as selected student-class pairs.

(b)Claim:
There exist a feasible selection if and only if the above network has a max-flow of value k

(c)Proof:
(flow value $\rightarrow$ feasible selection)
If there exists a max-flow of value k, we can construct k disjoint paths according this flow. Specifically, for each class, there must be one unit flow from S to each class node C_i, then finally it will reach the sink node T, which means it reaches the boundary point. Also this will be the most feasible plan as well.

(feasible selection $\rightarrow$ flow value)
If there exists a feasible selection like this, every class will select at most one student from their class, then the max flow value should be k.

4. Visiting Campus (25 points)

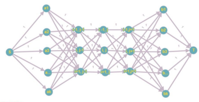
USC Admissions Center needs your help in planning paths for campus tours given to prospective students or interested groups. Let USC campus be modeled as a weighted, directed graph G containing locations V connected by one-way roads E. On a busy day, let k be the number of campus tours that have to be done at the same time. It is required that the paths of campus tours do not use the same roads. Let the tour have k starting locations $A=\{a_1, a_2, \ldots, a_k\} \subset V$. From the starting locations the groups are taken by a guide on a path through G to some ending location in $B=\{b_1, b_2, \ldots, b_k\} \subset V$. Your goal is to find a path for each group i from the starting location, a_i , to any ending location b_j such that no two paths share any edges, and no two groups end in the same location b_j.

- a. Design an algorithm to find k paths $a_i \rightarrow b_j$ that start and end at different vertices and such that they do not share any edges. (15 pts)
- b. Modify your algorithm to find k paths $a_i \rightarrow b_j$, that start and end in different locations and such that they share neither vertices nor edges. (10 pts)

Solution:

>

(a)Question:



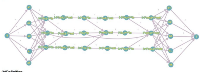
(a)Definition:
We construct the network as follows:
First, we can use point (i,j) to denote the location of each builidng. For those points (buildings) in V, but not belongs to A nor B, we say their location is (i,j). In addition, for all (i,j), we add directed edges to its next level adjacent nodes, specifically, (i,j) connected to $(i-1,j+1)$, $(i,j+1)$ and $(i+1,j+1)$.
About start points A and end points B, we firstly add a source node S and add an directed edge from S to each point in A with capacity 1. Next, we continue to add a target node T and connect all points in B to it with a directed edge of capacity 1. In this way, there won't have any intersection between the paths of any two tours. My algorithm is to first compute the max-flow value of this generated graph by using Ford Fulkerson algorithm. According to our construction, the value of the max-flow is no more than k.

(a)Claim:
There exists k disjoint paths if and only if the above network has a max-flow of value k

(a)Proof:
(flow value $\rightarrow$ generate path)
If there exists a max-flow of value k, we can construct k disjoint paths according this flow. Specifically, for each point in set A and B, there must be one unit flow from S to a specific point, and then finally it will reach the T. This one unit flow therefore indicates a path from a_i to one of the boundary point B. Also, these paths will never intersect. For the same reason, there is at most one unit flow reach the next level node from its upper level, If not, there would be two units coming into next level. As the corresponding next level node is of capacity 1, it is not feasible.
(generate path $\rightarrow$ flow value)
On the other hand, if the value of the max-flow is not k, we claim that there does not exist such k disjoint paths because if there exists k disjoint paths, we can construct a flow of value k by following these paths. Concretely, for a path $a_* \rightarrow (x_i y_i) \rightarrow (x_{i+1}, y_{i+1}) \rightarrow \ldots \rightarrow b_* \rightarrow T$, we would have a one unit flow from that path. This is a feasible flow as the k paths are disjoint.

>

(b)Question:



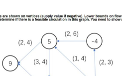
(b)Definition:
We construct the network as follows:
First, we can use point (i,j) to denote the location of each builidng. For those points in V, but not belongs to A nor B, we split every point (i,j) in that set into two points (i^{in}, j^{in}) and (i^{out}, j^{out}) and connected with an edge with capacity 1. In addition, for all (i^{out}, j^{out}), we add directed edges to its adjacent in-nodes, specifically, (i^{out}, j^{out}) connected to $(i+1^{in}, j^{in})$, $(i^{in}, j+1^{in})$, $(i-1^{in}, j^{in})$ and $(i^{in}, j-1^{in})$.
About start points A and end points B, we firstly add a source node S and add an directed edge from S to each point in A with capacity 1. Next, we continue to add a target node T and connect all points in B to it with a directed edge of capacity 1. In this way, there won't have any intersection between the paths of any two tours. My algorithm is to first compute the max-flow value of this generated graph by using Ford Fulkerson algorithm. According to our construction, the value of the max-flow is no more than k.

(b)Claim:
There exists k disjoint paths if and only if the above network has a max-flow of value k

(b)Proof:
(flow value $\rightarrow$ generate path)
If there exists a max-flow of value k, we can construct k disjoint paths according this flow. Specifically, for each point in set A and B, there must be one unit flow from S to a specific point, and then finally it will reach the T. This one unit flow therefore indicates a path from a_i to one of the boundary point B. Also, these paths will never intersect. For the same reason, there is at most one unit flow reach the middle position, in-node or our-node, If not, there would be two units coming into one in-node. As the corresponding out-node is of capacity 1, it is not feasible.
(generate path $\rightarrow$ flow value)
On the other hand, if the value of the max-flow is not k, we claim that there does not exist such k disjoint paths because if there exists k disjoint paths, we can construct a flow of value k by following these paths. Concretely, for a path $a_* \rightarrow (x_i y_i) \rightarrow (x_{i+1}, y_i) \rightarrow \ldots \rightarrow b_*$, we would have a one unit flow $S \rightarrow a_* \rightarrow (x_i^{in} x_i^{out}) \rightarrow (x_i^{out} y_i^{out}) \rightarrow (x_{i+1}^{in} y_i^{in}) \rightarrow (x_{i+1}^{out} y_i^{out}) \rightarrow \ldots \rightarrow b_* \rightarrow T$. This is a feasible flow as the k paths are disjoint.

5. Circulation (20 points)

In the network below, the demand values are shown on vertices (supply value if negative). Lower bounds on flow and edge capacities are shown as (lower bound, capacity) for each edge. Determine if there is a feasible circulation in this graph. You need to show all your steps.

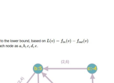


- a. Turn the circulation with lower bounds problem into a circulation problem without lower bounds. (8 pts)
- b. Turn the circulation with demands problem into the maximum flow problem. (8 pts)
- c. Does a feasible circulation exist? Explain your answer. (4 pts)

Solution:

>

(a):
Create a flow on edge equal to the lower bound, based on $L(v) = f_{in}(v) - f_{out}(v)$
For convenience, I denote each node as a, b, c, d, e.



$L(a) = 2+2-0 = 4$
$L(b) = 2+1-3-2 = -2$
$L(c) = 2-2 = 0$
$L(d) = 0-2-1-2 = -5$
$L(e) = 3+2-2 = 3$

After recpmpution, based on formula $d'(v) = d(v) - L(v)$, the demand value becomes:
$d'(a) = 9-4 = 5$
$d'(b) = 5+2 = 7$
$d'(c) = -4$
$d'(d) = -13+5 = -8$
$d'(e) = 3-3 = 0$

(b):
Create a source vertex S and connect it with c and d, with an edge whose weight are 4 and 8. Also, create a target vertex T and connect a and b with T, and the weights are 5 and 7 respectively.

(c):
Yes. since after compute the maximum flow of the new graph. we can found that for each node, the flow constraint is satisfied $(in == out)$

6. Installing Software (Not Graded)

Suppose that you have just bought a new computer and you want to install software on that. Specifically, two companies, which you can think of like Microsoft and Apple, are trying to sell their own copy of n different products, like Operation System, Spread Sheet, Web Browser. For each product $i, i \in \{1, 2, \ldots\ldots, n\}$, we have

- the price $p_i \geq 0$ that Microsoft charges and the price $p'_i \geq 0$ that Apple charges.
- the quality $q_i \geq 0$ of Microsoft version and the quality $q'_i \geq 0$ of Apple version.

For example, Apple may provide a better Web Browser Safari, but Microsoft a better Word Processor. You want to assemble your favorite computer by installing exactly one copy of each of the n products, e.g. you want to buy one operating system, one Web Browser, one Word Processor, etc. However, you don't want to spend too much money on that. Therefore, your goal is to maximize the quality minus total price.

However, as you may know, the products of different companies may not be compatible. More concretely, for each product pair (i, j), we will suffer a penalty $\tau_{ij} \geq 0$ if we install product i of Microsoft and product j of Apple. Note that τ_{ij} may not be equal to τ_{ij} just because Apple's Safari does not work well on Microsoft Windows doesn't mean that Microsoft's Edge does not work well in Mac-OS. We assume that products are always compatible internally, which means that there is no penalty for installing two products from the same company. All pairwise penalties will be subtracted from the total quality of the system.

Your task is then to give a polynomial-time algorithm for computing which product i to purchase from which of the two companies (Apple and Microsoft) for all $i \in \{1, 2, \ldots n\}$, to maximize the total system quality including the penalties) minus the total price.

Solution:

>

(a)Definition:

(b)Claim:

(c)Proof: