Assignment 7

CSCI 270

Paul Kim | pbkim@usc.edu | 1723717002

Question 1

(1) [10 points]
Decide whether you think the following statement is true or false. If it is true, please prove it. If it is false, please give a counterexample.

Let G be a flow network with a path from s to t. For every maximum flow f on G, there is an edge e such that $c_e = f(e)$

Answer:

// Note: The max flow value is unique. However, there can be multiple networks (assignment of flows along edges) that achieve that flow, such that all such networks are referred to as maximum flows.

// Piazza @435, Assume flows are integer value. C_e is the capacity of edge e (maximum amount of flow able to pass on that edge).

// Intuition: We can use the idea of residual capacities to claim this property is true.

TRUE

"Prove that in every max-flow, at least one edge is used to full capacity."

Proof by Contradiction:

Assume that there is a flow network that has a max flow with a path from s to t, such that for every maximum flow f on G, there is not an edge such that C_e = F(e).

Since f is a maximum flow on G, there is no other valid flow f' on G such that f' > f.

We keep note that since C_e is the capacity of each edge, f(e) can never be a value greater than C_e. Therefore, we can conclude from this statement that for every edge, f(e) < C_e.

If there exists a path from s to t in G. Since every edge in G is so that f(e) < C_e as we claimed above, then there exists a path from s to t in which for all edges, e*, along the path, f(e*) < C_e*.

For all these edges, let R(e) = C_e* - F(e*). (Residual Capacities of these e* edges).

If we assign some minimum residual capacity from this path from all edges e*, and it must be a positive value since f is a max flow. Given the non-negativity and capacity theorems, F(e*) > 0 and

C_e* > F(e*). We concluded that C_e* > F(e*) above earlier. R(e*) is positive as C_e* - F(e*), and all values are greater than 0. R(e*) > 0.

Let us consider the f' flow, along with adding the minimum residential capacity added to each edge on the path from S to T.

We can prove that the network flow that we are analyzing is a legal flow: @Piazza don't need to reprove.

1. Conservation property holds true. (Maintains conservation on both sides, flow in = flow out).

2. Non-Negativity property holds true. (Since f was already valid max flow, adding a residual capacity > 0 will still be valid and positive).

3. Capacity property holds true. (Because we captured the minimum residual value, and added it to each F(e*) edge, f' still maintains capacity.) (M <= C_e*-F(e*))

We can see that f' >= f.

However, we claimed that since f is a maximum flow on G, that there is no other legal flow f' on G such that f' > f. However, we show here that by adding the minimum residual value that we calculated to the edges into the flow, f' >= f, through this addition of the minimum residual capacity, and is proven to be a legal flow through Conservation, the Non-Negative Property, and Capacity theorems.

Proved by Contradiction: For every maximum flow on G, there is an edge e such that C_e = F(e).

We can also use the Lemma 1 provided by Kempe, saying that let F be any s-t flow, and (S,S') be any s-t cut. Then, V(f) <= C(S,S') implying that cuts are bottlenecks for flows. Proving by contradiction: If we were to assume that C_e != F(e), then than means that V(f) is strictly less than the capacity, but by that logic, the flow F is no longer the max flow, which is a contradiction.

Question 2

(2) [15 points]

USC plans to offer m courses in 2022 Fall. Each class i has an enrollment cap c_i. Suppose that USC has n students. Each student j has a list of classes that he/she wants to take at some point, as well as a number k_j of classes they would like to take next semester. (k_j will typically be smaller than the list of classes — a student might have a long list of 10 classes to take in the next few years, but only wants to take any 4 of them next semester.)

USC academic affairs office now starts to assign classes to students. They want to know whether with the given preference lists and class capacities, it is possible to give each student the number of classes they want. Give and analyze a polynomial-time algorithm for deciding if this is possible, and if so, outputting a class assignment satisfying all of the constraints.

Answer: *// Give an algorithm to determine if it is possible, given the preference list and class*

capacities, to give each student the number of classes they want (k_j) next semester.

*// *Just focus on the classes that students want to take next semester, i.e, K_j*

// Imagine you are given a long list, as well as a number. Think (CSCI 270, CSCI 201, CSCI 356, CSCI 350, CSCI 310, MATH 410, PHIL 350, PHIL 450) and $K_j = 3$. So a student wants any three courses out of this list of 8, and the algorithm needs to choose which ones that optimizes the choices of all students to classes.

//Similar to Bipartite Matching where we put a source on the left side with weights, and a sink on the right side with weights. The left nodes in the graph can represent the N students, and the right nodes can represent the M courses.

//USC plans to offer M courses. Each course M(i) has an enrollment cap C_i. There are N students. Each student N(j) has a list of classes that he/she wants to take at some point as well as a number of classes K_j they want to take next semester (K_j is typically smaller than the list of classes).

Find whether given the preference list and class capacities, if it is possible to give each student the number of classes they want. Give and analyze an algorithm for deciding if this is possible, and if so, output a class assignment satisfying all of the constraints.

Visualize a Graph:

Source node on the left with edges connecting to N student nodes with K_j classes desired as the flow value.

Sink node on the right with edges connecting to M class nodes with C_i class capacities as the flow value.

Connections between the N student nodes to M class nodes, assigned a weight of 1, to all the classes that they want from the list of classes.

Formal Algorithm: (Edmond's Karp).

Add a new source S and a new sink T.

Add a directed edges of capacity K_j from S to each N student nodes (”jth”) (on the left) (the number of classes a student wants to take next semester).

Add a directed edges of capacity C_i from each node M class nodes (”ith”) (on the right) (the enrollment capacity of each class).

Direct all edges of the original bipartite graph from left to right nodes (connecting the left student node to the nodes of classes that they want to take next year). It is not true that a student wants to take all courses offered so we only connect them to the courses they want. We will give these edges a capacity of value 1, since a student occupies one spot in a class.

Compute a max integer S-T flow F using Edmond's Karp Algorithm (Max Flow Algorithm).

RETURN as the matching M all edges e from the original bipartite graph such that F(e) = 1 and capture the flow as a variable X.

(We can also keep a set S in each class node that keeps track of the edges from students, to keep track of every student included in the class).

Capture the Bottleneck of flow coming out of the Source S (intuitively this is the sum of the number of classes that all students want to take for next semester).

*If it was possible to give each student the number of classes they want, it must be true that the max flow returned from the algorithm above equaled to the total sum of the number of classes students wanted. If the Max Flow returned is less than, then a student (or multiple students) were unable to get the number of classes they want.

If X (flow) != to the Bottleneck (Min-Cut)

RETURN that it was not possible to give each student the number of classes they wanted.

If X (flow) == to the Bottleneck (Min-Cut)

RETURN that it was possible to give each student the number of classes they wanted.

Runtime: O(M) for preprocessing and post processing, plus the runtime of Edmond's Karp Algorithm which runs in O(M^2*N) where M is the number of total edges in the graph, and N is the number of vertices in the network.

(*Number of iterations of path finding is at most the number of edges in the matching).

//What our flow algorithm is doing, is that it is constructing a max-flow network from students to classes, matching and optimizing the maximum number of students that can have the number of classes they want.

Correctness Proof: (Synonymous and a little more than the Lecture)

Let M be the set of edges returned by the algorithm.

Proof: The result M is a matching

Matching: A subset of edges such that for all vertices, at most one edge of M is incident on V. (A set of edges with no endpoints in common).

Proof by Contradiction:

Let us assume that there is a node V such that at least 2 edges e, e' are incident on V. (Assume V is on the left side) (Proof is synonymous if V is on the right). Because our algorithm picked them, both F(e) and F(e') were at least 1 (We assumed that we returned an integer max flow F, so if the flow on an edge is positive, it must be at least 1 (also makes sense because a student only takes 1 spot in a class capacity)). Because V has at least 2 units of flow leaving, it must have at least 2 units of flow entering, by conservation. Because (S,V) is the only edge into V, and it has a capacity of 1, this is a contradiction.

*Flow cannot use two edges into the same node, and similarly you can't have two edges leaving the same node.

Proof: M has a maximum size.

Proof by Contradiction:

Assume that there is a larger matching M': |M'| > |M|.

For each edge e = (u,v) of M', send one unit of flow along the path.

This defines some F' and we can verify that F' is a valid flow.

> Conservation holds because we defined the flow in terms of paths. (automatic when you give paths).
>
> Capacity: No edge can be part of two paths, because then, the node would be part of two edges in M', so M' would not be a matching graph.

So F' is a valid flow, and of value |M'|.

F had a value of |M|. So F' has more value that F, so F was not a valid max flow which is a contradiction. So M' does not exist and M is a maximum matching.

Proof: We check if it was possible to give each student the number of classes they want.

*The condition on the max-flow on the graph to give the enrollment required by the problem is that the value of the max flow must be equal to the sum of the number of classes desired by all N students.

We can apply Lemma 1 from the Max-Flow/Min-Cut Lecture. We let F be any S-T flow, and (S,S') be any S-T cut. Then V(F) <= C(S,S'). Implying that cuts are bottlenecks for flows. What we can use from this lemma is the fact that were are given the set of N (student) nodes and the number of classes that each student would like to take in the following semester. If we were to perform an initial cut of (S,S') of all the edges going out of the source S we constructed, what we will be doing is we will be capturing the "bottleneck", or total maximum possible flow that the algorithm can return.

For example, if there were 5 students that each wanted to take 5 classes each, then fully optimized, there are 25 spots of classes taken. Using this similar idea, if we sum the total edge flow weights of all the outgoing edges of source S, we get this value; the value of the fully optimized solution.

At the end of running the Edmond Karp's Algorithm, we are given a set of matchings and a flow value. At the end of our algorithm, if the returned flow value from our algorithm equals to the (S,S') cut that we performed initially, then it must be true that each student got the number of classes they want for the next semester.

However, if the returned flow value from our algorithm is less than the "bottleneck" max flow that we were supposed to get, that Lemma 1 proves it so, it must mean that a class filled up to capacity, and a student (or multiple students (generalized)) were unable to get their desired number of classes.

Therefore, if it was possible to give each student the number of classes they want, it must be true that the max flow returned from the algorithm above equaled to the total sum of the number of classes students wanted given by the (S,S') cut performed on the outgoing edges of S. If the Max Flow returned is less than, then a student (or multiple students) were unable to get the number of classes they want.

Also, by the capacity statement, there is only the flow outgoing from source S that distributes

through the entire network. As a result, if there is a max-flow value, then it must be equal to or less than the number of total classes desired by all students. *Put this little note here because just subtly show that max flow will never be higher than the flow emitted by the source. What comes out must come in.

Proof: That the set of matchings (class assignments) is accurate.

As we proved everything else above, this proof should be obvious. We only add to the set of each classes (to keep track of class assignments) the edges that connect from the student nodes to the class itself, of the graph that forms the maximum flow performed by Edmond's Karp. We can trust that the Max-Flow finding algorithm precisely calculates the correct max-flow pathing, and the matching/paths that it constructs is the most optimal. We just add the names of every connection of students enrolled in the class, given by the max flow graph, to the class itself.

Prove the Edmond's Karp Algorithm

We can assume that the algorithm works, as suggested by Kempe's Lectures and potential references to online sources. We can assume that there is a proof for it. The reason why we used Edmond's Karp is to achieve the polynomial runtime desired, a little subtle hint given by Piazza @454.

Proof Completed.