Problem Set 3 Solutions

This problem set is due **at 11:59pm** on **Thursday, September 29, 2016.**

EXERCISES (NOT TO BE TURNED IN)

Minimum Spanning Trees

- Do Exercise 23.1-5 in CLRS on page 629.
- Do Exercise 23.2-8 in CLRS on page 637.

Max Flow

- Do Exercise 26.1-2 in CLRS on page 713.
- Do Exercise 26.1-6 in CLRS on page 714.
- Do Exercise 26.2-10 in CLRS on page 731.

Problem 3-1. Crazy Ben [50 points]

Ben Bitdiddle has gone insane and has stolen a nuclear warhead, with which he plans to incite global nuclear warfare. He will launch the missile in exactly T hours from a nearby city. Your mission is to deter the missile by ambushing the launch site.

Specifically, you must send military trucks carrying troops from a base in City A (source) to the launch site in City B (sink). You are given a map $G = (V, E)$, with cities $c_i \in V$ and roads $r_j \in E$ connecting them. Each road takes t_j hours to traverse, and to prevent suspicion, trucks on the same road must be spaced at least 1 hour apart.

Note that:

- T and all t_j's are integers.
- All roads are one-way (directed edges).
- Each truck is either stationary or moving at a constant speed at any given time. All trucks travel at the same speed.
- Trucks may not stop in the middle of a road. They may only stop in cities.
- Trucks arrive and depart from cities on the hour. (Everything is an integer.)

(a) [30 points] Construct a flow network such that its maximum flow is equal to the maximum number of trucks you can send from City A to City B in T hours. Consider making multiple copies of the map G in constructing your network.

Solution: Make $T + 1$ copies of V, one for every hour $\{0, ..., T\}$. Let c_i^m be the copy of vertex c_i for hour m. Place an edge of capacity 1 between a pair of vertices (c_i^m, c_k^n) if there exists a road r_j connecting them in the original graph and if $n - m = t_j$. Place an edge of infinite capacity between every pair of vertices (c_i^m, c_i^{m+1}). Designate the copy of City A for hour 0 as the "source" and the copy of City B for hour T as the "sink."

Intuitively, you are constructing a "timeline," where flow across an edge represents a truck traveling between two cities. Since road r_j takes t_j hours to traverse, its incident vertices must lie on copies of the graph spaced t_j hours apart. Each of the inter-city capacities is 1 since only 1 truck can depart a city at a time (trucks must be spaced 1 hour apart on a road). Capacities between copies of the same city in adjacent hours of the timeline are infinite since any number of trucks can wait in a city.

Alternate Solution:

For every r_j, we can introduce "road vertices" for each hour on the road $(t_j - 1)$. Call this new graph $G' = (V', E')$. Then, we can copy G' and simply connect every $v_i \in V'$ on level i to $v_j \in V'$ on level $i + 1$ if there exists an edge between v_i and v_j in E' (i.e. we no longer have to jump t_j levels between cities).

(b) [5 points] What is the runtime of EDMONDS-KARP on your flow network from part (a)?

Solution: There are $O(T|V|)$ vertices and $O(T(|V| + |E|)) = O(T|E|)$ edges (vertices with no incident edges are negligible in flow algorithms, so $|E| = \Omega(|V|)$). Therefore, the runtime of EDMONDS-KARP on this flow network is $O(T|V|(T|E|)^2) = O(T^3|V||E|^2)$.

For Alternate Solution: There are $O(T|V|+T|E|) = O(T|E|)$ vertices and $O(T|V|+T^2|E|) = O(T^2|E|)$ edges (see rationale above). Therefore, the runtime of EDMONDS-KARP on this flow network is $O(T^5|E|^3)$.

Turns out Ben has placed covert operatives in each city to monitor suspicious activity. To prevent blowing the operation, each city c_i can only contain d_i trucks at any given time, **even if the trucks are momentarily passing through**.

(c) [15 points] Modify your flow network from part (a) to accommodate this new constraint.

Solution: This new constraint is equivalent to introducing vertex capacities. Instead of having infinite capacity edges between pairs of vertices (c_i^m, c_i^{m+1}), do the following: split each vertex c_i into two vertices – an "entrance" c_{i_1} and an "exit" c_{i_2} – and connect them with an edge of capacity d_i going from c_{i_1} to c_{i_2}. Do this for all copies of each vertex on every level (hour) of the graph. Edges originally entering c_i should now be entering c_{i_1}, and edges originally exiting c_i should now be exiting c_{i_2}. Finally, add an edge of infinite capacity between each pair of vertices $(c_{i_2}^m, c_{i_1}^{m+1})$, i.e. connect the exit vertex of a city to the entrance vertex of the same city on the next level.

Potential Wrong Answer: Change the capacity between city c_i on adjacent hours from infinity to d_i. This would work if the trucks passing through did not add to the number of trucks in the city. However, the bolded statement indicates that even trucks passing through contribute to the number of trucks occupying a city, therefore this solution does not work.

Problem 3-2. Dogs and Machine Learning [50 points]

Fakebook Inc. has developed a machine learning tool to help users differentiate between pictures of dogs and friends who look like dogs (apparently a common and embarrassing misconception). In order to deploy the tool, images must be transferred from the central image server (source) to the machine learning server (sink) through a "channel," i.e. a collection of intermediate servers. As an engineer at Fakebook Inc., your job is to test the throughput (in images/second) of this distributed "channel" system.

The channel servers (including the source and sink) are configured such that a given server $s_i \in V$ can forward images to other servers $s_j \in V$ through connections $c_{ij} \in E$. All connections are one-way (directed edges) and each connection has a maximum throughput of t_{ij} images/second. Furthermore, each server s_i can only handle up to p_i incoming images/second, except for the machine learning server (sink) which can handle any input rate.

(a) [15 points] Construct a flow network such that the maximum flow is equal to the maximum throughput of the channel.

Solution: The p_i constraint is equivalent to introducing vertex capacities. Construct a flow network from V and E, where the capacity of edge c_{ij} is t_{ij}. Now, split each vertex s_i into two vertices – an "entrance" s_{i_1} and an "exit" s_{i_2} – and connect them with an edge of capacity p_i going from s_{i_1} to s_{i_2}. This edge between the entrance and exit caps the intake rate of the server to p_i images/second. All edges coming into the original vertex s_i from different servers should now be coming in to the "entrance" vertex s_{i_1}, and all edges coming out of the original vertex s_i should now be coming out of the "exit" vertex s_{i_2}.

(b) [10 points] After you run EDMONDS-KARP on your network from part (a) and generate a flow graph, you notice that a single connection c_{ij} has slightly improved, increasing its maximum throughput value from t_{ij} to $t_{ij} + 1$. How can you update your max flow value to accommodate this small change (using the generated flow graph) in $O(|V| + |E|)$ time?

Solution: Since this change can only introduce at most 1 extra unit of flow, simply search for another augmenting path (using BFS or DFS) on the flow graph after incrementing the capacity of edge c_{ij}. If there exists no augmenting path, then the extra throughput of that connection does nothing and we do not update the max flow value. If there exists an augmenting path, then we can push exactly 1 extra unit of flow through c_{ij}, so we increment the max flow value by 1.

The channel engineers are satisfied with the results of your throughput test, but now they are concerned about the number of connections an image must traverse to get from the source to the sink.

Each image now carries an auxiliary variable – a "hop-count" – that is incremented every time the image traverses a connection between two servers. The source server will send a single wave of images, which all start with a hop-count of 0. Your task is to figure out how many of these images can reach the sink with a hop-count of at most M.

To simplify the scenario, each connection c_{ij} can only be used to send a single image per hop-count value. For example, if server s_a transfers an image with a hop-count of 6 to server s_b (through

connection c_{ab}), then server s_a cannot send another image with hop-count of 6 to server s_b ever again.

Note that in contrast to part (a), we are no longer concerned with *rate*, but rather the *number* of images that can reach the sink. As such, the p_i and t_{ij} values from part (a) are now irrelevant.

(c) [25 points] Construct a new flow network such that the maximum flow is equal to the maximum number of images that can be transferred from source to sink under these new constraints. Consider making multiple copies of G in constructing this network.

Solution: Make $M + 1$ copies of the graph in part (a), one for every hop-count $\{0, ..., M\}$. Let s_i^m be the copy of vertex s_i for hop-count m. Place an edge of capacity 1 between a pair of vertices (s_i^m, s_j^{m+1}) if there exists a connection $c_{ij} \in E$ in the original graph. Designate the copy of the central image server (original source) on hop-count level 0 as the new source vertex. Finally, add a new sink vertex that is connected to all copies of the machine learning server (original sink).

Problem 3-3. Pineapple Juice and Unicorns in Fantasyland [50 points]

The inhabitants of Fantasyland need to drink pineapple juice to feel happy. Pineapple juice is served at juice stands scattered throughout Fantasyland, and magical unicorns, which are endemic to Fantasyland, transport pineapple juice. To stock every juice stand properly with pineapple juice, every juice stand needs to be connected to every other juice stand through some sequence of unicorns.

Each unicorn U_i moves between a unique pair of juice stands, and the Queen of Fantasyland has to feed g_i bales of magical grass per year to each unicorn in exchange for its transportation services. Furthermore, each value of g_i is distinct.

The Queen enlists a set of unicorns, whom she dubs the **Royal Flushers**, such that the Royal Flushers can properly stock every juice stand and the amount of magical grass she has to feed them is minimized. The unicorns who are not a part of the Royal Flushers feed themselves.

In other words, Fantasyland can be thought of as a graph $G = (V, E)$, where the n juice stands are the vertices V and the m unicorns U_i are the edges, with g_i being the edge weights. The **Royal Flushers** are a minimum spanning tree T on G.

(a) [15 points] One day, Rarity the unicorn, who is not a part of the original Royal Flushers, lowers the amount of magical grass she is requesting by x bales.

The Queen wants to check whether or not she can properly stock every juice stand and use less magical grass than before by changing the set of unicorns in the Royal Flushers. Provide an algorithm that does so in $O(n)$ time.

Hint: For all parts of this problem, you may use the *cycle property* of minimum spanning trees without proof. The *cycle property* states that for any cycle C in the graph G, if the weight of an edge e of C is larger than the individual weights of all other

edges of C, then e can not belong to a minimum spanning tree. The proof is available in the recitation notes.

Solution: The question can be reduced to the following: If T is a Minimum Spanning Tree (MST) and an edge $e = (a, b)$ not in the MST has its weight decreased by x, how can we check if T is still an MST in $O(n)$ time?

To do so, run BFS on the vertices and edges of T to find a path from a to b in the MST. This path, combined with the edge e, forms a cycle. We can then apply the cycle property of MSTs, which states that the highest weight edge in a cycle in G can not be part of an MST in G. If e is the highest weight edge in the cycle, T is still an MST. If e is not the highest weight edge in the cycle, then T is no longer an MST because replacing the highest weight edge in the cycle with e results in a spanning tree with lower weight than T.

The BFS takes $O(|V| + |E|) = O(n)$ because the number of vertices and edges in T are both $O(n)$.

(b) [20 points] The Queen realizes that after Rarity's change of heart, she can still properly stock every juice stand while feeding unicorns less grass than before. Unfortunately, she doesn't know which set of unicorns she should enlist to minimize the amount of magical grass she is feeding them. Help her find the new optimal set of unicorns to enlist in $O(n)$ time and prove that your solution is correct.

Hint: Show that unicorns other than the original Royal Flushers and Rarity will not be in the new Royal Flushers.

Solution: First, other than edge e, I claim that any edges in G but not in T can not be in the new MST.

For any edge l in G but not in T, we can imagine appending it to the tree T, which forms a cycle that includes l. Then l must be the unique highest-weight edge in the cycle because otherwise adding l and removing another edge along the cycle would result in a spanning tree of lower weight than T in G, which contradicts the fact that T is an MST. The cycle property of MSTs then tells us that l can not be in any MSTs. Thus, only the edges of T and the edge e could possibly be in the new MST.

Now, as before, we can find the path from a to b. We then add the edge e to the MST T to form a cycle. The fact that the MST needs to be changed means that the new edge e is not the highest weight edge in the cycle. Then, we claim that removing the highest weight edge from the cycle results in the new MST. This is because the vertices of the cycle must be connected by edges in T or e, and the only way to connect them is to keep all but one edge in the cycle.

(c) [15 points] Inspired by Rarity's example, many other unicorns who were not a part of the original Royal Flushers decide to lower the number of bales of magical grass they

are requesting. Specifically, the n unicorns $U_{i_1}, U_{i_2} \dots U_{i_n}$ lower their magical grass requests from $g_{i_1}, g_{i_2} \dots g_{i_n}$ to $h_{i_1}, h_{i_2} \dots h_{i_n}$.

Just as before, the Queen wants to properly stock all juice stands while minimizing the total amount of magical grass she has to feed the unicorns that she enlists. Help her find the optimal set of unicorns in $O(n \log n)$.

Solution: Recall from the solution to part (b) that if an edge did not have its edge weight changed or was not part of the original MST, it will not be in the new MST. Thus, there are a total of $n + n = 2n = O(n)$ edges that could possibly be part of the new MST. Using Prim's or Kruskal's algorithm on these edges, we can find an MST in $O(n \log n)$ time.

Problem 3-4. Election Connection [50 points]

The 2016 Presidential Election is fast approaching, and it's more important than ever for Presidential Candidate Dillary Clump to be prepared. In particular, Candidate Clump would like to be able to communicate with various field offices across the country.

Clump has n field offices, and a cable exists between each pair of offices. Each cable has a unique daily usage cost. Each field office can be thought of as a vertex $v_j \in V$ in a graph $G = (V, E)$, and cables can be thought of as edges $e_i \in E$ with weights equal to their usage costs d_i. In order for Clump to succeed in the election, Clump must pay money to use a certain set of cables so that each field office is connected to each other field office through some sequences of used cables. Each cable only costs Clump money when it is being used.

The Clump Campaign has limited funds, so it has chosen to connect the field offices by using a set of cables that minimizes the total daily usage cost of the cables. This set of used cables is a minimum spanning tree T on the graph G.

During a campaign stop, a new field office is established by the Clump campaign in Fantasyland. The daily usage costs of connecting this new field office to the existing field offices $v_1, v_2 \dots v_n$ are $c_1, c_2 \dots c_n$. The Clump Campaign wants connect the last field office to the existing set of field offices. As before, it wants to choose a set of cables that minimizes the total daily usage cost of the cables.

(a) [10 points] A summer intern working for the Clump Campaign suggests that in order to minimize total cable usage costs, the new field office should be connected to the existing set of field offices by using the cable with the lowest value of c_i. Show that the summer intern's idea doesn't always result in the minimum total cost for the Clump Campaign.

Solution: This problem can be phrased as follows: Suppose we have an MST T on n vertices. A new vertex v is added, and we want to show that appending the minimum cost edge from v to T does not necessarily result in an MST on $n + 1$ vertices. For

this problem, it is easy to provide a counterexample. For example, if T consists of two vertices A and B connected by an edge of weight 3 and the new vertex C has edges of weights 1 and 2 connecting it to A and B respectively, then the correct MST on all 3 vertices uses the edges (B, C) and (A, C) and has weight 3 while the summer intern's algorithm uses the edges (A, B) and (A, C) and has weight 4.

Clump hires you, an MIT student, to find the most cost-efficient way to connect all of the field offices, including Fantasyland.

(b) [20 points] Prove that only the original set of cables used by the Clump Campaign and the n cables connected to Fantasyland can possibly be part of the most cost-efficient set of cables to use.

Hint: You may use the *cycle property* of minimum spanning trees without proof. The *cycle property* states that for any cycle C in the graph G, if the weight of an edge e of C is larger than the individual weights of all other edges of C, then e can not belong to a minimum spanning tree. The proof is available in the recitation notes.

Solution: First, let us ignore the new edges that have Fantasyland as one of their endpoints. I claim that any edges in G but not in T can not be in the new MST.

For any edge l in G but not in T, we can imagine appending it to the tree T, which forms a cycle that includes l. Then l must be the unique highest-weight edge in the cycle because otherwise adding l and removing another edge along the cycle would result in a spanning tree of lower weight than T in G, which contradicts the fact that T is an MST. The cycle property of MSTs then tells us that l can not be in any MSTs. Thus, only the edges of T and the edges with Fantasyland as one of its endpoints could possibly be in the new MST.

(c) [20 points] Help the Clump Campaign find the minimum cost set of cables to use in $O(n \log n)$ time.

Solution: Note that any edges not in the original MST will not be edges in the new MST. There are n edges in the original MST and there are n edges added as a result of the new vertex. Thus, we are trying to find an MST on a graph with $n + 1$ vertices and $2n$ edges. Using Prim's or Kruskal's algorithm, this procedure takes $O(n \log n)$ time.