- The assignment is due at Gradescope on Friday, Mar. 3 at 6:00p.
- You can either type your homework using LaTex or scan your handwritten work. We will provide a LaTex template for each homework. If you writing by hand, please fill in the solutions in this template, inserting additional sheets as necessary. This will facilitate the grading.
- You are permitted to study with up to 2 other students in the class (any section) and discuss the problems; however, *you must write up your own solutions, in your own words*. Do not submit anything you cannot explain. If you do collaborate with any of the other students on any problem, please do list all your collaborators in your submission for each problem.
- Similarly, please list any other source you have used for each problem, including other textbooks or websites. *Consulting problem solutions on the web is not allowed.*
- *Show your work.* Answers without justification will be given little credit.

PROBLEM 1 (35 POINTS) *Do K-T Chapter 7, Exercise 12 (on deleting k edges to reduce max flow value).*

Solution:
Input: $G = (V, E, s, t, c), k$, where G is a direct graph with $G = (V, E)$, with source node s, sink node t, edge capacity c, where $c_e = 1$ for every $e \in E$. Also we have k, which is amount of edges we can delete.
Output: Reduce the max flow in G by as much as possible. Delete the set of F edges from G, where $|F| = k$

Algorithm: $G = (V, E, s, t, c), k$
edge case checking
  if $k \geq |E|$ **do:**
    return E

  Run Edmonds and Karp algorithm to find residual graph G_f, with respect to G
  Compute minimum $s - t$ cut using G_f. Partition these sets as S and T, and add all edges that cross the partition to min cut-set M_c

  $F = \emptyset$

more edge cases
  if $|M_c| \leq k$ **do:**
    while $|M_c| \neq k$ **do:**
      let e be some edge $\in E$ and $\notin M_c$
      $F = F \cup e$
      $E = E - e$
      $k = k - 1$

    return $F \cup M_c$

  for i from 0 to k **do:**
    let e be some edge $\in M_c$
    $M_c = M_c - e$
    $F = F \cup e$

  return F

Proof of Correctness:
Lemma: Removing all k edges that cross the same minimum $s - t$ cut, M_c with respect to G, will always reduce maximum $s - t$ flow for G by as much as possible
First, note that in the design of this problem, capacity $c_e = 1$ for every $e \in E$. Second, note that if M_c is a minimum cut for G, and f is a max flow, $f = c(M_c)$, where $c(M_c)$ is the sum of the capacity of edges across the

minimum cut. As we want to minimize f, we can achieve this through minimizing capacity across M_c. As $c_e = 1$ for every $e \in E$, then $f = c(M_c) = |M_c|$. Therefore in this problem, max flow is bounded by the number of edges in the minimum cut.
If we remove all k edges from M_c we have $f = \max(|M_c| - k, 0)$. Note that we include the max with 0 as it is impossible to have negative max flow, and we want to consider the case of $k > |M_c|$, an edge case which is considered in the algorithm.

Note that the only modifications we are making to the graph of G is removing edges, and for any graph G' generated by removing edges from G, let the max flow $f' = M_c'$, where M_c' is the minimum cut for G'. Therefore we must show that for any G, $|M_c| - k \leq |M_c'|$ for any G', or that removing k edges from the min cut M_c will always result in M_c remaining the min cut for any other graph G', where G' is any G with k less edges.

Now we will prove that all edges should be removed from M_c by contradiction. Assume we can reach a smaller maximum flow through a strategy of removing edges where we do not remove all edges from M_c, or that we have $|M_c| - k > |M_c'|$, where M_c' is a min cut set generated by this other strategy of removing edges. However if M_c' is a min cut for G', then M_c' must be the reduction of some initial cut in G (let this initial cut be defined as $I(M_c')$). Note also that at most this cut can be reduced by k, if we remove every edge from $I(M_c')$. Therefore, we have $|I(M_c')| - k \leq |M_c'|$. Note finally, that M_c is the minimum cutset with respect to G, so by definition, we have $|M_c| \leq |I(M_c')|$. From here, we have

$|M_c| - k \leq |I(M_c')| - k \leq |M_c'|$ or $|M_c| - k \leq |M_c'|$

This proves that we cannot create a smaller min cut for any G' reduced from G with a strategy other than from removing k edges from M_c. Therefore, removing k edges from M_c reduces max flow f for G the most.

From this **Lemma**, we have essentially proved the correctness of our algorithm, as we find M_c through the residual graph G_f and then remove k edges to which reduces the max flow by as much as possible.

Runtime Analysis
First we check $|E|$ and k, which is $O(1)$. Next we run Edmonds and Karp algorithm to find residual graph G_f. This takes $O(|V||E|^2)$ run time. Next we compute the minimum $s - t$ cut using G_f, where we will do a search to find all vertices that are reachable from s through G_f and put them in S. All unreachable vertices in G_f are in T. This will be our minimum cut, M_c, and all edges that cross $S - T$ will be added to M_c. Given G_f from the previous step, this step will be worst case $O(E + V)$ run time. Now we have

$O(|V||E|^2) + O(E+V)$ at this point. From here we will enter one of the two loops, based on if $|M_c| \leq k$ or not. If we enter the first loop, we do a number of constant set operations to build the set of f, and this loop runs worst case $E-k$ iterations. The other loop does similar constant set operations and runs in worst case k iterations. Regardless, both $E \geq E-k$ and $E \geq k$ (based on the condition that we enter the second loop), so these loops are both upper bounded by E, so we can assume $O(E)$ run time, in worst case of either loop. This gives us $O(|V||E|^2) + O(E+V) + O(E)$, reducing to $O(|V||E|^2)$ which is overall run time.

PROBLEM 2 (35 POINTS) *Do K-T Chapter 7, Exercise 24 (on unique minimum cuts).*

Solution:
Input: $G = (V, E, s, t, c)$, where G is a direct graph with $G = (V, E)$, with source node s, sink node t, and a non-negative integer edge capacity c which gives some capacity c_e for every $e \in E$. *Also note that we are assuming that c is an integer weight function, in line with the work we've done in class. The same approach works for rational numbers if we increment by the minimum amount an edge can increase by, instead of increasing by* 1 *here, but we assume integer increasing in this case.*

Output: Return if G has a unique minimum $s - t$ cut or not.

Algorithm: $G = (V, E, s, t, c)$

Run Edmonds and Karp algorithm to find residual graph G_f, with respect to G

Compute minimum $s - t$ cut using G_f. Partition these sets as S and T, and add all edges that cross the partition to M_c

for every $e \in M_c$
  $c_e = c_e + 1$
  Find max flow f' through Edmonds and Karp with respect to G and the new capacity value c_e for e

  if $f' = f$ **do:**
    return **min-cut is not unique**
  else:
    $c_e = c_e - 1$

return **min-cut is unique**

Proof of Correctness: Before we consider the algorithm, we will consider the following observation, which is essentially an expanded definition of "uniqueness" of a minimum cut which incorporates that fact that maximum flow equals the capacity across a minimum cut: **In order for G to have a unique minimum $s - t$ cut, M_c, there cannot be any other $s - t$ cut in G, say M_c', such that for the maximum flow f with respect to G, $f = c(M_c')$ AND $M_c' \neq M_c$, or M_c' is not the same set of edges as M_c.**

In our approach we will increase each edge in turn and check the max flow, to see if M_c is unique. In order to show uniqueness we will first find a minimum cut with respect to the initial graph, G. Finding a minimum cut can be done by computing the residual graph from Edmonds and Karp and then finding all vertices reachable from the source node s. All edges which are from a reachable vertex from s to a non-reachable vertex will be added to M_c. Therefore, we know M_c contains the edges of an $s - t$ minimum cut-set.

Now we must determine if M_c is the only minimum cutset. From here, we will consider every edge in M_c and increment its capacity by 1, and calculate the changed max flow, f' with respect to the changed edge. Note that **we only change a single edge at a time**, so any modified graph which we will run max flow on is identical to G except that for some edge $e \in M_c$, $c(e) = c(e) + 1$. After we modify our given edge e, we find max flow f' with respect to G and the modified edge, which we will define as G'. Again, max flow can be found through Edmonds and Karp.

At this point in the loop we have two options, either $f' = f$ or $f' \neq f$ for any given iteration. Our algorithm follows the logic below, and with correctness proved in case analysis.

If $f' = f$ for any iteration of the loop
Then we know that M_c cannot be unique, as we have increased the capacity of one of the edges in M_c by 1 and still found the same amount of max flow, f' in the changed graph.

Let M_c' be the value of minimum cut value with respect to the modified graph G', which is the original graph G where one of the edges in M_c has increased by one. We know $f' = c(M_c')$, because max flow equals min cut, and we have $f' = f$ based on the condition, then we have $c(M_c') = f' = f$, and **therefore** $f = c(M_c')$. We also know that for G, $c(M_c) = f$, but for G', $c(M_c) = c(M_c) + 1$ from the increased edge capacity, so $c(M_c) \neq f$ for G'. This implies that M_c and M_c' must have at least one edge in difference, as $c(M_c) \neq f = c(M_c')$, so we have $c(M_c) \neq c(M_c')$, and **therefore** $M_c \neq M_c'$.

Note that we have found M_c' on a modified graph G', and we will now prove that M_c' is a possible cut for G, because the capacity is the same for $M_c' \in G$. This is true because $G = G'$ for every edge except e, and we will prove that $e \notin M_c'$, which implies M_c' is a possible min cut for G. Let $I(M_c')$ be the initial value of the capacity of M_c' in G. If $e \in M_c'$, we would have $M_c' = I(M_c') + 1$ for G', and then $M_c' \neq f$, because f will be bound by $I(M_c')$ in G, so $I(M_c') = f \neq M_c'$. This is a contradiction of $M_c' = f$ that we proved above, so therefore $e \notin M_c'$. Therefore, this implies that for every edge in M_c' the capacity is the same with respect to G, meaning it could be a cutset for G.

Therefore we have shown that $M_c \neq M_c'$ and $f = c(M_c')$ for $M_c' \in G$, which proves that if $f' = f$ there is another minimum cut in G. Therefore, we know that M_c is not unique, and return **min-cut is not unique**.

If $f' \neq f$ for every iteration of the loop
Then we know that M_c must be unique as we cannot achieve the same max flow switching out any edge in M_c. *Note that for every edge we consider, if we have $f' \neq f$, we subtract 1 after to revert the edge to its initial value after the comparison, and continue to iterate through the list of M_c edges. Therefore, we have G' as G but with one edge in M_c with capacity increased by one.*

If we have $f' \neq f$ for every $e \in M_c$, then we know that M_c is unique. This is because, if M_c is not unique, there must be a way to calculate the same max flow for G without using every edge in M_c, which means for some edge $e \in M_c$, increasing its capacity by 1 will not increase the max flow on G', because the Edmonds and Karp algorithm will bound f' by the min cutset M_c', where $M_c' \neq M_c$. As we know the algorithm always returns flow bounded by the min cutset, if we never have $f' = f$ for any $e \in M_c$, then there cannot be any $e \in M_c$ that can be exchanged out to form a different cut set which is also a min cutsets. In other terms, there cannot be any min cutset M_c' such that $M_c' \neq M_c$ and $c(M_c') = f$, proving M_c is a unique cutset. As it is impossible to find the same max flow through using a different set of the edges than M_c, we can return **min-cut is unique**.

These cases essentially prove the algorithm and as we have proven both of our cases, and in our implementation we check every edge in M_c this algorithm will correctly return if G has a unique minimum $s - t$ cut or not.

Runtime: First we find max flow and residual graph with respect to the unchanged graph G. This can be done in one iteration of Edmonds and Karp, which is $O(|V||E|^2)$. Next we compute a minimum $s - t$ cut on G_f which is done in $O(|V| + |E|)$ run time. Therefore, before entering the loop we have $O(|V||E|^2) + O(|V| + |E|)$ run time. We have worst case $M_c = E$, so we have worst case E iterations for the for loop. Within each iteration we increment the capacity of a given edge and then run the Edmonds and Karp algorithm on a modified graph, which is $O(|V||E|^2)$. Therefore, in the worst case the overall loop iterations costs $O(|V||E|^3)$. This gives us an overall run time of $O(|V||E|^2) + O(V + E) + O(|V||E|^3)$, which reduces to $O(|V||E|^3)$, which is polynomial.