1. (30 points) **(Finishing the Arborescence proofs)** You will complete the formal arguments for the proofs we only sketched in class.

   (a) (5 points) Recap the primal-dual algorithm (with reverse delete) for min-cost arborescence.

Solution:

Proof. Let $G = (V, E)$ be a graph where $e \in E(u, v)$. Given a root r, we have to find a directed tree such that all vertices $v \neq r$ ave a directed path from r.
Primal

$$Min : \sum_a c_a x_a \tag{1}$$

$$\sum_a x_a \geq 1 | \text{ a is in-arc into s}, \quad \forall s \subset V \ \& \ r \notin s \tag{2}$$

$$x_a \in \{0, 1\} \tag{3}$$

Dual

$$Max : \sum_s y_s \ \& \ r \notin s \tag{4}$$

$$\sum_s y_s \leq c_a | \text{s are subsets to which a is an in-arc}, \quad \forall a \in E(G) \tag{5}$$

$$y_s \geq 0 \tag{6}$$

□

   (b) (10 points) In the iterative step, recall that we find a minimal strongly connected component S which has incoming arcs in the current solution. If such a component exists and does not contain the root r, then we raise its dual variable y_S and proceed. Show that if we cannot find such a component, then the current solution (before reverse delete) is feasible.

Solution:

Proof. We will prove this by contradiction, assume the solution is not feasible. This implies that there is a subset of vertices who have incoming arcs less than 1, since x_a is either set to 0 or 1 in our algorithm eimplies that there exists a subset that has incoming arcs equal to zero.
Let this subset(call it s) be partitioned into connected components(called $s_1, s_2 \ldots$).
Choose s_1, this subset is strongly connected by our choice of s_1. And it has no incoming arcs as if the incoming arc was from $V \setminus s$, then s would also have an incoming arc which is not possible to our assumption. And an incoming arc can't come from $s \setminus s_1$ as they are separate connected components. Thus

we have found a strongly connected component with no incoming arc. By well ordering principle there is a subset which is minimal and strongly connected and has no incoming arcs. But our assumption was based on the fact that no such component can be found.
$\Rightarrow\Leftarrow$ □

(c) (15 points) Let F^* be the final solution after reverse delete. Then, show that for any variable $y_S > 0$ (i.e., it has strictly positive contribution) to the dual, then $|F^* \cap \delta^-(S)| = 1$, i.e., we satisfy the relaxed complementary slackness condition with $\lambda = 1$. This should use the property of reverse delete, and also how we choose the minimal strongly connected components at any time to raise the dual.

Solution:

Proof. Let there be a set S such that $|F^* \cap \delta^-(S)| > 1$. Since S was chosen to be frozen by our algorithm at some point as y_S 0it means that there was an in arc added let it be a_1, all other in-arcs must be added after a_1 in our reverse delete. We will prove that it must have been the case that all the in-arcs added after a_1 to S are useless and hence will be deleted. S is strongly connected means that we can use any edge whose existence was before a_1 to travel within S. Let a_2 be the second in-arc added after a_1. Reverse delete should delete this arc as it is useless.
Lets assume this arc is not useless hence it means that there is a vertex lets say u that must be connected to r using a_2. Since a_2 is used, the path is $r - u$. a_2 is made by joining two vertices as its an in-arc to S, the right most one lets say v is present inside S. The path now is $r - v - u$. We will claim that there exits an alternate path from r to v that does not involve a_2. We will use a_1 to arrive into S from r where we use the property of it being strongly connected and hence we can travel within S to reach v, this is an alternate path that does not require use of a_2. Hence our assumption is wrong and arc a_2 is useless and hence will be removed. Hence complimentary slackness condition is still satisfied. □

2. (15 points) **(Gap Example for Local Search)** In class, we saw that local search yields a 1/2-approximation for Max-k-Coverage. Now you will construct an example where it can be stuck at such a solution which is factor 1/2-off from the optimal.

(a) (15 points) Indeed, we said that if we start with any arbitrary collection of k sets, and keep making swaps as long as we improve the total coverage, we repeat until we stop. Construct an instance of max-k-coverage where, if we started off with a bad solution (you can choose this solution), the local search algorithm would not even make one improvement. That is, it stops there. Moreover, if this starting solution only covers 1/2 the number of elements of an optimal solution, then we would have shown a tight bad example for our local search analysis. (Hint: try to construct an instance where all the inequalities we used in our swap-based proof are almost tight. Indeed, if they were sloppy, then we could have done a better analysis).

Solution:

Proof. We will create a construction of $2k-1$ elements, such that they can be covered by k sets completely. But one particular starting solution can get stuck and choose only k elements.

$$U = \{e_1, e_2, \dots, e_{2k-1}\}$$
$$S = \{s_1, s_2, \dots, s_{2k}\}$$
$$\text{where,}$$
$$\forall i \in \{1, 2, \dots, 2k-1\}, s_i = \{e_i\} \ \& \ s_{2k} = \{e_1, e_2, \dots, e_k\}$$

Let the starting set be denoted by I. Let $I = \{s_1, s_2, \dots, s_k\}$. Clearly the optimal set is $O = \{s_{k+1}, s_{k+2}, \dots, s_{2k}\}$, as it covers all the elements. Bringing in a new set in I and removing one based on the swap in our local algorithm step we observe.
$I = \{s_1, s_2, \dots, s_k\}$, bringing in any set between s_{k+1} and s_{2k-1} instead of any set present in I, leads to exactly k individual elements. Bringing in s_{2k} instead of any set is useless too as it also covers exactly k elements(i.e all elements from 1 to k). As the algorithm is not making any improvement this answer is reported by our algorithm.

$$\frac{Alg}{Opt} = \frac{k}{2k-1}.$$
$$\Rightarrow \frac{Alg}{Opt} = \lim_{k\to\infty} \frac{k}{2k-1}$$
$$\Rightarrow \frac{Alg}{Opt} = \frac{1}{2}$$

The algorithm can't do better than 1/2 optimal as its approximation ratio. □

3. (10 points) **(Connectivity Problem)** Consider the following problem: we have a graph $G = (V, E)$, and edges have cost $c_e \geq 0$. Now, we have a set S of senders, and a set R of receivers such that $S \cap R = \emptyset$. The goal is to find a set of edges F with minimum total cost $\sum_{e\in F} c_e$ such that each receiver $r \in R$ is connected to at least one sender $s \in S$ (it can be any sender, doesn't matter which).

   (a) (10 points) Design a 2-approximation algorithm for this problem. You may reduce it to some problem we've already studied in class.

Solution:

Proof. To create a solution to this problem we convert it into a steiner tree problem and since in class we have studied a 2- approximation algorithm for steiner tree. Thus we use the two results to create a 2-approximation algorithm.

Conversion To Steiner Tree

Let $G' = (V', E')$, and edges have cost $c_e \geq 0$. G' is defined by joining all the sender vertices with another vertex called x. i.e $V' = V \cup x$ and $E' \supset E$. Let $Edelta' = (s_1, x) \cup (s_2, x) \ldots \cup (s_{|S|}, x)$. $E' = E \cup Edelta'$. The weights of the extra added edges are zero.

We will denote steiner tree problem by ST, and connectivity problem by CP.

The ST problem is solved with considering the source vertex as x, and the destination vertex set to be R. Let the solution be F' be achieved via a half approx obtained in class of the steiner tree problem. We first define how to create a CP solution from F'. $Let F = F' \backslash Edelta'$. Since the only vertex added is x and its edges have been removed from F'. Implies that all the edges originating only and only from G are present in F. Also F is a feasible solution because all receivers are connected to x in steiner tree. And hence they are connected to atleast one of the sender as they must have reached the tree source(vertex x) somehow. Also note that the $cost(F) = cost(F')$ as all the edge weights of Edelta' are zero. We also observe that the optimum must also be the same from our reduction and hence both are alpha-approximations. And since we have a 2-approximation in steiner tree we have a 2-approximation in CP.

□

4. (20 points) **(Some Non-Approximability Problems)** We saw in class that the Steiner Tree and Steiner Forest had 2-approximation algorithms. Now we show that a slight change to the problem makes them quite different. Suppose we have a vertex-cost version of the problem. That is, we have a graph $G = (V, E)$ and each vertex has a cost $c_v \geq 0$ (and edges have no cost). We are given a root $r \in V$, and a set of terminals $T \subseteq V$. The goal is to find a set of vertices $V' \subseteq V$ such that in the sub-graph induced by V' (i.e. take vertices in V' and all edges between any pair of vertices in V'), the root is connected to every terminal.

   (a) (5 points) Show that if we have an α-approximation for this problem, then we can use this to design an α-approximation for the Steiner Tree problem also.

Solution:

Proof. To prove an alpha approximation to the original graph we construct a new graph G' = (V,E) such that we remove weights from edges and for every removed weight we add a new vertex of the corresponding weight which is connected to the two ends of the edge whose weight was removed. i.e let edge $e_{u,v}$ be removed. Then add x such that $c_x = 0$ and $(u, x) \in E(G')$ and $(x, v) \in E(G')$. Also the original vertices in the graph must have weight zero. Since this new

problem is similar to the Non-Approximability Problem we have seen we have proved a reduction from Steiner Tree to this problem.
$\Rightarrow$If we found an α approximation for the new problem then we would have found alpha approxmation for steiner trees as the cost function still hasn't changed. cost(Steiner Tree) = Cost(Some approximability problem reduction). Also the optimum costs are same as the two problems are interconnected, and it is not possible for optimal solution of one to be better than optimal solution for other as cost function is same. Hence are reduction leads to same approximation factor. □

(b) (15 points) More interestingly, show that if we have an α-approximation for this problem, then we can use this to design an α-approximation for the Set Cover problem also. Using this and results mentioned in class, what is the factor of non-approximability you can prove for this problem?

Solution:

Proof. We will prove that there exists a reduction from Set-Cover to our problem, for if there exists a reduction it means that giving a alpha aaproximation to our problem would mean that we have given an alpha approximation to the set cover problem.
The Reduction

$$U = \{e_1, e_2, \ldots, e_m\}$$
$$S = \{s_1, s_2, \ldots, s_n\}$$
$$\text{where,}$$
$$\forall i, s_i \subset U$$

We now construct a graph $G = (V, E)$ where $V = U \cup S \cup s$, consider all elements in universe to be the destination vertex in our Graph and an extra source vertex is present and in the middle there are n vertices being represented by each set. For all $s_i \in S$ if s_i covers some elements join the corresponding edges. After doing this, join the source vertex(s) to all the n vertices in the middle. The cost function for the vertices is such that all the n vertices have cost 1, rest of the vertices have cost zero.
Now we have a graphical construction of the set cover problem, we now prove the approximation results, lets say there is a set cover of size k. First we observe that choosing a set from S is equivalent to choosing that vertex in our set V in steiner tree. If there is an alpha approximation for our problem then we have found a tree. All the vertices in tree that have a cost 1, or are in the middle, the corresponding set they belong to choose those sets. That will be our alpha approximation set cover.
We observe that the cost of our algo in steiner tree setting is same as the cost

of our algo in the set cover setting. Now if we prove the same optimum value we have given an alpha approximation.
The optimal solution for the tree has the same value as the optimal solution for set cover. As if the steiner tree had an optimal tree with cose less than set cover that would mean it has lesser vertices from the middle layer than the min set cover in its solution such that all elements on the right are covered. Which means there is a corresponding set cover solution which is lesser than the minimum set cover solution. A similar argument shows that a steiner tree minimum for our graph cannot be greater than a set cover minimum. As alg costs and opt costs are same the same α approximation. □