Assignment 1

Navneel Singhal

2018CS10360

Contents

1 Problem 1

1.1 Statement

Give a greedy algorithm that achieves an approximation guarantee of $O(\log n)$ for set multicover, which is a generalization of set cover in which an integral coverage requirement is also specified for each element and sets can be picked multiple number of times to satisfy all coverage requirements. Assume that the cost of picking α copies of set S_i is $\alpha \cdot cost(S_i)$.

1.2 Solution

TODO

2 Problem 2

2.1 Statement

Consider the following 2-approximation algorithm for the vertex cover problem. Find a depth first search tree in the given graph, G, and output the set, S, of non-leaf vertices of this tree. Show that S is indeed a vertex cover for G and $|S| \leq 2 \cdot \mathbf{OPT}$.

2.2 Solution

We shall use the abbreviation DFS-tree for depth-first-search tree in what follows.

Since the DFS tree is a rooted tree, we will use the definition of a leaf as a vertex which doesn't have any children in the DFS tree (since otherwise, the claim in the problem statement doesn't hold for graphs consisting of a single cycle, or a connected simple graph with 2 vertices).

Claim 2.1

S is a vertex cover for G.

Proof. Suppose S is not a vertex cover for G. Then there must exist an edge (u, v) with none of u, v in S, i.e., an edge between two leaves in the DFS-tree.

Note that the DFS-tree of any undirected graph partitions the set of edges of the graph into two sets - tree edges and back edges, where the tree edges correspond to the edges which appear in the DFS-tree, and the back edges (due to the properties of the DFS-tree) join a vertex to an ancestor of its own.

(u, v) can't be a tree edge, since for that to happen, either of u, v has to be the parent of the other in the DFS-tree, and since u, v are distinct leaves, this is a contradiction. It can't be a back edge either, since otherwise either of u, v has to be an ancestor of the other in the DFS-tree, which leads to a similar contradiction.

Hence we have shown that such a case can never arise, which shows that S is a vertex cover for G. □

Claim 2.2

The minimum vertex cover of a graph is at least as large as the minimum vertex cover of its DFS-tree.

Proof. For this, we note that the minimum vertex cover of a graph is also a vertex cover of its DFS-tree, since the set of edges of the DFS-tree is a subset of the set of edges of the graph itself, and hence any edge of the DFS-tree has an endpoint in the vertex cover of the original graph.

Hence the minimum vertex cover of a graph, being a vertex cover of the DFS-tree, is at least as large as the vertex cover of its DFS tree. □

Now if we show that the number of non-leaf vertices of a tree is at most twice the size of a vertex cover of a tree, we shall be done, since the size of S will then be at most twice the size of a vertex cover of the DFS tree, and hence at most twice the size of a vertex cover for the graph, which would complete the proof. Hence, it suffices to show the following claim:

Claim 2.3

For any tree, the size of the minimum vertex cover $VC(T)$ is at most twice the number of non-leaf vertices in the tree.

Proof. For this, we shall need the following two claims.

Claim 2.4

For a tree, there is always a minimum vertex cover which doesn't contain any leaf, but contains all vertices adjacent to at least one leaf.

Proof. Note that there is no edge between two leaves by the definition of leaves in a rooted tree.

Consider the set of edges that are incident on a leaf. Note that the number of vertices adjacent to a leaf is at most the number of leaves, since the degree of a leaf is at most 1 (to see why, note that the partial function from the set of leaves to the set of vertices adjacent to a leaf induced by the adjacency relation is surjective).

Now we do the following: replace each leaf in the minimum vertex cover with the vertex it is joined to by an edge. The resulting set is at most the size of the original vertex cover, and it is still a vertex cover, since the edges not incident to a leaf are still covered, while the edges incident to a leaf are now covered by a possibly different vertex.

Note that since this set is a vertex cover, it must consist of all vertices adjacent to a leaf, since otherwise the corresponding edge won't be covered by any vertex.

All in all, we have shown that we can modify a minimum vertex cover to another vertex cover which has no more vertices (hence it is also a minimum vertex cover) such that no leaf is in the vertex cover but all vertices adjacent to a leaf are, which proves this claim. □

Claim 2.5

Let $L(T)$ be the set of leaves of a non-empty tree T, and let $P(T)$ be the set of vertices adjacent to a leaf. Then $P(T) \cup VC(T \setminus (P(T) \cup L(T))$ is a minimum vertex cover of T (here $VC(\cdot)$ gives an arbitrary minimum vertex cover of a possibly empty tree (rooted or unrooted)).

Proof. From the previous claim, we know that there exists a vertex cover $VC(T)$ such that $P(T) \subseteq VC(T)$.

Note that removing $L(T)$ from T makes $P(T)$ the set of leaves of the resulting tree, so $T \setminus (L(T) \cup P(T))$ is still a tree.

Firstly we show that $VC(T) \setminus P(T)$ is a vertex cover of $T \setminus (P(T) \cup L(T))$. Consider any edge (u, v) in the tree formed by removing $P(T)$ and $L(T)$. It can't be incident to any vertex in $P(T)$ or $L(T)$ since we have removed these vertices. Since $VC(T)$ is a vertex cover, at least one of u, v must be in $VC(T)$. Since none of u, v is in $P(T)$, at least one of u, v must be in $VC(T) \setminus P(T)$. Since our edge was arbitrarily chosen, we are done with this part.

Now we show that $P(T) \cup VC(T \setminus (P(T) \cup L(T)))$ is a vertex cover of T. Consider any edge of T. If the edge is incident to a leaf of T, it is covered by $P(T)$. If the edge is incident to any vertex of $P(T)$, it is covered by that vertex in $P(T)$. If none of these is true, then both endpoints must be in $T \setminus (P(T) \cup L(T))$, and hence it is covered by a vertex in $VC(T \setminus (P(T) \cup L(T)))$. So the exhibited set is indeed a vertex cover.

Now we show that this is a minimum vertex cover. Suppose this is not a minimum set cover. By the previous claim, there exists a minimum vertex cover of T which contains $P(T)$. Then since we have shown that if we remove $P(T)$ from any minimum vertex cover of T, we get a vertex cover for $T \setminus (P(T) \cup L(T))$. Doing this to a minimum set cover of T which has $P(T)$ as a subset gives us a smaller vertex cover of $T \setminus (P(T) \cup L(T))$ than $VC(T \setminus (P(T) \cup L(T)))$, which is a contradiction. This completes the proof. □

From the claims above, we can note that the following algorithm gives us a minimum vertex cover of a rooted tree:

```
1:  function MinVertexCover(Rooted Tree $T$)
2:      let $A \leftarrow \{\}$
3:      let $i \leftarrow 1$
4:      while $T$ has at least 1 vertex do
5:          let $L_i \leftarrow$ the set of leaves of $C$
6:          let $P_i \leftarrow$ the set of vertices adjacent to a leaf of $C$.
7:          $A \leftarrow A \cup P_i$
8:          $T \leftarrow T \setminus (L_i \cup P_i)$
9:          $i \leftarrow i + 1$
10:     end while
11:     return $A$
```

```
12: end function
```

Consider the sets L_i and P_i. We can partition the set of vertices of T as $\cup_{i=1}^m (L_i \cup P_i)$ where m is the number of steps taken by the while loop. The corresponding minimum set cover is $\cup_{i=1}^m P_i$. The non-leaf vertices of this tree are $S = \cup_{i=1}^m (P_i \cup L_{i+1})$, where $L_{m+1} = \emptyset$ for the sake of convenience. Note that all unions here are disjoint unions.

As observed before, we have that P_i is the set of leaves of the tree formed when we remove L_i from T at the i^{th} step, and L_{i+1} is the set of leaves of the tree formed when we remove P_i from this tree. So by an argument similar to the one in the proof of Claim 2.4, we have that $|P_i| \geq |L_{i+1}|$. Hence we have

$$\begin{aligned}
\mathbf{OPT} &= | \cup_{i=1}^m P_i| \\
&= \sum_{i=1}^m |P_i| \\
&\geq \sum_{i=1}^m \frac{|P_i| + |L_{i+1}|}{2} \\
&= \frac{1}{2} \sum_{i=1}^m |P_i| + |L_{i+1}| \\
&= \frac{1}{2} | \cup_{i=1}^m (P_i \cup L_{i+1})| \\
&= \frac{|S|}{2}
\end{aligned}$$

Rearranging this gives $|S| \leq 2 \cdot \mathbf{OPT}$, as needed. □

3 Problem 3

3.1 Statement

In the *maximum coverage problem*, we have a set of elements E, and m subsets of elements $S_1, \ldots, S_m \subseteq E$. The goal is to choose k sets such that we maximize the number of elements that are covered; an element s is covered if it belongs to one of the sets picked. Give a $\left(1 - \frac{1}{e}\right)$-approximation algorithm for this problem.

3.2 Solution

TODO

4 Problem 4

4.1 Statement

In the *related machines scheduling problem*, we are given m machines and n jobs. Job j has processing time p_j and machine i has speed s_i. Machine i takes p_j/s_i time to complete job j. Given T design a polynomial time algorithm which either schedules jobs on the m machines such that they finish by time $2T$ or proves that there is no way of scheduling jobs so that they all finish by time T.

4.2 Solution

TODO