Problem 1. **(Reduction from Vertex Cover to Set Cover)** Prove: If the set cover problem admits a polynomial time algorithm (for finding an optimal solution), then the vertex cover problem admits a polynomial time algorithm (for finding an optimal solution).
Remark. Unless **P** = **NP**, the set cover problem does not admit a polynomial time algorithm.

Proof. Let $G = (V, E)$ be the input graph to the vertex cover problem. For each vertex $u \in V$, create a set S_u that includes all the edges of E incident on u. Now, define a collection of sets $\mathcal{S} = \{S_u | u \in V\}$. The universe U equals $\bigcup_{u \in V} S_u = E$. Let $V(\mathcal{C}) = \{u | S_u \in \mathcal{C}\}$.

Lemma. $(U, \mathcal{S})$ contains a set cover $\mathcal{C} \subseteq \mathcal{S}$ of size k if and only if $V(\mathcal{C})$ is a vertex cover of G.

Suppose that $\mathcal{C}$ is a set cover of size k. Consider $V(\mathcal{C})$ with size k. For any e, since $\bigcup_{S_u \in \mathcal{C}} S_u = E$, $e \in S_u$ in some S_u. Hence, $u \in V(\mathcal{C})$ and e must be covered, meaning that $V(\mathcal{C})$ is a set cover.

Suppose that $W \subseteq V$ is a vertex cover of G. Consider the collection $\mathcal{T} = \{S_{u'} | u' \in W\}$ with size k. For any $e = (u, v)$, either u or v must be picked. Without loss of generality, suppose that $u \in W$, then $e \in S_u$ and $S_u \in \mathcal{T}$. Hence, $\bigcup_{S_u \in \mathcal{T}} S_u = E$, meaning that $\mathcal{T}$ is a set cover.

Therefore, we can solve vertex cover optimally by finding an optimal set cover $\mathcal{C}$ and return $V(\mathcal{C})$. Suppose that $V(\mathcal{C})$ is not optimal, then there is an optimal vertex cover V^* corresponding to the set cover $\mathcal{S}^*$ with $|\mathcal{S}^*| < |\mathcal{S}|$, which is a contradiction. □

Problem 2. Let $\mathcal{C}^*$ be an optimal universe cover for the set cover problem. Consider running the set cover approximation algorithm and consider the moment before the i-th set S_i is chosen by the algorithm. Let $z_i = |U \backslash (S_1 \cup S_2 \cup \ldots \cup S_{i-1})|$ and $\mathcal{C} = \{S_1, S_2, \ldots, S_{i-1}\} \cap \mathcal{C}^*$. Prove:

- $\mathcal{C} \neq \mathcal{C}^*$.
- S_i has benefit at least $z_{i-1}/(|\mathcal{C}^* \backslash \mathcal{C}|)$.

Proof.

(a) Suppose the otherwise that $\mathcal{C} = \mathcal{C}^*$, then $\mathcal{C}$ has covered the whole universe U, which is a contradiction that S_i is picked by the algorithm.

(b) The benefit of S_i cannot be smaller than $|z_{i-1}|/(|\mathcal{C}^* \backslash \mathcal{C}|)$. Consider $\mathcal{C}' = \mathcal{C}^* \backslash \mathcal{C}$ and $\mathcal{F} = U \backslash (S_1 \cup S_2 \cup \ldots \cup S_{i-1})$. As every element of $\mathcal{F}$ must at least appear in one $S_i \in \mathcal{F}$, it follows the result by pigeonhole principle that at least one S_i has benefit more than $|\mathcal{F}|/|\mathcal{C}'| = z_{i-1}/(|\mathcal{C}^* \backslash \mathcal{C}|)$.

□

Problem 3. Let $\mathcal{I}$ be a set of n intervals in 1D space (i.e., each interval has the form $[x, y]$) and P be a set of n 1D points. A subset $S \subseteq P$ is a stabbing set of $\mathcal{I}$ if every interval of $\mathcal{I}$ covers at least one point in S. Let OPT be the size of the smallest stabbing set of $\mathcal{I}$, and it is guaranteed that OPT ≥ 1. Design an algorithm to find a stabbing set of size at most OPT $\cdot O(\log n)$. Your algorithm must run in time polynomial to n.

Solution. Let U be the set of intervals $\mathcal{I}$. Denote $S_p = \{I \in \mathcal{I} | p \in I\}$ and the collection $\mathcal{S} = \{S_p | p \in P\}$, which defines a set cover problem with $U = \bigcup_{S \in \mathcal{S}} S$. Furthermore, a subset $S \subseteq P$ is a stabbing set of $\mathcal{I}$ if and only if $\mathcal{C} \subseteq \mathcal{S}$ is a set cover of U. By running the greedy algorithm, a set cover with size $\text{OPT} \cdot O(\log n)$ can be obtained in polynomial time. To construct the stabbing set T, we consider every $S \in \mathcal{C}$, and add any $p \in P$ that satisfies $S = S_p$ to T.

Problem 4. Let $\mathcal{I}$ be a set of n intervals in 1D space (i.e., each interval has the form $[x, y]$). A subset $S \subset \mathbb{R}$ is a stabbing set of $\mathcal{I}$ if every interval of $\mathcal{I}$ covers at least one point in S. Let OPT be the size of the smallest stabbing set of $\mathcal{I}$. Design an algorithm to find a stabbing set of size at most $\text{OPT} \cdot O(\log n)$. Your algorithm must run in time polynomial to n.

Solution. It suffices to consider the $2n$ endpoints of each interval I. To see this, note that the set of lines stabbed by any point p does not change unless a boundary point is hit, and hence using a non-endpoint point p is no better than using an endpoint.

Now, apply the solution of Problem 3 with P as the $2n$ endpoints, the returned solution is a stabbing set that satisfies the requirement.

Problem 5. (Reduction from Hitting Set to Set Cover) Given an instance of the hitting set problem, explain how to convert it to a set cover problem.

Solution. Let $\mathcal{U}, \mathcal{S}$ be the universe, the collection of sets of the hitting set problem, respectively.

Define a bipartite graph G, where every left vertex corresponds to a set $S \in \mathcal{S}$ and every right vertex corresponds to a element of U. Furthermore, G has an edge from $S \in \mathcal{S}$ to $e \in U$ if and only if $e \in S$.

Solving the original hitting set problem is equivalent to finding a smallest set R of right vertices such that every left vertex is adjacent to at least one vertex in R.

For each $e \in U$, define N_e as the set of neighbors of e. Note that a left vertex S is in N_e if and only if $e \in S$. The set collection $\{N_e | e \in U\}$ defines a set cover problem, whose universe is the set of left vertices and has a size of $|\mathcal{S}|$. Let $\mathcal{C}$ be an optimal set cover of this problem. Then $H = \{e \in U | N_e \in \mathcal{C}\}$ must be an optimal hitting set for the original problem.

Problem 6. (Reduction from Set Cover to Hitting Set) Given an instance of the set cover problem, explain how to convert it to a hitting set problem.

Solution. Let $\mathcal{U}, \mathcal{S}$ be the universe, the collection of sets of the set cover problem, respectively.

Define a bipartite graph G, where every left vertex corresponds to a set $S \in \mathcal{S}$ and every right vertex corresponds to a element of U. Furthermore, G has an edge from $S \in \mathcal{S}$ to $e \in U$ if and only if $e \in S$.

Solving the original set cover problem is equivalent to finding a smallest set L of left vertices such that every right vertex is adjacent to at least one left vertex in L.

For each $e \in U$, define N_e as the set of neighbors of e. Note that $e \in N_i$ if and only if a left vertex N_i contains e. The set collection $\{N_e | e \in U\}$ defines a hitting set problem.

Find an optimal hitting set H of this problem. Then, the collection $\{S \in \mathcal{S} | S \in H\}$ must be an optimal set cover for the original problem.

Problem 7. In the hitting set problem, we are given a collection of sets $\mathcal{S}$, where each set $S \in \mathcal{S}$ is a subset of some universe U. We want to find a hitting set $H \subseteq U$ of the smallest size (recall that H is an hitting set if $H \cap S \neq \emptyset$ for every $S \in \mathcal{S}$). Let OPT be the size of an optimal hitting set. Design a polynomial time algorithm that returns a hitting set of size at most $\text{OPT} \cdot (1 + \ln|S|)$.

Solution. Create for every $i \in U$, $H_i = \{j | i \in S_j\}$. Let $U' = [|\mathcal{S}|]$ and the collection of sets $\mathcal{H} = \{H_i | i \in U\}$. Since a hitting set $H \subseteq U$ of size k exists if and only if a set cover of size k with $\mathcal{H}$ on U' exists, hence by running the set cover approximation algorithm, a set cover with size of at most $\text{OPT} \cdot (1 + \ln|S|)$ can be found. We can construct the hitting set H by consider the subset $\mathcal{T} \subseteq \mathcal{H}$ returned by the algorithm. For each $T \in \mathcal{T}$, we find an i that makes $H_i = T$, then we include i in the hitting set H.

Problem 8. Consider $\mathcal{S} = \{\texttt{arid}, \texttt{dash}, \texttt{drain}, \texttt{heard}, \texttt{lost}, \texttt{nose}, \texttt{shun}, \texttt{slate}, \texttt{snare}, \texttt{thread}\}$. Treat each word in $\mathcal{S}$ as a set of letters. Run the set-cover algorithm on $\mathcal{S}$.

Solution. $\mathcal{U} = \{\texttt{a}, \texttt{r}, \texttt{i}, \texttt{d}, \texttt{s}, \texttt{h}, \texttt{n}, \texttt{e}, \texttt{l}, \texttt{o}, \texttt{t}, \texttt{u}\}$.

Step 1. $\mathcal{F} = \mathcal{U}$. The word with the largest benefit 6 is `thread`.

Step 2. $\mathcal{F} = \{\texttt{i}, \texttt{s}, \texttt{n}, \texttt{l}, \texttt{o}, \texttt{u}\}$. `lost`, `nose`, `shun` all have the benefit of 3.
Without loss of generality, suppose that the algorithm picks `shun`.

Step 3. $\mathcal{F} = \{\texttt{i}, \texttt{l}, \texttt{o}\}$. `lost` has a benefit of 2.

Step 4. $\mathcal{F} = \{\texttt{i}\}$. `arid`, `drain` all have the benefit of 1.
Without loss of generality, suppose that the algorithm picks `drain`.

Hence, $\mathcal{C} = \{\texttt{thread}, \texttt{shun}, \texttt{lost}, \texttt{drain}\}$.

Problem 9. In each iteration of the set cover approximation algorithm, a set would be picked with the largest benefit.
Prove: if we lay out the sets in order they are picked, their benefits are non-ascending.

Proof. Suppose the otherwise that the benefits are not non-ascending, i.e. $|\mathcal{F}_i \cap S_i| > |\mathcal{F}_{i-1} \cap S_{i-1}|$. Then, since $\mathcal{F}_i = \mathcal{F}_{i-1} \backslash S_{i-1}$, $|\mathcal{F}_{i-1} \cap S_{i-1}| < |\mathcal{F}_i \cap S_i| \leq |\mathcal{F}_{i-1} \cap S_i|$, which is a contradiction since S_i must be picked in the $(i-1)$-th round instead. □

Problem 10. Give a counterexample input to show that the approximation ratio of the set-cover algorithm cannot be bounded by 2.

Solution. Construct a bipartite graph, where A is the set of n vertices, and $B_2, \ldots, B_n$ be sets of vertices where B_i has the size $|B_i| = \lfloor n/i \rfloor$. Each vertex in A is connected to exactly one vertex of B_i, for all i. Also, let the edges into B_i be uniformly distributed so that the degree of any vertex in B_i is as close to i as possible.

We construct a set cover instance by creating $2n$ sets corresponding to each vertex $v \in V$. For the set S_v which corresponds to a vertex $v \in V$, we let $S_v = \{e : e \text{ is incident on } v\}$. Observe that the collection $\mathcal{S} = \{S_v : v \in V\}$ with size of $2n$ forms an instance of the set cover problem with the universe E. Each set has the size equal to the degree of the vertex that it corresponds to.

Then, note that the degree of the vertex in B_n is n and the degree of the vertices in part A is $n-1$. Also, observe that B_n, with only one vertex, is the first to be removed, this subtracts one from the degree of each vertex in A, and hence also the benefit. We then have forced the algorithm to choose all sets that correspond to the vertices in part B, which has the size $\sum_{i=2}^{n} \lfloor n/i \rfloor$, while the optimal solution is to choose all vertices that corresponds to part A, which has the size of n. Set $n = 16$ and part B has 34 vertices, while part A has only 16 vertices. Hence, the best set cover has the size 16. The approximation ratio of this algorithm is hence 34/16, greater than 2.

Problem 11. Prove: Unless $\mathbf{P} = \mathbf{NP}$, the decision version of the set cover problem (decide whether there is a set cover $\mathcal{C} \subseteq \mathcal{S}$ such that $|\mathcal{C}| = k$) does not admit any polynomial-time algorithm.

Proof. Suppose the otherwise that the set cover decision problem has a polynomial time algorithm. Then, we can solve the vertex cover decision problem in polynomial time by transforming the instance to set cover problem. A graph contains a vertex cover of size k if and only if there is a set cover of size k on the collection $\{S_u : u \in V\}$, where $S_u = \{e : e \text{ is incident on } u\}$. Then, we can solve the vertex cover problem by incrementing k, check if there is a solution of size k, and stop as soon as we have found a solution. This can be done in polynomial time, hence contradicting the fact that there is no polynomial time algorithm for vertex cover unless $\mathbf{P} = \mathbf{NP}$. □

Problem 12. Let $\boldsymbol{M}_{n\times m}$ be a 0/1 matrix. It is guaranteed that every row of $\boldsymbol{M}$ has at least one 1. A set S of columns is a *column cover* if every row of $\boldsymbol{M}$ has a 1 in at least one column of S. If OPT is the minimum size of all column covers, describe a poly(n, m)-time algorithm that finds a column cover of size $O(\text{OPT} \cdot \log n)$.

Solution. We construct an instance of the set cover problem by letting the columns containing at least one 1 form a collection $\mathcal{C}$. Namely for column i, let $C_i = \{j : \boldsymbol{M}[j, i] = 1\}$, $\mathcal{C} = \{C_i : i \in [m]\}$, and the universe $U = [n]$.

Hence, a column cover can be found by running the set cover approximation algorithm in polynomial time and return the columns $C_i \in \mathcal{D} \subseteq \mathcal{C}$, which has the same size of the set cover $\mathcal{D}$ of at most $\text{OPT} \cdot (1 + \log n) = O(\text{OPT} \cdot \log n)$.

Problem 13. Show that the set cover algorithm is h-approximate, where $h = \max_{S\in\mathcal{S}}|S|$.

Proof. Suppose that the algorithm picks t sets, then at least one new element from U is discovered. For each $i \in [1, t]$, denote by e_i an arbitrary element that is newly covered when the i-th set is discovered.

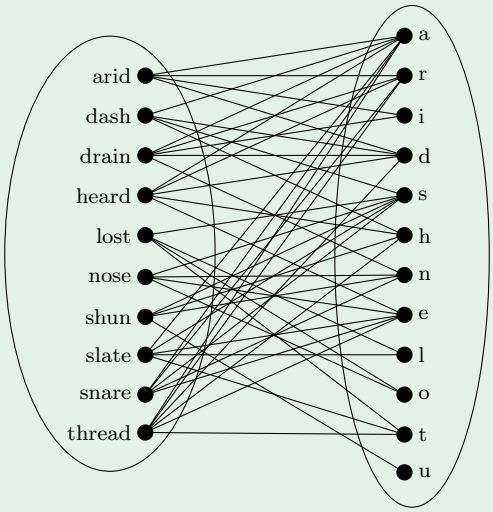
Let $\mathcal{C}^*$ be an optimal set cover. Since each e_i exists in at least one set of $\mathcal{C}^*$, we have

$$t = \sum_{i=1}^{t} 1 \le \sum_{i=1}^{t} \#\text{ sets in } \mathcal{C}^* \text{ containing } e_i \le \sum_{e\in U} \#\text{ sets in } \mathcal{C}^* \text{ containing } e = \sum_{S\in\mathcal{C}^*} |S| \le |\mathcal{C}^*| \cdot h.$$

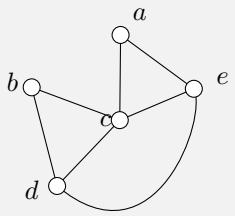
□

Problem 14. Consider $\mathcal{S} = \{\texttt{arid}, \texttt{dash}, \texttt{drain}, \texttt{heard}, \texttt{lost}, \texttt{nose}, \texttt{shun}, \texttt{slate}, \texttt{snare}, \texttt{thread}\}$. Given a set L of letters, we call L a hitting set if every word in S uses at least one letter in L. Our goal is to find a hitting set of the smallest size. Re-formulate the problem as a set cover problem.

Solution. Consider the bipartite graph below. Finding a hitting set L is equivalent to finding a subset of right vertices V such that every left vertex has to be adjacent to at least one right vertex in V. Hence, construct $S_v = \{w : \ell(v) \in w\}$, where $\ell(v)$ is the letter that v corresponds to and w is a word formed by a set of letters, for every right vertex v. Now, we can find a set cover on the collection $\{S_v : v \in V\}$ to solve the hitting set problem.



Problem 15. Let $G = (V, E)$ be a simple undirected graph. A 5-cycle is a cycle with 5 edges. We say that a subset $D \subseteq E$ is a 5-cycle destroyer if removing the edges of D destroys all the 5-cycles in G, namely, $G' = (V, E \backslash D)$ has no 5-cycles. For example, if G is the graph below, there is only one 5-cycle *acbdea*; a 5-cycle destroyer is $\{(d, e)\}$, and so is $\{(d, e), (c, d)\}$.



Let D^* be a 5-cycle destroyer with the minimum size. Design an algorithm to find a 5-cycle destroyer of size $O(|D^*| \cdot \log|V|)$ in time polynomial to $|V|$.

Solution. For every distinct order of vertices of size 5, we check if there is a cycle, which can be done in $O(|V|^5)$. Hence, the number of cycles of the length of 5 is at most $|V|^5$ and is listed in $O(|V|^5)$ time.

We reformulate the cycle destroyer problem as a hitting set instance. Note that removing an edge e from the graph essentially destroys a cycle containing e. Hence, for every cycle c_i, we construct a set S_i containing all the edges on c_i. Then, run the hitting set approximation algorithm to find a hitting set with size at most $\text{OPT} \cdot (1 + \ln n)$, which ensures that removing the edges from G breaks all 5-cycles (removing cycles does not result in more cycles).