# Problem Set 9

Ekkapot Charoenwanit
Efficient Algorithms

**Problem 1. NP-Completeness**

1) The Hamiltonian Cycle (Ham-Cycle) Problem

Given a directed graph $G = (V, E)$, a Hamiltonian cycle of $G$ is a simple cycle that contains each vertex in $V$. The Hamiltonian Cycle (Ham-Cycle) problem is to figure out if there is at least one Hamiltonian cycle in the given graph $G$. We can show that Ham-Cycle is NP-complete by reducing from the Vertex Cover problem.

Show that if Ham-Cycle $\in P$, then the problem of showing a sequence of vertices that form a Hamiltonian cycle in a given directed graph $G = (V, E)$ is also polynomial-time solvable.

2) Given a boolean formular $\phi$, show that the problem of determining whether the given boolean formular $\phi$ is a tautology is in the $co - NP$ class.

3) Prove the following claims:

(3.1) $P \subseteq NP$

(3.2) $P \subseteq co - NP$

**Problem 2. Approximation Algorithms**

1) Implement the approximation algorithm for the Vertex Cover problem in the lecture using an adjacency list and show that the running time of this implementation is polynomial in the input size $|V|$ and $E$.

2) Design a greedy algorithm for finding an optimal vertex cover given an undirected tree. Give an exchange argument you use to argue that your greedy strategy yields an optimal solution? You do not have to give a formal proof.

3) Show that the following weaker form of Theorem III on Slide 36 in Lecture 13 holds:

$|\mathbb{C}| \leq \mathbb{C}^* \max\{S | S \in \mathbb{F}\}$

**Problem 3. Bin Packing**

Suppose we are given a set of $n$ items, where the size $s_i$ of the $i^{th}$ item satisfies $0 < s_i < 1$. We want to pack these $n$ items into the minimum number of identical bins, each of which has a capacity of 1 unit. It is assumed that each bin can only contain a subset of items whose total size does not exceed its capacity.

Since the Bin Packing problem is NP-Hard, its decision version can be shown to be NP-complete by reducing from the Subset Sum problem.

Therefore, we have come up with an approximation algorithm called **First-Fit** that works by taking each item in turn and putting it into the first bin that has enough capacity to accommodate it.

(1) Design a polynomial-time algorithm for **First-Fit**.

(2) Analyze the running time.

(3) Show that **First-Fit** is a 2-approximation algorithm.

You might want to follow the following steps in order to prove the claim:

  (3.1) Show that the minimum number of bins is at least $\lceil S \rceil$.

  (3.2) Show that **First-Fit** leaves at most one bin less than half-full.

  (3.3) Show that the number of bins **First-Fit** uses never exceeds $\lceil 2S \rceil$.