CS/ECE 374 Fall 2018 Anqi Yao (anqiyao2@illinois.edu)
Homework 5 Problem 1 Zhe Zhang (zzhan157@illinois.edu)
Ray Ying (xinruiy2@illinois.edu)

Given a graph $G = (V, E)$ a vertex cover of G is a subset $S \subseteq V$ of vertices such that for every edge $(u, v) \in E$, u or v is in S. The goal in the minimum vertex cover problem is to find a vertex cover S of smallest size. In the weighted version of the problem, vertices have non-negative weights $w : V \to \mathbb{Z}_+$, and the goal is to find a vertex cover of minimum weight. Describe a *recursive* algorithm that given a graph $G = (V, E)$ and weights $w(v), v \in V$ outputs a vertex cover of G with minimum weight. Do not worry about the running time.

Solution:

Let's start from an arbitrary vertex v_{start}. The way to consider this problem is two compare the minimum vertex cover weight of two cases:

Case 1: remove v_{start} from S.

Case 2: keep v_{start} and remove the neighbor vertices of v_{start} from S.
Then for each of these sub-problems, take another arbitrary vertex v'_{start} and consider the two cases like above according to v'_{start}.
This way forms a recursion with the base case where no more vertices could be removed to cover all edges.

The pseudo code is as following:

minCover(G, V, E):

if(E is empty) OUTPUT $S = \emptyset$
if(E has only one edge (u, v)) OUTPUT S = the one with smaller weight between u and v

v <- An arbitrary element of V
V' <- V with v removed
E' <- E with edges incident to v removed
$cover'$ <- minCover(G, V', E')
$weight'$ <- sum(weight($cover'$))

V'' <- V with v's neighbor vertices removed
E'' <- E with edges incident to v's neighbor vertices removed
$cover''$ <- minCover(G, V'', E'')
$weight''$ <- sum(weight($cover''$))

if $weight' < weight''$: OUTPUT $S = cover' \cup v$
else: OUTPUT $S = cover'' \cup$ neighbor vertices of v

The running time of this algorithm is approximately $O(2^n)$ because in the worst case we have two case for each sub-problem and the height of the recursion tree could be $O(n)$. Thus the total running time is $O(2^n)$. ■