CS300 Homework #5

TA in charge: Hangyul Park

lawliet2004@kaist.ac.kr

Due: May 21, 2019, at **noon**

1. (30 points)

Consider the following code from the lecture and the textbook, which solves the knapsack problem without repetitions.

Algorithm 1 Knapsack Without Repetitions

```
Initialize all $K(0,j) = 0$ and all $K(w,0) = 0$
for $j = 1$ to $n$ do
    for $w = 1$ to $W$ do
        if $w_j > w$ then
            $K(w,j) = K(w,j-1)$
        else
            $K(w,j) = \max\{K(w,j-1), K(w-w_j, j-1) + v_j\}$
return $K(W,n)$
```

The above algorithm requires $O(nW)$ space for a two-dimensional table. Find an algorithm that solves the *knapsack without repetitions*, and has a space complexity of $O(W)$ and a time complexity of $O(nW)$. (*Hint*: Examine line 3 and line 7 of the code.)

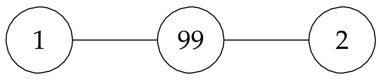
2. (40 points)

A *vertex cover* of a graph $G = (V,E)$ is a subset of vertices $S \subseteq V$ that includes at least one endpoint of every edge in E. In this problem, we define the *cost of a vertex cover* as the sum of the node values in the vertex cover. Give a linear-time algorithm that outputs the minimum cost of a vertex cover in the given tree.

Input: An undirected tree $T = (V,E)$, and the values of the vertices $C_1, C_2, \cdots, C_{|V|}$. Values are an integer and may be negative.
Output: The minimum cost of the optimal set of vertices. The set must be one of the possible vertex covers.

Since we are interested in reducing the cost, the size of the vertex cover does not necessarily have to be minimal. For example, the size of the smallest vertex cover of the following tree is one, but the minimum cost in this problem, is three.

Sample Input



Sample Output

3

3. (30 points)

Suppose that $L_1 <_p L_2$. For each of the following statements, determine whether it is true, false, or an open problem. Justify your answers.

(a) If $L_1, L_2 \in$ **NP** and $L_2 \in$ **P** , then $L_1 \in$ **P**.

(b) If $L_2 \in$ **NP**, then L_2 is either **NP-complete** or is in **P**.

(c) If $SAT <_p L_1$, then L_2 is **NP-complete**

(d) If SAT problem is in **P**, then **co − NP** ≠ **P**.

(e) If a problem in **NP** can be solved in polynomial time, then all problems in **NP** can be solved in polynomial time.

(f) If an **NP-complete** problem can be solved in linear time, then all **NP-complete** problems can be solved in linear time