(2). We will build a $n \times k$ table to solve this problem. let

$M[V,k]$ = the ~~wei~~ maximum weight independent set of the subtree rooted at V with $\le k$ nodes.

The base cases are ~~$M[V, 0]$~~ $M[V_{leave}, 1] = W(V_{leave})$ when V is a leave and we can only pick one. and $M[V, 0] = 0$ for all ~~leaves~~ nodes. that's obvious. Then the general case is:

$$M[V,k] = \max \begin{cases} S(\text{descendent of } V, k) \\ W(V) + S(\text{descendents of descendents of } V, k-1) \end{cases}$$

S is just the solution for problem 1, in general case:

$$S(\{V_1, V_2, V_3 \dots V_n\}, k) = \max \sum_{i=1}^{n} M[V_i, k_i]$$

$$\text{subject to } \sum_{i=1}^{n} k_i \le k$$

evaluation order: for V it will be from leaves to the root. for k it will be from 0 to k.

Complexity: following our evaluation order, when solving $M[V,k]$ all the descendents of V. $M[V_d, k_i]$ for all $k_i < k$ should have already been calculated. then we can access $M[V_d, k_i]$ ~~is~~ in constant ~~time~~ then for S it's just $O(k^2 n)$ as we discussed in problem 1, then for $M[V,k]$ it's $\underbrace{O(k^2 n)}_{\text{calling } S} + O(k^2 n) + O(n)$ for getting maximum, $\Rightarrow O(k^2 n)$

there are $O(nk)$ entries in total, and therefore $O(nk) \cdot O(k^2 n) = O(n^2 k^3)$ time in total and for space, it's $O(nk)$ table.

Correctness: Notice, when proving the correctness of my current algorithm, we assume that S is correct and has running time $O(k^2 n)$. First, we garantee that the nodes we picked will still be a independ set, since for a node V, we either don't take V but consider V's descendents or take V and skip V's descendents to consider the descendents of V's descendents which in both case will generate an independent set. Then we will prove that indeed return the maximum, ~~con~~ we can consider each picked node as a dollar, we have k dollars and since we fill the DP from leaves to roots, 0 to k, we already know the ~~maximum~~ M[Vdescendent, k_i] $k_i < k$, we can treat ~~Vdescendent of~~ every V's descendent as a investment, of f_i and $f_i(k_i)$ is just M[V_i, k_i] and therefore we can spend at most k when not taking V itself or at most k-1 when taking V, therefore, since S is assumed correct, ~~we~~ our algorithm takes the max of the two will also return the maximum weight. So, we return the maximum weight independent set.