Find the weight of the minimum weight dominating set of size at least k.

First, we define an algorithm to find the mininum weight dominating set of size exactly k.

Define relations:

$$\mathrm{OPT}_0(r,k) = \min \begin{cases} A + w(r) \\ \min_{v \in N(r)} B \end{cases}$$

$$\mathrm{OPT}_1(r,k) = C + w(r)$$

$$\mathrm{OPT}_2(r,k) = \min \begin{cases} D + w(r) \\ E \end{cases}$$

$$n = |N(r)|$$

$$A = \min_{k_1,\ldots k_n} \left[\sum_{x=1}^{n} \mathrm{OPT}_2(N(r)_x, k_x) \right], \sum_{i=1}^{n} k_i = k-1$$

$$B = \min_{k_1,\ldots k_n} \left[\mathrm{OPT}_1(v,k_1) + \sum_{x=2}^{n} \mathrm{OPT}_0((N(r) \setminus \{v\})_x, k_x) \right], \sum_{i=1}^{n} k_i = k$$

$$C = \min_{k_1,\ldots,k_n} \left[\sum_{x=1}^{n} \mathrm{OPT}_2(N(r)_x, k_x) \right], \sum_{i=1}^{n} k_i = k-1$$

$$D = \min_{k_1,\ldots,k_n} \left[\sum_{x=1}^{n} \mathrm{OPT}_2(N(r)_x, k_x) \right], \sum_{i=1}^{n} k_i = k-1$$

$$E = \min_{k_1,\ldots,k_n} \left[\sum_{x=1}^{n} \mathrm{OPT}_0(N(r)_x, k_x) \right], \sum_{i=1}^{n} k_i = k$$

Note that the values of these expressions A to E can be computed by the algorithm from problem 1.

To implement this as a dynamic programming solution, we have to examine the dependency graph. When looking at the vertex parameter, since any vertex only depends on OPT values of its children, we can compute the sets of values for the vertices in post-order, which ensures that any vertex will have the OPT values of its children available when it is being computed.

For the k parameter, since for any $\mathrm{OPT}_a(v,b)$ is never dependent on $\mathrm{OPT}_c(v,d)$ for any (a,b,c,d), but only on values less than v, the entries of a row corresponding to any vertex can be computed in any order.

Time complexity analysis:

First, we compute a post-order traversal order for the tree, which takes $O(n)$ time. Next, we fill out 3 n by k tables. For each set of 3 corresponding entries in the three OPT tables, we perform the algorithm from problem 1 on the input size $n_{p1} = |N(v)|, k_{p1} = O(k)$. For A, C, D, and E, this is done once per vertex. For B, this is done $|N(v)|$ times per vertex v. We repeat the entire algorithm

for k values of k up to n to find the minimum weight dominating set of size at least k. This yields a total runtime of $O(n^3k^3(n-k))$.