We root the tree at some node r and associate a subtree T_v with each node $v \in T$, and let n_v denote the number of nodes in the subtree T_v. A solutions for the graph partitioning problem may split the subtree T_v in any ratio, however, it is not hard to see that of (A, B) is the maximum weight cut splitting the tree T into two equal sides, and assume $v \in A$ and $|A \cap T_v| = k$, then the induced partition of T_v by having $A_v = A \cap T_v$, and $B_v = A \cap T_v$ is the maximum weight cut separating T_v into two sides, where the side containing v has size k. This is true as the only interaction of the partition of T_v with the remaining graph is through node v. More precisely, assume it is false, and consider the optimal such cut (A', B'). Replacing A_v by A' keeps all edges across the cut, expect replacing edges that cross the (A_v, B_v) cut with edges that cross the (A', B') cut, and hence it results in a cut of larger weight. This contradiction proves the claim.

Given the above observation, we define subproblems for each subtree T_v and each integer $k \leq n_v$, where $OPT(v, k)$ is the maximum cut of the subgraph T_v separating T_v into two sides where the side containing v has size k. The final answer is $OPT(r, n/2)$ if $n = n_r$ is the number of nodes in T.

We will use $c(u, v)$ as the cost or weight of the edge $e = (v, w)$ of the tree. For a leaf v we have $k = 1$ as the only possible value and $OPT(v, 1) = 0$. Now consider $OPT(v, k)$ for some non-leaf v. The side A containing v must contain $k - 1$ nodes in addition to node v. If v has only one child u then we need to consider two cases depending on whether or not the child u is on the same side of the cut as v, so we get that in this case

$$OPT(v, k) = \max(opt(u, k - 1), c(v, u) + OPT(u, n_u - k + 1)).$$

If v has two children u and w, then we will consider all possible ways of dividing these $k - 1 = \ell_1 + \ell_2$ nodes between the two subtrees rooted at the two children of v. For each such division, there are further case depending whether or not the root of these subtrees is on the same side of the cut as v. We get the following recurrence in this case

$$\begin{aligned} OPT(v, k) \quad = \quad & \max_{0 \leq \ell_1 \leq k-1, \ell_2 = k-1-\ell_1} (\max(OPT(u, \ell_1), c(v, u) + OPT(u, n_u - \ell_1)) \\ & + \max(OPT(w, \ell_2), c(v, w) + OPT(w, n_w - \ell_2)). \end{aligned}$$

There are $O(n^2)$ subproblems. We can compute the values of the subproblems starting at the leaves, and gradually considering bigger subtrees. This recurrence allows us to compute the value of a subproblem in $O(1)$ time given the values of the subproblems associated with smaller trees. So the total time required is $O(n^2)$.

[1] ex34.58.909