AN ALGORITHMIC APPROACH TO NETWORK LOCATION PROBLEMS. II: THE p-MEDIANS*

O. KARIV† AND S. L. HAKIMI‡

Abstract. It is shown that the problem of finding a p-median of a network is an NP-hard problem even when the network has a simple structure (e.g., planar graph of maximum vertex degree 3). However, results leading to efficient algorithms are presented when the network is a tree: In particular, we first show that a 1-median of a tree is identical to its w-centroid, and obtain Goldman’s O(n) algorithm for finding a 1-median of a tree out of more general considerations. Then, we present an algorithm which finds a p-median of a tree (for p > 1) in time O(n^2 \cdot p^2).

1. Introduction. Our basic terminology will be the same as in part I of this paper [1].

A network is a connected undirected graph G(V, E) with a nonnegative number w(v) (called the weight of v) associated with each of its |V| = n vertices, and a positive number l(e) (called the length of e) associated with each of its |E| edges. Let X_p = \{x_1, x_2, \ldots, x_p\} be a set of p points on G, where by a point on G we mean a point along any edge of G which may or may not be a vertex of G. We define the distance d(v, X_p) between a vertex of G and a set X_p on G by

\[ d(v, X_p) = \min_{1 \leq i \leq p} \{d(v, x_i)\} \]

where d(v, x_i) is the length of a shortest path in G between vertex v and point x_i. For each set X_p = \{x_1, x_2, \ldots, x_p\} of p points on G, we define:

\[ H(X_p) = \sum_{v \in V} w(v) \cdot d(v, X_p). \]

We call H(X_p) the distance-sum of the set X_p. If X_p^* on G is such that

\[ H(X_p^*) = \min_{X_p \text{ on } G} \{H(X_p)\}, \]

then X_p^* is called a p-median of G [2],[3]. Hakimi [3] has shown that there exists a set of p vertices V_p^* \subset V, such that H(V_p^*) = H(X_p^*). Thus, there exists a p-median whose points are all vertices. Therefore, in this paper, by a p-median we mean a set V_p^* of p vertices whose distance-sum is minimum:

\[ H(V_p^*) = \min_{V_p \text{ a set of p vertices}} \{H(V_p)\}. \]

We shall assume that p < n, since if p = n then V_p^* = V, H(V_p^*) = 0, while p > n has no mathematical significance. We further assume, as usual and without the loss of generality, that graph G contains neither loops nor multiple edges. Finally, we assume that for each edge e = (v_r, v_s) the length of e is equal to the distance between v_r and v_s.

* Received by the editors January 5, 1977 and in revised form October 9, 1978. This work was supported by Air Force Office of Scientific Research, Air Force Systems Command under Grant AFOSR-76-3017.
† Department of Computer Science, Technion—Israel Institute of Technology, Haifa, Israel.
(i.e., \(l(e) = d(v_r, v_s)\)); because otherwise, the edge \(e\) could be eliminated without affecting a \(p\)-median of \(G\).

Since the publication of [3], there have been many attempts to efficiently compute a \(p\)-median of a network [5], [6], [7] (see also [12], [13], [14], [15]). In this paper we first show that finding a \(p\)-median of a network is at least as hard as \(NP\)-complete problems even in the case where the network has a simple structure (e.g., planar graph of maximum vertex degree 3) (§ 2). However, if the underlying graph of the network is a tree, then there are known algorithms which solve the problem in polynomial-time: Goldman [8] gave an \(O(n)\) algorithm for finding a 1-median of a tree while Matula and Kolde [9] gave an \(O(n^3 \cdot p^2)\) algorithm for finding a \(p\)-median of a tree where \(p > 1\). In this paper we show that a 1-median of a tree is identical to its \(w\)-centroid, and thus derive Goldman’s algorithm out of more general considerations (§ 3). Finally, we give an \(O(n^2 \cdot p^2)\) algorithm for finding a \(p\)-median of a tree where \(p > 1\) (§ 4). An example which demonstrates this algorithm is given in the Appendix.

For generalization of the concept of the \(p\)-median, one is referred to Hakimi and Maheshwari [4], Wendell and Hunter [16], Mirchandani [17] and Minieka [18]. Handler [19] and Halpern [20] defined some combinations of centers and medians. When the network is a tree, both the finding of the “medi-center” as defined by Handler, and the “cent-median” as defined by Halpern involve the finding of a 1-center and a 1-median of the tree, for which purpose our algorithms can be used.

**2. The general \(p\)-median problem is \(NP\)-hard.** In this section we show that the general problem of finding a \(p\)-median \((p > 1)\) of a network is \(NP\)-hard, namely, that there exists an \(O(f(n, p))\) algorithm for finding a \(p\)-median of a general network where \(f(n, p)\) is a polynomial function in each of the variables \(n\) and \(p\) only if \(P = NP\) [10], [23]. In fact, we prove a stronger result: The problem of finding a \(p\)-median is \(NP\)-hard even in the case where the network is a planar graph of maximum vertex degree 3, all of whose edges are of length 1 and all its vertices have weight 1 (namely, the problem of finding a \(p\)-median is \(NP\)-hard even when the network has a simple structure).

We start by quoting a result of Garey and Johnson about the \(NP\)-completeness of the dominating set problem: The dominating set problem is defined as follows: Given a graph \(G(V, E)\) and a positive integer \(p\), \((1 < p < n)\), does there exist a subset \(V^*_p\) of \(p\) or less vertices such that each vertex of \(G\) is either in \(V^*_p\) or is adjacent to a vertex of \(V^*_p\). Garey and Johnson have proved the following result [22]:

**Lemma 2.1.** Let \(G(V, E)\) be a planar graph of maximum vertex degree 3 and let \(p\) be an integer \(1 < p < n\). The problem of finding if there exists in \(G\) a dominating set of cardinality \(p\) is \(NP\)-complete.

**Proof.** The proof of Lemma 2.1 is based on a previous result of Garey and Johnson about the \(NP\)-completeness of the vertex cover problem which is defined as follows: Given a graph \(G(V, E)\) and an integer \(k\), find a subset \(V^*_k\) of the vertices of \(G\) such that \(|V^*_k| \leq k\) and each edge of \(G\) is incident with a vertex of \(V^*_k\). While the general vertex cover problem is known to be \(NP\)-complete [10], Garey and Johnson have shown that the problem is \(NP\)-complete even when \(G\) is a planar graph of maximum vertex degree 3 [21]. Lemma 2.1 is proved by performing the following reduction from the vertex cover problem to the dominating set problem: Replace each edge \((x, y)\) of \(G\) by

\[
\begin{align*}
\text{and set } p &= |E| + k.
\end{align*}
\]

**Theorem 2.1.** The problem of finding a \(p\)-median is \(NP\)-hard even in the case when the network is a planar graph of maximum vertex degree 3 all whose edges are of length 1 and all whose vertices have weight 1.
Proof. Let $G(V, E)$ be a planar graph of maximum vertex degree 3, all of whose edges are of length 1 and all of whose vertices have weight 1. We need only to show that the problem of whether there exists a dominating set of cardinality $p$ in $G$ is polynomial time reducible to the problem of finding a $p$-median of $G$. For, let $V_p$ be an arbitrary subset of $p$ vertices of $G$. Then, by the special structure of $G$, we have: $H(V_p) = \sum_{v \in V} d(v, V_p) \geq n - p$. Thus, if there exists any subset $V^*_p$ for which $H(V^*_p) = n-p$ holds, then $V^*_p$ is a $p$-median of $G$. On the other hand, the equation $H(V^*_p) = n-p$ is satisfied if and only if $d(v, V^*_p) = 1$ for each of the $n-p$ vertices not in $V^*_p$, namely, if and only if $V^*_p$ is a dominating set of cardinality $p$ in $G$. Therefore, there exists a dominating set of cardinality $p$ in $G$ if and only if the distance-sum $H(V^*_p)$ of a $p$-median $V^*_p$ of $G$ is $n-p$. This shows that the problem of finding a dominating set in $G$ is polynomial time reducible to the problem of finding a $p$-median of $G$, and thus the latter problem is NP-hard. Q.E.D.

Remark. Consider the following decision problem which is derived from the (optimization) $p$-median problem: “Given a graph $G(V, E)$, an integer $p$ ($1 < p < n$), and a real positive value $h$. Is there a subset $V^*_p$ of $p$ vertices of $V$, such that $H(V^*_p) = \sum_{v \in V} w(v) \cdot d(v, V^*_p) \leq h$?” Clearly, this problem belongs to the set NP [10], and the proof of Theorem 2.1 implies that it is an NP-hard problem (set $h = n-p$). Therefore, this decision problem is NP-complete.

One observes that if the vertex weights and the edge lengths of $G$ are integers or rationals expressed in binary notation to the same degree of accuracy (which is the usual assumption), then the values of the function $H(V_p)$ (for all possible subsets $V_p$) are bounded by an exponential function of the input length. In this case, one can find a $p$-median $V_p^*$ of $G$ and the optimal value of $H(V_p^*)$ by applying a binary search on the values of $H(V_p)$, using a polynomial number of calls to the NP-complete decision problem which is defined above. This indicates that (under the above assumption), there exists a polynomial-time algorithm for the $p$-median problem if and only if $P = NP$, namely, that the $p$-median problem is in fact “NP-equivalent” rather than “NP-hard”.

3. The 1-median and the centroid of a tree. If the underlying graph of a network $G$ is a tree $T = T(V, E)$, one can find a 1-median of $G$ in $O(n^2)$ steps by performing an exhaustive search on the vertices of the network. Goldman [8] gave an $O(n)$ algorithm for finding a 1-median of a tree. In this section we prove that a 1-median of a tree is identical to its “$w$-centroid”, and we show how this result relates to Goldman’s algorithm.

Let $v$ be a vertex of the tree $T = T(V, E)$ whose degree is $d(v)$. Let $T - v$ be the (unconnected) graph which is obtained by removing $v$ from $T$, and let $T_{v,1}, T_{v,2}, \cdots , T_{v,d(v)}$ be the connected subtrees (the components) of $T - v$. We define:

$$w(T_{v,i}) = \sum_{v' \in T_{v,i}} w(v'), \quad i = 1, 2, \cdots , d(v),$$

$$M(v) = \max_{1 \leq i \leq d(v)} \{w(T_{v,i})\}.$$  \hspace{1cm} (3.2)

A vertex $v_0 \in V$ is called a $w$-centroid of $T$ if and only if

$$M(v_0) = \min_{v \in V} \{M(v)\}. \hspace{1cm} (3.3)$$

\[\text{The term "NP-equivalent" was suggested to the authors by David S. Johnson [22]. The authors wish to thank D. Johnson for his illuminative remarks which helped to clarify the terminology used in this section.}\]
We observe that the lengths of the edges do not play any role in the definition of the \(w\)-centroid. Furthermore, if all vertices of \(T\) have the same weight, then the \(w\)-centroid of \(T\) is also a centroid of \(T\) [11].

The following lemma provides an essential property of a \(w\)-centroid of a tree and can serve as an alternative definition of the \(w\)-centroid:

**Lemma 3.1.** A vertex \(v_0\) of a tree \(T\) is a \(w\)-centroid of \(T\) if and only if

\[
M(v_0) \leq \frac{1}{2} \sum_{v' \in V} w(v').
\]

**Proof.** We first show that if \(v_0\) is a \(w\)-centroid then condition (3.4) holds. Suppose that \(v_0\) is a \(w\)-centroid of \(T\) which does not satisfy (3.4). Let \(j_0\) be such that

\[
w(T_{v_0,j_0}) = M(v_0);
\]

then

\[
w(T_{v_0,j_0}) > \frac{1}{2} \sum_{v' \in V} w(v').
\]

Let vertex \(v_1 \in T_{v_0,j_0}\) be the vertex adjacent to \(v_0\), and let \(T_{v_1,k}\) be the component of \(T - v_1\) containing \(v_0\). Then

\[
w(T_{v_1,k}) = \sum_{v' \in V} w(v') - w(T_{v_0,j_0})
\]

which, by (3.5), implies:

\[
w(T_{v_1,k}) < w(T_{v_0,j_0}) = M(v_0).
\]

Since for each \(l \neq k\), the subtree \(T_{v_1,l}\) is a proper subtree of \(T_{v_0,j_0}\), then \(w(T_{v_1,l}) \leq w(T_{v_0,j_0}) = M(v_0)\), and therefore:

\[
M(v_1) \leq M(v_0).
\]

If \(M(v_1) < M(v_0)\) then a contradiction to the fact that \(v_0\) is a \(w\)-centroid is reached, and this proves that condition (3.4) is satisfied. If \(M(v_1) = M(v_0)\) (which is possible only if \(w(v_1) = 0\)), then \(v_1\) is also a \(w\)-centroid which does not satisfy condition (3.4), and we can repeat the arguments for the vertex \(v_1\) and obtain another vertex \(v_2\) such that

\[
M(v_2) \leq M(v_1) = M(v_0) \text{ and } v_2 \in T_{v_1,j_1}\]

where (because of (3.7)) \(j_1 \neq k\) namely \(T_{v_1,j_1}\) is a proper subtree of \(T_{v_0,j_0}\). Clearly, this process may be repeated less than \(n\) times, leading eventually to a contradiction, which proves the validity of (3.4).

We now show that if a vertex \(v_0\) satisfies condition (3.4), then \(v_0\) is a \(w\)-centroid of \(T\). Let \(v_1 \neq v_0\) be a \(w\)-centroid of \(T\) and let \(T_{v_0,j_0}\) be the component of \(T - v_0\) which contains \(v_1\), and \(T_{v_1,l}\) be the component of \(T - v_1\) which contains \(v_0\). Then, condition (3.4) implies:

\[
M(v_1) \geq w(T_{v_1,l}) \geq \sum_{v' \in V} w(v') - w(T_{v_0,j_0}) = \frac{1}{2} \sum_{v' \in V} w(v') \geq M(v_0).
\]

However, since \(v_1\) is a \(w\)-centroid of \(T\), (3.9) can hold only if \(M(v_1) = M(v_0)\) which in return implies that \(v_0\) is also a \(w\)-centroid of \(T\). Q.E.D.

The following theorem establishes the identity of a 1-median and a \(w\)-centroid of a tree.²

**Theorem 3.1.** A vertex of a tree is a \(w\)-centroid if and only if it is a 1-median of the tree.

²Theorem 3.1 was discovered by the second author and others [9] about four years ago.
Proof. First we show that a 1-median of a tree \( T \) is also a \( w \)-centroid. Let \( v^* \) be a 1-median of \( T \) and suppose it is not a \( w \)-centroid. Let \( T_{v^*,t} \) be the component of \( T - v^* \) such that \( M(v^*) = w(T_{v^*,t}) \), and let \( v_1 \) be the vertex of \( T_{v^*,t} \) which is adjacent to \( v^* \) in \( T \). Then, by Lemma 3.1, we have:

\[
(3.10) \quad w(T_{v^*,t}) > \frac{1}{2} \sum_{v' \in V} w(v').
\]

However, by (1.1) we have:

\[
(3.11) \quad H(v_1) = H(v^*) + \left[ \sum_{v' \in V} w(v') - w(T_{v^*,t}) \right] \cdot d(v^*, v_1) - w(T_{v^*,t}) \cdot d(v^*, v_1).
\]

Equations (3.10) and (3.11) imply \( H(v_1) < H(v^*) \), which contradicts the fact that \( v^* \) is a 1-median. This proves that every 1-median of a tree is also a \( w \)-centroid.

We now show that every \( w \)-centroid of \( T \) is also a 1-median. Let \( v_0 \) be a \( w \)-centroid of \( T \) and let \( v^* \neq v_0 \) be a 1-median of \( T \) (by the first part of the proof \( v^* \) is also a \( w \)-centroid). Let \( T_{v_0,0} \) be the component of \( T - v_0 \) which contains \( v^* \), and let \( T_{v^*,t} \) be the component of \( T - v^* \) which contains \( v_0 \). By Lemma 3.1 we have that \( w(T_{v_0,0}) \leq (1/2) \sum_{v' \in V} w(v') \). However, if \( w(T_{v_0,0}) < (1/2) \sum_{v' \in V} w(v') \) then \( w(T_{v^*,t}) > (1/2) \sum_{v' \in V} w(v') \), which contradicts Lemma 3.1 as \( v^* \) is also a \( w \)-centroid. Therefore we obtain:

\[
(3.12) \quad w(T_{v_0,0}) = \frac{1}{2} \sum_{v' \in V} w(v').
\]

Since \( v^* \) is also a \( w \)-centroid, by similar reasoning we have:

\[
(3.13) \quad w(T_{v^*,t}) = \frac{1}{2} \sum_{v' \in V} w(v').
\]

Equations (3.12) and (3.13) imply:

\[
(3.14) \quad v' \in T_{v_0,0} \cap T_{v^*,t} \Rightarrow w(v') = 0
\]

and therefore

\[
(3.15) \quad H(v_0) = H(v^*) + \sum_{v' \in T_{v_0,0}} w(v') \cdot d(v_0, v^*) - \sum_{v' \notin T_{v_0,0}} w(v') \cdot d(v_0, v^*). 
\]

Finally, (3.12) and (3.15) lead to the conclusion that \( H(v_0) = H(v^*) \), namely that \( v_0 \) is also a 1-median of \( T \). Q.E.D.

Lemma 3.1 and Theorem 3.1 show that a vertex \( v_0 \) is a 1-median of a tree if and only if \( M(v_0) \leq (1/2) \sum_{v' \in V} w(v') \). This property is implemented in a very simple way in Goldman's algorithm for finding a 1-median of a tree [8]:

**Algorithm 3.1**: 1-median of a tree \( T \).

1. [Initialization] Assign: \( T' \leftarrow T \) [\( T' \) is an auxiliary tree], \( W_0 \leftarrow \sum_{v' \in V} w(v') \). For each \( v \in T' \) assign: \( W(v) \leftarrow w(v) \).
2. If the auxiliary tree \( T' \) consists of a single vertex \( v_0 \) then Halt [the vertex \( v_0 \) is the 1-median of \( T \)].
3. Let \( v \) be a leaf of the auxiliary tree \( T' \). If \( W(v) \geq W_0/2 \) then Halt [\( v \) is a 1-median of \( T \)]. Else, let \( u \) be the vertex adjacent to \( v \) in the auxiliary tree \( T' \). Assign: \( W(u) \leftarrow W(u) + W(v) \), remove vertex \( v \) (and edge \( (u, v) \)) from the auxiliary tree: \( T' \leftarrow T' - \{v\} \) and go to 2.
4. On finding a p-median \((p > 1)\) of a tree. In this section we describe an \(O(n^2 \cdot p^3)\) algorithm for finding a p-median \((p > 1)\) of a tree \(T\). (Notice that the exhaustive method requires \(O(n^p)\) steps for finding a p-median while the algorithm of Matula and Kolde\[9\] requires \(O(n^3 \cdot p^3)\) steps). An example which demonstrates the different stages of the algorithm and the notations which are used, is presented in the Appendix.

We begin by converting the given tree into a rooted tree as follows: We pick an arbitrary vertex \(v_0 \in V\) to be the “root” of the tree \(T\). Let \(v\) be a vertex of the tree. We define the level \(\text{Lev}(v)\) of \(v\) to be the number of edges on the path \(p(v_0, v)\) which leads from the root \(v_0\) to \(v\) [in particular, \(\text{Lev}(v_0) = 0\)]. We denote: \(L_v = \max_{v \in V} \{\text{Lev}(v)\}\). If \(v \neq v_0\), then by removing the last edge of the path \(p(v_0, v)\) we obtain two connected subtrees. We denote that subtree which contains \(v\) by \(T_v\), and we define \(u\) as the root of \(T_v\) [in particular, if \(u\) is a leaf of \(T\), then \(T_v\) is the single vertex \(v\)]. The number of vertices of \(T_v\) is denoted by \(|T_v|\) [in particular, \(|T_0| = n\)].

Let \(v \in V\), and let \(E(v)\) be the set of all edges of \(T_v\) which are adjacent to \(v\) [in particular, if \(v\) is a leaf of \(T\) then \(E(v) = \emptyset\)]. We define an arbitrary order among the edges of \(E(v)\), and we denote the \(l\)th edge (according to this order) by \(e(v, l)\). If vertex \(v_s\) is the other endpoint of the edge \(e(v, l)\) (namely, \(e(v, l) = (v, v_s)\)), then we say that \(v_s\) is the \(l\)th son of \(v\), and \(v\) is the father \(F(v_s)\) of \(v_s\).

As a pre-procedure for the algorithm we compute the distance-matrix of the tree (this requires \(O(n^2)\) steps). The algorithm itself is of dynamic-programming type, and it consists of two phases: During the first phase we traverse the edges of the tree “upward”, from the vertices of higher levels towards the vertices of lower levels, and we compute certain values to be associated with the edges and the vertices of the tree. These values are in fact the corresponding distance-sums of \(k\)-medians \((1 \leq k \leq p)\) as calculated over the different subtrees \(T_v\) and over other subtrees of the original tree \(T\). In particular, we find the distance sum \(H(v_0, p)\) corresponding to a \(p\)-median of the whole tree. We use these values throughout the second phase in order to traverse the tree “downward,” from lower levels to higher levels, and to locate the points of a \(p\)-median at \(p\) selected vertices of \(T\).

4.1. Notations. In addition to the above concepts, we shall use the following notations (where \(T'\) denotes any connected subtree of \(T\), and \(k\) is any integer in the range \(1 \leq k \leq \min \{|T'|, p\}\):

1. \(V^*(T, k)\). This is a \(k\)-median of \(T'\). In particular: \(V^*(T_v, k)\) is a \(k\)-median of \(T_v\) and if there are more than one \(k\)-median of \(T_v\) then \(V^*(T_v, k)\) is a \(k\)-median of \(T_v\) whose distance from \(v\) is minimal, i.e., \(d(v, V^*(T_v, k)) = \min \{d(v, V^*')|V^*\prime\text{is a }k-\text{median of }T_v\}\).

2. \(H(T', k)\). This is the distance-sum corresponding to \(V^*(T', k)\); i.e., \(H(T', k) = \sum_{v' \in T'} w(v') \cdot d(v', V^*(T', k))\). [In particular, \(H(T_0, p) = H(T, p)\) is the distance-sum corresponding to a \(p\)-median of the original tree \(T\)].

3. A vertex \(v' \in T'\) is covered by a vertex \(v^* \in V^*(T', k)\) if \(v^*\) is a vertex of \(V^*(T', k)\) which is closest to \(v'\); i.e., \(d(v', v^*) = d(v', V^*(T', k))\).

4. \(c(v, k)\). This is a vertex of \(V^*(T_v, k)\) which covers \(v\); i.e., \(d(v, c(v, k)) = d(v, V^*(T_v, k))\).

5. \(T_{e(v, l)}\). This is the maximal connected subtree of \(T_v\) which contains \(v\) but does not contain any edge \(e(v, j)\) for \(j > l\). [In particular, \(T_{e(v, 0)}\) is the single vertex \(v\); also, if \(v\) has \(r\) sons, then \(T_{e(v, r)} = T_v\)]. The number of vertices of \(T_{e(v, l)}\) is \(|T_{e(v, l)}|\).

We now define the essential value \(R(k, e(v, l), v)\). Before we give the formal definition of this value, let us first explain its interpretation:
Let $e(v, l)$ be an arbitrary edge of the tree $T$. Assume that $V_p^*$ is a $p$-median of the tree $T$, such that the vertices of the subtree $T_{e(v, l)}$ are “covered” by $k$ vertices $V_k^*$ of $V_p^*$, where, in particular, the vertex $v$ (the root of $T_{e(v, l)}$) is covered by some vertex $v_r$ of $V_k^*$. Then, we define the value $R(k, e(v, l), v_r)$ to be the appropriate partial distance-sum of the $p$-median over the subtree $T_{e(v, l)}$:

$$R(k, e(v, l), v_r) = \sum_{v' \in T_{e(v, l)}} w(v') \cdot d(v', V_p^*).$$

In fact, given $v_r$ (the vertex of $V_p^*$ which covers $v$), and given $k$ (the number of points of $V_p^*$ which cover $T_{e(v, l)}$), one can compute $R(k, e(v, l), v_r)$ even when the set $V_p^*$ itself is not available. To do this we proceed as follows: Let $V_1$ denote the set of vertices of $T_{e(v, l)}$, which are to be covered by $v_r$. Then:

$$R(k, e(v, l), v_r) = \sum_{v' \in V_1} w(v') \cdot d(v', v_r) + \sum_{v' \in T_{e(v, l)} - V_1} w(v') \cdot d(v', V_k^* - \{v_r\}).$$

We note that the second term on the right-hand side of the equation is in fact the distance-sum corresponding to the $(k - 1)$-median of the graph $T_{e(v, l)} - V_1$; namely: $H(T_{e(v, l)} - V_1, k - 1)$. Thus, $R(k, e(v, l), v_r) = \sum_{v' \in V_1} w(v') \cdot d(v', v_r) + H(T_{e(v, l)} - V_1, k - 1)$. Notice that now the value $R(k, e(v, l), v_r)$ does not depend on the set $V_p^*$ but on the set of vertices $V_1$ (which is a subset of the vertices of $T_{e(v, l)}$). One can observe that the vertices of $V_1$ constitute a connected subtree $T_1(V_1, E_1)$ of $T_{e(v, l)}$ whose root is $v$. Moreover, if $p(v, v)$ is the path in $T$ which connects $v$ to $v$, and $V'$ is the set of vertices of $p(v, v)$ which belong to $T_{e(v, l)}$, then $V'$ is a subset of $V_1$ (in fact, if $v_r \in T_{e(v, l)}$ then $V'$ contains all vertices of $p(v, v)$; otherwise, $V' = \{v\}$). On the other hand, it is clear that the graph $T_{e(v, l)} - V_1$ consists of at most $k - 1$ components. Clearly, those properties of the set $V_1$ are not sufficient to define $V_1$ uniquely. However, since $R(k, e(v, l), v_r)$ is a partial distance-sum of the entire distance-sum $H(T_{e(v, l)}, p)$ (which corresponds to the $p$-median of the whole tree $T$), $V_1$ must be a subset which minimizes the value $R(k, e(v, l), v_r)$. Thus

$$R(k, e(v, l), v_r) = \min_{V_1} \left\{ \sum_{v' \in V_1} w(v') \cdot d(v', v_r) + H(T_{e(v, l)} - V_1, k - 1) \right\}.$$

Since a priori, the vertex $v_r$ and the integer $k$ are not known, we use the latter equation in order to define the value $R(k, e(v, l), v_r)$ for any arbitrary $v_r$ and for all possible values of $k$ as follows:

**Definition.** Let $e(v, l)$ be any arbitrary edge of $T$ and let $v_r$ be any vertex of $T$. Let $p(v, v)$ be the path which connects $v$ to $v$ in $T$, and let $V'$ be the set of vertices of $p(v, v)$ which belong to $T_{e(v, l)}$ (if $v_r \in T_{e(v, l)}$ then $V'$ is the set of all vertices on $p(v, v)$; else $V' = \{v\}$). For each $k$, $1 \leq k \leq \min\{p, |T_{e(v, l)}| - |V'| + 1\}$, let $V_1$ be a subset of the vertices of $T_{e(v, l)}$ such that:

(a) the subgraph induced by the vertices of $V_1$ constitutes a connected subtree $T_1(V_1, E_1)$ of $T_{e(v, l)}$ whose root is $v$ and $V' \subseteq V_1$; and

(b) the graph $T_{e(v, l)} - V_1$ consists of at most $k - 1$ components. Then:

$$R(k, e(v, l), v_r) = \min_{V_1} \left\{ \sum_{v' \in V_1} w(v') \cdot d(v', v_r) + H(T_{e(v, l)} - V_1, k - 1) \right\}.$$

In particular we have:

$$R(1, e(v, l), v_r) = \sum_{v' \in T_{e(v, l)}} w(v') \cdot d(v', v_r).$$
and
\[ R(k, e(v, 0), v_r) = \begin{cases} \omega(v) \cdot d(v, v_r), & \text{if } k = 1 \\ \text{undefined}, & \text{otherwise.} \end{cases} \]

4.2. The first phase. The first phase of the algorithm is carried out in stages. The goal of this phase is to compute all the relevant partial sums \( R(k, e(v, l), v_r) \) and \( H(T_v, k) \) (and in particular, to obtain the distance-sum \( H(T_v, p) \) of the whole tree). This goal is achieved by employing a dynamic-programming type procedure, as follows:

When the \( i \)th stage begins we already know all the values \( H(T_v, k) \) and \( c(v, k) \) such that \( \text{Lev}(v) \geq L_m - i + 1 \), and we also know all the values \( R(k, e(v, l), v_r) \) such that every vertex on the path \( p(v, v_r) \) (which connects \( v \) and \( v_r \)) has a level \( \geq L_m - i + 1 \). During the \( i \)th stage, we search through all vertices of level \( L_m - i \). For each such a vertex \( v_o \), we first compute all values \( R(k, e(v, j), v_2) \) such that \( v_1, v_2 \in T_o \) and the path \( p(v_1, v_2) \) passes through \( v_o \). Then we compute the values \( H(T_o, k) \) and \( c(v, k) \) and go on to the next vertex of level \( L_m - i \). Thus, when the \( i \)th stage is completed we know all values \( H(T_v, k) \) and \( c(v, k) \) such that \( \text{Lev}(v) \geq L_m - i \), and we know all values \( R(k, e(v, l), v_r) \) such that every vertex of the path \( p(v, v_r) \) has a level \( \geq L_m - i \). In particular after the \( L_m \)th stage, all possible values \( H(T_v, k) \), \( c(v, k) \) and \( R(k, e(v, l), v_r) \) are known, and especially, we obtain the distance-sum \( H(T_v, p) \) of the whole tree.

We now describe the first phase in detail (a formal presentation of the algorithm is given at the end of this section).

The initial values. When the first phase begins, we assign for each leaf \( v \) of the tree \( T \) the following values:
\[ H(T_v, 1) = 0, \quad c(v, 1) = v. \]
We also assign for each vertex \( v \) of \( T \): \( R(1, e(v, 0), v) = 0 \).

The computation of \( R(k, e(v, l), v_r) \) during the \( i \)th stage. Let \( v_s \) be a vertex such that \( \text{Lev}(v_s) = L_m - i \). During the \( i \)th stage we compute all values \( R(k, e(v, j), v_2) \) such that \( v_1, v_2 \in T_o \) and \( p(v_1, v_2) \) passes through \( v_o \), in the following order: First we compute all values \( R(k, e(v_1, j), v_2) \) such that both \( e(v_1, j) \) and \( v_2 \) belong to \( T_e(e(v_1, j)) \). Then we compute all values \( R(k, e(v, j), v_2) \) such that \( e(v_1, j) \) and \( v_2 \) belong to \( T_e(e(v_2, j)) \) but not both of them belong to \( T_e(e(v_2, j)) \). Then we compute all values \( R(k, e(v, j), v_2) \) such that both \( e(v_1, j) \) and \( v_2 \) belong to \( T_e(e(v_2, j)) \) but not both of them belong to \( T_e(e(v_2, j)) \).— and so on, until all values \( R(k, e(v, j), v_2) \) where \( v_1, v_2 \in T_o \) and \( p(v_1, v_2) \) passes through \( v_o \) are known [clearly, if \( v_s \) is a leaf, no computation is required].

Let us describe the computation of the values \( R(k, e(v_1, j), v_2) \) where both \( e(v_1, j) \) and \( v_2 \) belong to \( T_e(e(v_1, j)) \) but not both of them belong to \( T_e(e(v_1, j)) \). Actually, we deal separately with the following four cases:

(i) \( v_1 \in T_e(e(v_1, j)) - T_e(e(v_1, j-1)), \quad v_2 \in T_e(e(v_1, j-1)) \)
(ii) \( e(v_1, j) = e(v_2, l), \quad v_2 \in T_e(e(v_2, l)) \)
(iii) \( e(v_1, j) = e(v_2, l), \quad v_2 \in T_e(e(v_2, l)) - T_e(e(v_2, l-1)) \)
(iv) \( e(v_1, j) \in T_e(e(v_1, j-1)), \quad v_2 \in T_e(e(v_2, l)) - T_e(e(v_2, l-1)) \)

Case (i) \( [\text{Fig. 1}]: v_1 \in T_e(e(v_1, j)) - T_e(e(v_1, j-1)), v_2 \in T_e(e(v_2, l)) \). The computation of the values \( R(k, e(v_1, j), v_2) \) is carried out in the following order: We fix an arbitrary vertex \( v_2 \in T_e(e(v_2, l-1)) \) and allow \( e(v_1, j) \) to run in a certain order over all edges of \( T_e(e(v_1, j)) - T_e(e(v_1, j-1)) \) [except for the edge \( e(v_2, l) \) which is treated in case (ii)]. Then we repeat this process for another choice of \( v_2 \in T_e(e(v_2, l-1)) \) and so on until all the values corresponding to case (i) are computed.
Thus, assume that we fix a vertex $v_2 \in T_{e(v_i, i_1-1)}$. The computation of the values $R(k, e(v_1, j_1), v_2)$ is performed according to the levels of the vertices $v_1$ of $T_{e(v_i, i_1-1)}$, starting from the highest levels and proceeding to the lower ones. Assume that all values $R(k, e(v_1, j_1), v_2)$ for vertices $v_1$ of levels greater than (some) $i_1$ have already been computed. We describe now the computation of the values $R(k, e(v_1, j_1), v_2)$ for some vertex $v_1$ of level $i_1$.

We start by assigning:

$$R(1, e(v_1, 0), v_2) = w(v_1) \cdot d(v_1, v_2).$$

If $v_1$ is a leaf then no more computation is required and we proceed to the next vertex of level $i_1$.

If $v_1$ is not a leaf then we first compute all values $R(k, e(v_1, 1), v_2)$, then we compute all values $R(k, e(v_1, 2), v_2)$, and so on until all values of the form $R(k, e(v_1, j_1), v_2)$ are computed ($1 \leq j_1 \leq t_1$, where $t_1$ is the number of the sons of $v_1$).

Thus, assume that we have already computed all values $R(k, e(v_1, j_1), v_2)$ for $j < j_1$, and we now compute the values $R(k, e(v_1, j_1), v_2)$. Let $v_3$ be the vertex at the other end of $e(v_1, j_1)$ (namely, $e(v_1, j_1) = (v_1, v_3)$). Let $t_3$ be the number of the sons of $v_3$ (in particular, if $v_3$ is a leaf then $t_3 = 0$). Then, for the appropriate values of $k$ (since in Case (i) $p(v_1, v_2) = \{v_1\}$, the range of values of $k$ is $1 \leq k \leq \min \{|T_{e(v_1, j_1)}|, p\}$), we assign the following values:

$$R(k, e(v_1, j_1), v_2) = \min_{k_1, k_2} \{R(k_1, e(v_1, j_1 - 1), v_2) + R(k + 1 - k_1, e(v_3, t_3), v_2),$$

$$R(k_2, e(v_1, j_1 - 1), v_2) + H(T_{v_3}, k - k_2)\}.$$
term exists for those values of $k_1$ which satisfy both conditions: 1 is $\leq k_1 \leq \min \{ k, |T_{e(v_3, l-1)}| \}$ and $1 \leq k + 1 - k_1 \leq |T_{v_3}|$, namely those values of $k_1$ which satisfy:
\[
\max \{ 1, k + 1 - |T_{v_3}| \} \leq k_1 \leq \min \{ k, |T_{e(v_3, l-1)}| \}.
\]
The second term on the right-hand side of (4.2) corresponds to the case where $v_3$ is not "covered" by $v_2$, namely $T_{v_3}$ is a component of $T_{e(v_3, l-1)} - T_1$ (see the definition of $R(k, e(v_3, l_1), v_2)$). This term exists for those values of $k_2$ which satisfy both conditions:
\[
1 \leq k_2 \leq \min \{ k - 1, |T_{e(v_3, l-1)}| \} \text{ and } 1 \leq k - k_2 \leq |T_{v_3}|,
\]
namely those values of $k_2$ which satisfy:
\[
\max \{ 1, k - |T_{v_3}| \} \leq k_2 \leq \min \{ k - 1, |T_{e(v_3, l-1)}| \}.
\]

Case (ii) [Fig. 2]: $e(v_1, l_1) = e(v_3, l)$, $v_2 \in T_{e(v_3, l-1)}$. For each $v_2 \in T_{e(v_3, l-1)}$, the computation of $R(k, e(v_3, l), v_2)$ is carried out as follows: Let $v_3$ be the vertex at the other end of $e(v_3, l)$ (namely, $e(v_3, l) = (v_3, v_2)$), and let $t_3$ be the number of sons of $v_3$ (if $v_3$ is a leaf then $t_3 = 0$). Then, for the appropriate values of $k$ (since in Case (ii) $T_{e(v_3, l)} \cap p(v_3, v_2) = p(v_3, v_2)$ and $|p(v_3, v_2)| = \text{Lev}(v_2) - \text{Lev}(v_3) + 1$, these values of $k$ are $1 \leq k \leq \min \{ p, |T_{e(v_3, l)}| + \text{Lev}(v_3) - \text{Lev}(v_2) \}$), we assign the following values:

\[
R(k, e(v_3, l), v_2) = \min_{k_1,k_2} \{ R(k_1, e(v_3, l-1), v_2) + R(k + 1 - k_1, e(v_3, t_3), v_2),
\]
\[
R(k_2, e(v_3, l-1), v_2) + H(T_{v_3}, k - k_2) \}.
\]

[The first term on the right-hand side of (4.3) corresponds to the case where $v_3$ is "covered" by $v_2$, namely $v_3 \in T_1(V_1, E_1)$ (see the definition of $R(k, e(v_3, l), v_2)$). This term exists for those values of $k_1$ which satisfy both conditions: $1 \leq k_1 \leq \min \{ k, |T_{e(v_3, l-1)}| + \text{Lev}(v_3) - \text{Lev}(v_2) \}$ and $1 \leq k + 1 - k_1 \leq |T_{v_3}|$, namely those values of $k_1$ which satisfy: $\max \{ 1, k + 1 - |T_{v_3}| \} \leq k_1 \leq \min \{ k, |T_{e(v_3, l-1)}| + \text{Lev}(v_3) - \text{Lev}(v_2) \}$.

\]

\[
3 \text{ Notice that in Case (ii), } |p(v_3, v_2)| = \text{Lev}(v_2) - \text{Lev}(v_3) + 1.
\]
The second term on the right-hand side of (4.3) corresponds to the case where $v_3$ is not "covered" by $v_2$, namely $T_{v_3}$ is a component of $T_{v_2} - T_1$ (see the definition of $R(k, e(v_2, l), v_2)$). This term exists for those values of $k_2$ which satisfy both conditions (see footnote 3): $1 \leq k_2 \leq \min \{k - 1, |T_{e(v_2,l-1)}| + \text{Lev}(v_2) - \text{Lev}(v_3)\}$ and $1 \leq k - k_2 \leq |T_{v_3}|$, namely those values of $k_2$ which satisfy: $\max \{1, k - |T_{v_3}|\} \leq k_2 \leq \min \{k - 1, |T_{e(v_2,l-1)}| + \text{Lev}(v_2) - \text{Lev}(v_3)\}$.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig3.png}
\caption{Case (iii). $e(v_1, f_1) = e(v_3, l)$, $v_2 \in T_{e(v_2,l-1)}$.}
\end{figure}

Case (iii) [Fig. 3]: $e(v_1, f_1) = e(v_3, l)$, $v_2 \in T_{e(v_2,l-1)}$. For each $v_2 \in T_{e(v_2,l-1)}$, the computation of $R(k, e(v_2, l), v_2)$ is carried out as follows: Let $v_3$ be the vertex at the other end of $e(v_2, l)$ (namely, $e(v_2, l) = (v_3, v_1)$) and let $r_3$ be the number of sons of $v_3$ (if $v_3$ is a leaf then $r_3 = 0$). In this case the path $p(v_2, v_3)$ from $v_2$ to $v_3$ passes through $v_3$ and thus $v_3$ is "covered" by $v_2$, namely $v_3 \in T_1(V_1, E_1)$ (see the definition of $R(k, e(v_2, l), v_2)$). Thus, for the appropriate values of $k_2$ (1 $\leq k_2 \leq \min \{p, T_{e(v_2,l)}| + \text{Lev}(v_3) - \text{Lev}(v_2)\}$) we assign the following values:\footnote{Notice that in Case (iii), $|p(v_2, v_3)| = \text{Lev}(v_2) - \text{Lev}(v_3) + 1$, and $|p(v_2, v_3)| = \text{Lev}(v_2) - \text{Lev}(v_3) + 1 = \text{Lev}(v_2) - \text{Lev}(v_3)$.}

\begin{equation}
R(k, e(v_2, l), v_2) = \min \{R(k_1, e(v_2, l - 1), v_2) + R(k + 1 - k_1, e(v_3, r_3), v_2)\}
\end{equation}

where $k_1$ satisfies the following conditions:\footnote{Notice that in Case (iii), $|p(v_2, v_3)| = \text{Lev}(v_2) - \text{Lev}(v_3) + 1$, and $|p(v_2, v_3)| = \text{Lev}(v_2) - \text{Lev}(v_3) + 1 = \text{Lev}(v_2) - \text{Lev}(v_3)$.}

\begin{equation}
1 \leq k_1 \leq \min \{k, |T_{e(v_2,l-1)}|\}
\end{equation}

and

\begin{equation}
1 \leq k + 1 - k_1 \leq \min \{p, |T_{v_3}| + 1 + \text{Lev}(v_3) - \text{Lev}(v_2)\}
\end{equation}

namely:

\begin{equation}
\max \{1, k + \text{Lev}(v_2) - \text{Lev}(v_3) - |T_{v_3}|\} \leq k_1 \leq \min \{k, |T_{e(v_2,l-1)}|\}.
\end{equation}
Case (iv): \( e(v_1, j_1) \in T_{e(v_0, l-1)}, \ v_2 \in T_{e(v_0, l)} - T_{e(v_0, l-1)} \). The computation of \( R(k, e(v_1, j_1), v_2) \) in this case is carried out in an entirely similar way as in Case (i) and thus the relations (4.1) and (4.2) may be applied in this case.

The computation of \( H(T_{v_0}, k) \) and \( c(v, k) \). By the definition, \( H(T_{v_0}, k) \) is the distance-sum corresponding to a \( k \)-median of \( T_{v_0} \). Let \( t_0 \) be the number of sons of \( v_0 \) (thus \( T_{e(v_0, t_0)} = T_{v_0} \)). Let \( V^{it}(T_{v_0}, k) \) be a \( k \)-median of \( T_{v_0} \) and \( v'_i \) be the point of \( V^{it}(T_{v_0}, k) \) which covers \( v_0 \). Then, by the definition: \( H(T_{v_0}, k) = R(k, e(v_0, t_0), v'_i) \).

Therefore we can compute \( H(T_{v_0}, k) \) by the following relation:

\[
H(T_{v_0}, k) = \min_{v_2 \in T_{v_0}} \{ R(k, e(v_0, t_0), v_2) \}.
\]

Clearly, every vertex \( v'_i \in T_{v_0} \) for which \( R(k, e(v_0, t_0), v'_i) = H(T_{v_0}, k) \) is a member of a \( k \)-median of \( T_{v_0} \). Therefore, the vertex \( c(v, k) \) can be found by the relation:

\[
d(v_0, c(v, k)) = \min_{v'_i \in T_{v_0}} \{ d(v_0, v'_i) | R(k, e(v_0, t_0), v'_i) = H(T_{v_0}, k) \}.
\]

Relations (4.5) and (4.6) hold for \( 1 \leq k \leq \min \{ p, |T_{v_0}| \} \).

We can now formulate the algorithm for the first phase.

Preprocessing. As it was explained at the beginning of § 4, a preprocessing for the algorithm of finding a \( p \)-median of a tree, includes the choosing of a vertex \( v_0 \) as the root of the tree, the assigning of levels to the vertices, the ordering of the edges of each set \( E(v) \), and the computing of the distance matrix of the tree. This preprocessing requires \( O(n^2) \) steps.

As a preprocessing for the first phase we must arrange the vertices of \( T \) according to their levels, we must have the number of sons of each vertex and we must construct a list of the leaves of \( T \). Each of those processings requires \( O(n) \) steps. We also must construct for each edge \( e(v, j) \) of \( T \) the lists of all vertices of the subtrees \( T_{e(v, j)} \) and \( T_{e(v, j)} - T_{e(v, j-1)} \), where these lists are arranged according to the levels of the vertices. Since the construction of each such a list requires \( O(n) \) steps, the total complexity of the preprocessing for the first phase is \( O(n^2) \).

In fact, most of the information which is gained by the preprocessing need not be computed before the performance of the first phase, but may be calculated each time it is required during the first phase. Such a procedure will save a lot of memory-space. However, for the simplicity and convenience of stating the algorithm itself, we assume that all this information is prepared before the algorithm starts.

Algorithm 4.1.

Algorithm 4.1. The First Phase.

1. [Initialization] for each leaf \( v \) of the tree \( T \) assign: \( H(T_v, 1) \leftarrow 0 \), \( c(v, 1) \leftarrow v \); For each vertex \( v \) of \( T \) assign: \( R(1, e(v, 0), v) \leftarrow 0 \). Assign: \( i \leftarrow 0 \). Let \( L_m \) be the maximum level of a vertex of \( T \).

2. [The \( i \)th stage] If \( i = L_m \) then Halt [proceed to the second phase]. Else, assign \( i \leftarrow i + 1 \).

3. [Choose a vertex \( v_i \) of level \( L_m - i \)] If for every vertex \( v_i \) such that \( \text{Lev} (v_i) = L_m - i \) the values \( H(T_{v_i}, k) \) and \( c(v_i, k) \) for \( 1 \leq k \leq \min \{ p, |T_{v_i}| \} \) are already computed, then return to 2. Else, let \( v_j \) be a vertex of level \( L_m - i \) for which the values \( H(T_{v_j}, k) \) and \( c(v_j, k) \) for \( 1 \leq k \leq \min \{ p, |T_{v_j}| \} \) are not yet computed. Let \( t_j \) be the number of sons of \( v_j \). Assign \( i \leftarrow 0 \).

4. If \( i = t_j \) then go to 22. Else, assign: \( i \leftarrow i + 1 \). Let \( L_1 \) be the maximum level of a vertex of \( T_{e(v_j, i)} - T_{e(v_j, i-1)} \).
5. [Case (i)] If all the vertices $v_2$ where $v_2 \in T_{e(v_1,l-1)}$ have already been chosen in this step, then go to 10. Else, let $v_2$ be a vertex of $T_{e(v_1,l-1)}$ which has not yet been chosen. Let $i_1 \leftarrow L_1 + 1$.

6. If $i_1 = \text{Lev}(v_2) + 1$ then return to 5. Else, assign $i_1 \leftarrow i_1 - 1$.

7. [Choose $v_1$] If all the vertices $v_1$ where $v_1 \in T_{e(v_1,l-1)}$ and $\text{Lev}(v_1) = i_1$ have already been chosen in this step, then return to 6. Else, let $v_1$ be a vertex of level $i_1$, such that $v_1 \in T_{e(v_1,l-1)}$ and $v_1$ has not yet been chosen. Perform the assignment (4.1). Let $t_1$ be the number of sons of $v_1$. If $t_1 = 0$ then repeat 7. Else, let $j_1 \leftarrow 0$.

8. If $j_1 = t_1$ then return to 7. Else assign: $j_1 \leftarrow j_1 + 1$.

9. Let $v_3$ be the vertex at the other end of $e(v_1, j_1)$ and let $t_3$ be the number of sons of $v_3$. Perform the assignments (4.2) and return to 8.

10. Let $L_2$ be the maximum level of a vertex of $T_{e(v_1,l-1)}$.

11. [Case (iv)] If all the vertices $v_2$ where $v_2 \in T_{e(v_1,l-1)}$ have already been chosen in this step, then go to 19. Else let $v_2$ be a vertex of $T_{e(v_1,l-1)}$ which has not yet been chosen. Let $i_1 \leftarrow L_2 + 1$.

12. If $i_1 = \text{Lev}(v_2) + 1$ then go to 16. Else, assign $i_1 \leftarrow i_1 - 1$.

13. [Choose $v_1$] If all the vertices $v_1$ where $v_1 \in T_{e(v_1,l-1)}$ and $\text{Lev}(v_1) = i_1$ have already been chosen in this step, then return to 12. Else let $v_1$ be a vertex such that $\text{Lev}(v_1) = i_1$, $v_1 \in T_{e(v_1,l-1)}$ and $v_1$ has not yet been chosen. Perform the assignment (4.1). Let $t_1$ be the number of sons of $v_1$. If $t_1 = 0$ then repeat 13. Else, let $j_1 \leftarrow 0$.

14. If $j_1 = t_1$ then return to 13. Else, assign $j_1 \leftarrow j_1 + 1$.

15. Let $v_3$ be the vertex at the other end of $e(v_1, j_1)$ and let $t_3$ be the number of sons of $v_3$. Perform the assignments (4.2) and return to 14.

16. Let $v_3 \leftarrow v_n$, $j_1 \leftarrow l$ Let $v_3 \leftarrow v_n$. Perform the assignment (4.1). Assign: $j_1 \leftarrow 0$.

17. If $j_1 = l - 1$ then return to 11. Else, assign $j_1 \leftarrow j_1 + 1$.

18. Let $v_3$ be the vertex at the other end of $e(v_1, j_1)$ and let $t_3$ be the number of sons of $v_3$. Perform the assignments (4.2) and return to 17.

19. Let $v_3$ be the vertex at the other end of $e(v_n, l)$ and let $t_3$ be the number of sons of $v_3$.

20. [Case (iii)] If all the vertices $v_2$ where $v_2 \in T_{e(v_1,l-1)}$ have already been chosen in this step, then go to 21. Else, let $v_2$ be a vertex of $T_{e(v_1,l-1)}$ which has not yet been chosen. Perform the assignments (4.3) and repeat 20.

21. [Case (iii)] If all the vertices $v_2$ where $v_2 \in T_{e(v_1,l-1)}$ have already been chosen in this step, then go to 4. Else, let $v_2$ be a vertex of $T_{e(v_1,l-1)}$ which has not yet been chosen. Perform the assignments (4.4) and repeat 21.

22. [Assignment of $H(T_{v_1}, k)$ and $c(v_2, k)$] Perform the assignments (4.5) and (4.6) and return to 3.

The complexity of the first phase. For each edge $e(v_1, j_1)$ and for each vertex $v_2$ there are at most $p$ values of $k$ for which the value $R(k, e(v_1, j_1), v_2)$ must be computed. Thus, there are at most $n^2 \cdot p$ values of the form $R(k, e(v_1, j_1), v_2)$ to be computed. Each of those computations involves the finding of a minimum over at most $2k$ terms (notice that in (4.2), (4.3) and (4.4), $k_1$ and $k_2$ vary at most between 1 and $k$). Therefore, the total complexity of computing the values $R(k, e(v_1, j_1), v_2)$ is $O(n^2 \cdot p^2)$.

For each vertex $v_n$, there are at most $p$ values of $k$ for which the values $H(T_{v_n}, k)$ and $c(v_n, k)$ are computed. Each of these computations requires at most $O(n)$ steps (see (4.5) and 4.6)), and thus the complexity of computing all values $H(T_{v_n}, k)$ and $c(v_n, k)$ is $O(n^2 \cdot p)$. The total complexity of the first phase is therefore $O(n^2 \cdot p^2)$. 
4.3. The second phase. When we begin the second phase, we already know all the possible values $H(T_v, k)$, $c(v, k)$ and $R(k, e(v, l), v_l)$, and in particular we know the distance-sum $H(T_v, p)$ which corresponds to a $p$-median of $T$. During the second phase we intend to actually construct a $p$-median of $T$ and to find which vertices of $T$ are to be covered by each point of the $p$-median. This is done by applying a depth first search on the edges of $T$ in a direction from its root toward its leaves. The search is carried out in an order opposite to the order by which the edges were searched during the first phase; namely, when a vertex $v_l$ is reached in the search, the edge $e(v_l, l_1) = (v_l, v_3)$ and the subtree $T_{v_3}$ are searched before the edge $e(v_l, l_1 - 1)$ and the subtree $T_{e(v_l, l_1 - 1)}$ are searched.

When the search starts (from the root $v_0$), we already know that the tree $T = T_{v_0}$ is to be covered by a $p$-median whose distance-sum is $H(T_{v_0}, p)$, and we also know that the root $v_0$ itself is to be covered by the vertex $c(v_0, p)$. Suppose that when an edge $e(v_l, l_1) = (v_l, v_3)$ is searched we already know the number $k$ of points of the $p$-median of $T$ which are to cover the subtree $T_{e(v_l, l_1)}$, and in particular we know that vertex $v_l$ is to be covered by a certain vertex $v_2$. The distance-sum which corresponds to these $k$ points is therefore $S = R(k, e(v_l, l_1), v_2)$. Using (for all possible $k$'s) the values $R(k, e(v, l), v_2)$ and $H(T_{v_3}, k)$ and the equations (4.1)–(4.5), we can find the number $k_1$ of points of the $p$-median of $T$ which are to cover the subtree $T_{e(v_l, l_1 - 1)}$, and we can also find which point is to cover $v_3$; If $v_3$ is to be covered by $v_2$ then the search is continued on the subtree $T_{v_3}$ which is to be covered by $k - k_1 + 1$ points whose distance-sum is $S - R(k_1, e(v_l, l_1 - 1), v_2)$; else, the whole second phase should be recursively applied to the subtree $T_{v_3}$ in order to find a $(k - k_1)$-median whose distance-sum is $H(T_{v_3}, k - k_1)$, such that $v_3$ is to be covered by $c(v_3, k - k_1)$. In both cases, we continue the search on the subtree $T_{e(v_l, l_1 - 1)}$ to find $k_1$ points of the $p$-median of distance-sum $R(k_1, e(v_l, l_1 - 1), v_2)$ which cover this subtree. Thus, in the second phase we in fact, decompose the tree $T$ into $p$ disjoint components, each of which is to be covered by another point of the $p$-median, and therefore all the points of the $p$-median of $T$ are identified and located.

We now give a formal description of a (recursive) procedure FIND($p$, $T_v$) which finds a $p$-median of a tree $T_v$ whose root is $v$. In this algorithm we use a variable $M(v)$ which gives at each stage the number of points of the $p$-median of $T$ which cover that part of the tree $T_v$ which has not been searched yet. We use a set $V^*$ in order to store the points of the $p$-median of the original tree $T$ ($V^*_p$ is a universal storage which is common to all the recursive activations of the procedure FIND($p$, $T_v$)).

The second phase is started by assigning $V^*_p \leftarrow \emptyset$ and calling the procedure FIND($p$, $T_{v_0}$).

Algorithm 4.2.

**Algorithm 4.2: The recursive procedure FIND($p$, $T_v$).**

1. Let $v_2 \leftarrow c(v, p)$ [$v_2$ is a member of the $p$-median to be constructed]. Assign: $V^*_p \leftarrow V^*_p \cup \{v_2\}$. If $p = 1$ then Return [all the points of the $p$-median of $T_v$ are already contained in $V^*_p$]. Else, assign: $v_1 \leftarrow v$, $k \leftarrow p$, $M(v_1) \leftarrow k$.

2. [$v_1$ is covered by $v_2$]. Let $l_1$ be the number of sons of $v_1$ [since $k \neq 1$, $v_1$ is not a leaf]. Assign: $S \leftarrow R(k, e(v_1, l_1), v_2)$. Let $v_3$ be the vertex at the other end of $e(v_1, l_1)$ (namely, $e(v_1, l_1) = (v_1, v_3)$). If $v_3$ is a leaf then go to 3, else go to 4.

3. [$v_3$ is a leaf]. If $v_3 = v_2$, or if $S = R(k, e(v_1, l_1 - 1), v_2) + w(v_3) \cdot d(v_2, v_3)$ then $v_3$ is covered by $v_2$ go to 5. Else assign $V^*_p \leftarrow V^*_p \cup \{v_3\}$ [$v_3$ is a member of the $p$-median which covers only itself and $S = R(k - 1, e(v_1, l_1 - 1), v_2)$]. Assign $k \leftarrow k - 1$, $M(v_1) \leftarrow k$. If $k = 1$ then go to 6. Else go to 5.
4. \([v_3 \text{ is not a leaf}]. \) Let \(l_3\) be the number of sons of \(v_3\). If \(v_2 \not\in T_{v_3}\), and if for some \(k_1\), \(\max\{1, k - |T_{v_2}|\} \equiv k_1 \equiv \min\{k, -|T_{e(v_1, l_1-1)}| - A\}^5\) there exists: \(S = R(k_1, e(v_1, l_1-1), v_2) + H(T_{v_3}, k - k_1)\) [namely, \(v_3\) is not covered by \(v_2\)], then call \(\text{FIND}\ (k - k_1, T_{v_3})\), then assign \(k \leftarrow k_1, M(v_1) \leftarrow k_1, \text{ and if } k_1 = 1 \text{ then go to 6, else go to 5. Else } [v_3 \text{ is covered by } v_2]\) there exists \(k_1, \max\{1, k - |T_{v_2}|\} \equiv k_1 \equiv \min\{k, -|T_{e(v_1, l_1-1)}| - A\}^6\) such that \(S = R(k_1, e(v_1, l_1-1), v_2) + R(k + 1 - k_1, e(v_3, l_3), v_2). \)\ If \(k + 1 - k_1 = 1 \text{ then go to 5. Else, assign: } M(v_1) \leftarrow k_1, M(v_3) \leftarrow k + 1 - k_1, k \leftarrow k + 1 - k_1, \text{ and go to 2.}

5. \([\text{The next subtree to be searched is } T_{e(v_1, l_1-1)}]\). Assign: \(k \leftarrow M(v_1)\). \ If \(k_1 = 1, \) or if \(l_1 = 1 \text{ then go to 6. [The search of the subtree } T_{v_3} \text{ is already completed]. Else, assign } l_1 \leftarrow l_1 - 1, S \leftarrow R(k, e(v_1, l_1), v_2). \) Let \(v_3\) be the vertex at the other end of \(e(v_1, l_1)\) (namely, \(e(v_1, l_1) = (v_1, v_3)\)). \ If \(v_3\) is a leaf then go to 3, else go to 4.

6. \([\text{The search of the subtree } T_{v_3} \text{ is already completed}]. \) If \(v_1 = v \text{ then Return. Else, assign: } v_3 \leftarrow v_1 + v, v_3 \leftarrow \text{Father}(v_1), l_1 \leftarrow \{\text{the index for which } e(v_1, l_1) = (v_1, v_3)\}\) and go to 5.

The complexity of the second phase. During the second phase, each edge of the tree is traversed at most twice: In the first time (steps 2–5) the edge is traversed in the direction from its end-point of lower level to its end-point of higher level. During this traversal at most \(O(n)\) steps are required to find whether \(v_2 \in T_{v_3}\) or not and whether \(v_2 \in T_{e(v_1, l_1-1)}\) or not, and at most \(O(k)\) steps are required to find the appropriate \(k_1\) in step 4 of the algorithm. The second traversal of the edges (steps 3–6) is done in the direction opposite to the first traversal and it requires a constant number of steps. Thus the total complexity of the second phase is \(O(n^2)\).

4.4. Final remarks.
(a) As the complexity of the first phase is \(O(n^2 \cdot p^2)\), while the complexity of the second phase is \(O(n^2)\), the overall complexity of finding a \(p\)-median of a tree is therefore \(O(n^2 \cdot p^2)\).

It may be observed that during the first phase we compute all the values \(R(k, e(v_1, j_1), v_2), c(v, k)\) and \(H(T_{v_3}, k)\) for all possible values of \(k, 1 \leq k \leq p\). Therefore, we can repeat the second phase \(p\) times, where at the \(j\)th time \((j = 1, 2, \ldots, p)\) we find a \(\text{j-}\)median of the tree, using the same values of \(R(k, e(v_1, j_1), v_2), c(v, k)\) and \(H(T_{v_3}, k)\) as assigned during the sole performance of the first phase. Since the total number of steps which are required in this case by the second phase would be \(O(n^2 \cdot p)\), this implies that one can find \(j\)-medians for all \(j, 1 \leq j \leq p\), in no more than \(O(n^2 \cdot p^2)\) steps.

In particular, finding \(p\)-medians of a tree for all the \(n\) possible values of \(p\) \((1 \leq p \leq n)\), requires at most \(O(n^4)\) steps.

(b) The algorithm described in this section, can also be used to solve the following generalization of the \(p\)-median problem due to Matula and Kolde [9]: Given a set \(Y_{p_1}^*\) of \(p_1\) points on the tree, find a set \(Y_{p_2}^*\) of \(p_2\) points, such that

\[
\sum_{v \in V} w(v) \cdot d(v, Y_{p_1}^* \cup Y_{p_2}^*) = \min_{Y_{p_2}} \left\{ \sum_{v \in V} w(v) \cdot d(v, Y_{p_1}^* \cup Y_{p_2}) \right\}
\]

(namely, the set \(X_p^* = Y_{p_1}^* \cup Y_{p_2}^*\) is a "constrained \(p\)-median" where \(p_1\) of the points are fixed).
First we observe that without the loss of generality we may restrict the search for the \( p_2 \) points of \( Y^*_{p_2} \) to the vertices of the tree (even if the given \( p_1 \) points of \( Y^*_{p_1} \) are not located on vertices). In order to find the set \( Y^* \), we can use the above algorithm for finding a (regular) \( p \)-median of a tree (where \( p = p_1 + p_2 \)) with the following modifications: During the first phase, instead of computing the variables \( R(k, e(v, l), v) \) for any pair consisting of an edge \( e(v, l) \) and a vertex \( v \), we compute these variables only for those pairs consisting of an edge \( e(v, l) \) and any point \( v, v \in V \cup Y^*_{p_1} \) such that no point of \( Y^*_{p_1} \) lies on the path \( p(v, v) \) (except maybe \( v \), itself). It is not difficult to see that for each edge \( e(v, l) \) the total number of such points \( v \) is not greater than \( n \), and thus the total number of pairs consisting of an edge \( e(v, l) \) and a point \( v \), for which the values \( R(k, e(v, l), v) \) are computed is at most \( n^2 \). On the other hand, since \( p_1 \) of the points of the \( p \)-median are given, for each such a pair at most \( p_2 \) different values of \( k \) are relevant, and thus at most \( p_2 \) different values \( R(k, e(v, l), v) \) must be computed. Therefore, the total number of values \( R(k, e(v, l), v) \) which should be computed throughout the first phase is at most \( n^2 \cdot p_2 \), and the complexity which is involved is \( O(n^2 \cdot p_2) \). Similar considerations show that the complexity of computing the variables \( H(T_v, k) \) and \( c(v, k) \) is \( O(n^2 \cdot p_2) \). Since no modification of the second phase is required, this leads to the conclusion that the total complexity of the algorithm in this case is \( O(n^2 \cdot p_2) \).

**Appendix.** In this appendix we demonstrate how the algorithm of § 4 works on a given tree. The tree in our example is illustrated in Fig. 4. It consists of 9 vertices \( \{0, 1, \cdots, 8\} \), where vertex 0 was chosen to be the root. The structure and the parameters of the tree are given below ("the given data"). We show how to find a 4-median of this tree.

![Fig. 4. An example.](image)

**A1. The given data.**
(a) \( n = 9, p = 4. \)
(b) Vertex table (Table A.1)

<table>
<thead>
<tr>
<th>Vertex ( v )</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>( F(v) ) (the father of ( v ))</td>
<td>—</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>7</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>( w(v) ) (the weight of ( v ))</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Length of the edge ( (v, F(v)) )</td>
<td>—</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>3</td>
</tr>
</tbody>
</table>
A2. Preprocessing.

(a) $L_m = 3$.
(b) A list of vertices according to their levels, and the order imposed on the edges (Table A.2).

<table>
<thead>
<tr>
<th>The Level, $\text{Lev}(v)$</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>vertex $v$</td>
<td>8</td>
<td>6</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>Number of sons of $v$</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>$e(v, 1)$</td>
<td>(7, 6)</td>
<td>(4, 5)</td>
<td>(3, 1)</td>
<td>(0, 3)</td>
</tr>
<tr>
<td>$e(v, 2)$</td>
<td>(7, 8)</td>
<td>(4, 7)</td>
<td>(3, 2)</td>
<td>(0, 4)</td>
</tr>
<tr>
<td>$e(v, 3)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$e(v, 4)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(c) The distance-matrix.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>—</td>
<td>7</td>
<td>9</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>12</td>
<td>8</td>
<td>11</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>—</td>
<td>8</td>
<td>3</td>
<td>10</td>
<td>12</td>
<td>19</td>
<td>15</td>
<td>18</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>8</td>
<td>—</td>
<td>5</td>
<td>12</td>
<td>14</td>
<td>21</td>
<td>17</td>
<td>20</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>—</td>
<td>7</td>
<td>9</td>
<td>16</td>
<td>12</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>10</td>
<td>12</td>
<td>7</td>
<td>—</td>
<td>2</td>
<td>9</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>12</td>
<td>14</td>
<td>9</td>
<td>2</td>
<td>—</td>
<td>11</td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>12</td>
<td>19</td>
<td>21</td>
<td>16</td>
<td>9</td>
<td>11</td>
<td>—</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>15</td>
<td>17</td>
<td>12</td>
<td>5</td>
<td>7</td>
<td>—</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>11</td>
<td>18</td>
<td>20</td>
<td>15</td>
<td>8</td>
<td>10</td>
<td>7</td>
<td>3</td>
<td>—</td>
</tr>
</tbody>
</table>

(d) A list of the leaves of the tree (arranged by their levels): 8, 6, 5, 2, 1.
(e) The subtrees $T_{e(v,f)}$ and $T_{e(v,f)} - T_{e(v,f-1)}$, where these subtrees as well as the vertices of each subtree are arranged according to nonincreasing order of the levels:

(i) $\text{Lev}(v) = 3$:

\[ T_8 = T_{e(8,0)} = \{8\} \]
\[ T_6 = T_{e(6,0)} = \{6\} \]
(ii) \text{Lev} (v) = 2:

\[ T_{e(7,0)} = \{7\} \]
\[ T_{e(7,1)} = \{6, 7\} \quad T_{e(7,1)} - T_{e(7,0)} = \{6\} \]
\[ T_7 = T_{e(7,2)} = \{8, 6, 7\} \quad T_{e(7,2)} - T_{e(7,1)} = \{8\} \]
\[ T_5 = T_{e(5,0)} = \{5\} \]
\[ T_2 = T_{e(2,0)} = \{2\} \]
\[ T_1 = T_{e(1,0)} = \{1\} \]

(iii) \text{Lev} (v) = 1:

\[ T_{e(4,0)} = \{4\} \]
\[ T_{e(4,1)} = \{5, 4\} \quad T_{e(4,1)} - T_{e(4,0)} = \{5\} \]
\[ T_4 = T_{e(4,2)} = \{8, 6, 7, 5, 4\} \quad T_{e(4,2)} - T_{e(4,1)} = \{8, 6, 7\} \]
\[ T_{e(3,0)} = \{3\} \]
\[ T_{e(3,1)} = \{1, 3\} \quad T_{e(3,1)} - T_{e(3,0)} = \{1\} \]
\[ T_3 = T_{e(3,2)} = \{2, 1, 3\} \quad T_{e(3,2)} - T_{e(3,1)} = \{2\} \]

(iv) \text{Lev} (v) = 0:

\[ T_{e(0,0)} = \{0\} \]
\[ T_{e(0,1)} = \{2, 1, 3, 0\} \quad T_{e(0,1)} - T_{e(0,0)} = \{2, 1, 3\} \]
\[ T = T_0 = T_{e(0,2)} = \{8, 6, 7, 5, 2, 1, 4, 3, 0\} \quad T_{e(0,2)} - T_{e(0,1)} = \{8, 6, 7, 5, 4\} \]

A3. First phase.

The first phase is summarized in Table A.3 where:

(a) In the stage which corresponds to level \(L_m - i\), all the sums of the subtrees whose roots are of level \(L_m - i\) are computed (column 1).

(b) In the substage which corresponds to vertex \(v_s\), the sums of subtree \(T_{e_s}\) are treated (column 2).

(c) In the sub-substage which corresponds to edge \(e(v_s, l)\), all the sums of subtree \(T_{e(v_s,l)}\) are computed (column 3).

(d) In column 5 we list the step of the algorithm in which the sums of column 4 are computed, where in parentheses appear the steps which precede the current one.

[Because of space limitations, we list in Table A.3 only the first 2 values of \(R(k, e(v_1, j_1), v_2)\) which are computed for each edge \(e(v_n, l)\) in the first phase.]
Table A.3

<table>
<thead>
<tr>
<th>Level $L_m - i$</th>
<th>Vertex $v_i$</th>
<th>Edge $e(v_i, i)$</th>
<th>The computed sums</th>
<th>The steps in the algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_m - i = 3$</td>
<td>$v_7 = 7$</td>
<td>$e(7, 1) = (7, 6)$</td>
<td>For each leaf of $T$</td>
<td>For each vertex $v$ of $T$</td>
</tr>
<tr>
<td>$L_m - i = 2$</td>
<td>$v_7 = 7$</td>
<td>$e(7, 1) = (7, 6)$</td>
<td>$R(1, e(6, 0), 7) \leftarrow 20$</td>
<td>$R(1, e(v, 0), v) \leftarrow 0$</td>
</tr>
<tr>
<td>$L_m - i = 1$</td>
<td>$v_7 = 7$</td>
<td>$e(7, 1) = (7, 6)$</td>
<td>$R(1, e(7, 0), 6) \leftarrow 12$</td>
<td>$(2, 3, 4, 5, 6, 7)$</td>
</tr>
<tr>
<td></td>
<td>$v_4 = 4$</td>
<td>$e(4, 1) = (4, 5)$</td>
<td>$R(1, e(5, 0), 4) \leftarrow 4$</td>
<td>$(3, 2, 3, 4, 5, 6, 7)$</td>
</tr>
<tr>
<td></td>
<td>$v_3 = 3$</td>
<td>$e(3, 1) = (3, 1)$</td>
<td>$R(1, e(1, 0), 3) \leftarrow 15$</td>
<td>$(3, 4, 5, 6, 7)$</td>
</tr>
</tbody>
</table>

$T_7 = \{8, 6, 7\}$
$T_{e(7,1)} = \{6, 7\}$
$T_8 = \{8, 6, 7, 5, 4\}$
$T_{e(8,1)} = \{5, 4\}$
$T_5 = \{2, 1, 3\}$
$T_{e(3,1)} = \{1, 3\}$

$H(T_7, 1) = 32; c(7, 1) = 7$
$H(T_7, 2) = 9; c(7, 2) = 8$
$H(T_7, 3) = 0; c(7, 3) = 7$

$H(T_4, 1) = 66; c(4, 1) = 7$
$H(T_4, 2) = 36; c(4, 2) = 4$
$H(T_4, 3) = 13; c(4, 3) = 4$
$H(T_4, 4) = 4; c(4, 4) = 4$

$H(T_3, 1) = 7; c(3, 1) = 9$
$H(T_3, 2) = 9; c(3, 2) = 9$
$H(T_3, 3) = 7; c(3, 3) = 7$

$H(T_2, 1) = 3; c(2, 1) = 3$
$H(T_2, 2) = 6; c(2, 2) = 6$
$H(T_2, 3) = 1; c(2, 3) = 1$
$H(T_2, 4) = 1; c(2, 4) = 1$

$H(T_1, 1) = 1; c(1, 1) = 1$
$H(T_1, 2) = 2; c(1, 2) = 2$
$H(T_1, 3) = 3; c(1, 3) = 3$
$H(T_1, 4) = 4; c(1, 4) = 4$

$H(T_0, 1) = 1; c(0, 1) = 1$
$H(T_0, 2) = 2; c(0, 2) = 2$
$H(T_0, 3) = 3; c(0, 3) = 3$
$H(T_0, 4) = 4; c(0, 4) = 4$
<table>
<thead>
<tr>
<th>Level $L_{m}-i$</th>
<th>Vertex $v_{i}$</th>
<th>Edge $e(v_{i}, l)$</th>
<th>The computed sums</th>
<th>The steps in the algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_{m}-i = 0$</td>
<td>$v_{0} = 0$</td>
<td>$e(0, 1) = (0, 3)$</td>
<td>$R(1, e(2, 0), 0) \leftarrow 54$</td>
<td>$(3, 2, 3, 4, 5, 6), 7$</td>
</tr>
<tr>
<td></td>
<td>$T_{0} = {8, 6, 7, 5, 2, 1, 4, 3, 0}$</td>
<td>$e(0, 2) = (0, 4)$</td>
<td>$R(1, e(6, 0), 2) \leftarrow 105$</td>
<td>$7$</td>
</tr>
<tr>
<td></td>
<td>$T_{e(0,1)} = {2, 1, 3, 0}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$e(0, 2) = (0, 4)$</td>
<td>$T_{e(0,2)} = {8, 6, 7, 5, 2, 1, 4, 3, 0}$</td>
<td>$R(1, e(8, 0), 2) \leftarrow 80$</td>
<td>$(21, 4, 5, 6), 7$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$R(1, e(6, 0), 2) \leftarrow 105$</td>
<td>$7$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$R(1, e(0, 2), 7) \leftarrow 319$</td>
<td>$21$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$R(2, e(0, 2), 7) \leftarrow 151$</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$R(3, e(0, 2), 7) \leftarrow 115$</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$R(4, e(0, 2), 7) \leftarrow 95$</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$R(1, e(0, 2), 4) \leftarrow 254$</td>
<td>$21$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$R(2, e(0, 2), 4) \leftarrow 156$</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$R(3, e(0, 2), 4) \leftarrow 96$</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$R(4, e(0, 2), 4) \leftarrow 60$</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$H(T_{0}, 1) \leftarrow 251; c(0, 1) \leftarrow 0$</td>
<td>$(21, 4, 22)$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$H(T_{0}, 2) \leftarrow 131; c(0, 2) \leftarrow 3$</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$H(T_{0}, 3) \leftarrow 96; c(0, 3) \leftarrow 4$</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$H(T_{0}, 4) \leftarrow 60; c(0, 4) \leftarrow 4$</td>
<td></td>
</tr>
</tbody>
</table>

The values $H(T_{0}, k)$ and $c(0, k)$

Proceed to the second phase.
The distance-sum which corresponds to a 4-median of the tree is $H(T_{0}, 4) = 60$. The root (vertex 0) is covered by the vertex $c(0, 4) = 4$. (3), 2
The second phase is summarized in Table A.4.

<table>
<thead>
<tr>
<th>Depth of recursion</th>
<th>k-median to be found</th>
<th>Subtree to be searched</th>
<th>Sequence of steps in the algorithm</th>
<th>Points which are added to $V_2^*$ (Steps 1, 3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4-median of $T_0$</td>
<td>$T_0 = {8, 6, 7, 5, 2, 1, 4, 3, 0}$</td>
<td>Start</td>
<td>$V_2^* \leftarrow \emptyset$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>$V_2^* \leftarrow V_2^* \cup {4}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$T_0 = {8, 6, 7, 5, 2, 1, 4, 3, 0}$</td>
<td>2, 4</td>
<td></td>
</tr>
<tr>
<td>2-median of $T_4$</td>
<td></td>
<td>$T_4 = {8, 6, 7, 5, 4}$</td>
<td>2, 4</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1-median of $T_4$</td>
<td>$T_4 = {8, 6, 7}$</td>
<td>1</td>
<td>$V_4^* \leftarrow V_4^* \cup {7}$</td>
</tr>
<tr>
<td>1-median of $T_{e(4,1)}$</td>
<td>$T_{e(4,1)} = {5, 4}$</td>
<td>4, 6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3-median of $T_{e(0,1)}$</td>
<td>$T_{e(0,1)} = {2, 1, 3, 0}$</td>
<td>5, 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2-median of $T_3$</td>
<td>$T_3 = {2, 1, 3}$</td>
<td>2</td>
<td>$V_4^* \leftarrow V_4^* \cup {1}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2, 3</td>
<td>$V_4^* \leftarrow V_4^* \cup {2}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$T_3 = {2, 1, 3}$</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>1-median of $T_{e(0,0)}$</td>
<td>$T_{e(0,0)} = {0}$</td>
<td>4, 6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A 4-median of the tree is the set $V_4^* = \{4, 7, 1, 2\}$, and the corresponding distance-sum is $H(T, 4) = 60$.

Using the same data which was computed in the first phase, we can apply the second phase with $p$ having the values 3, 2 or 1, in which cases we find that (see § 4.4):

A 3-median of the tree is the set $V_3^* = \{4, 7, 3\}$ and the sum is $H(T, 3) = 96$.

A 2-median of the tree is the set $V_2^* = \{3, 7\}$ and the sum is $H(T, 2) = 131$.

A 1-median of the tree is the set $V_1^* = \{0\}$ and the sum is $H(T, 1) = 251$.

REFERENCES


