8
Algorithm for Binary Searching in Trees

In this section we present our algorithm for binary searching in trees. A crucial observation employed by the algorithm is that this problem can be efficiently solved when the input is a path-like tree. This is true because it can be easily reduced to the well-solved problem of searching a hidden marked element from a total order set U in a sorted list $L \subseteq U$ of elements where each element U has a given probability of being the marked one [PS93] (see appendix). Due to this correspondence, an approximate strategy for searching in lists gives an approximation (with the same guarantee) for searching in path-like trees.

Motivated by this observation, the algorithm decomposes the input tree into special paths, finds decision trees for each of these paths (with modified weight functions) and combine them into a decision tree for the original tree. In our analysis, we obtain a lower bound on the optimal solution and an upper bound on the returned solution in terms of the costs of the decision trees for the paths. Thus, the approximation guarantee of the algorithm is basically a constant times the guarantee of the approximation used to compute the decision trees for the paths. Throughout the text, we present the execution and analysis of the algorithm over an instance (T, w), where T is rooted at node r.

Recall that for every node $u \in T$, the *cumulative weight* of u is the sum of the weights of its descendants, namely $w(T_u)$. Moreover, recall that a *heavy path* Q of T is defined recursively as follows: r belongs to Q; for every node u in Q, the non-leaf children of u with greatest cumulative weight also belongs to Q.

Let $Q = (q_1 \rightarrow \ldots \rightarrow q_{|Q|})$ be a heavy path of T. We define $\overline{T}_{q_i} = T_{q_i} - T_{q_{i+1}}$, for $i < |Q|$ and $\overline{T}_{q_{|Q|}} = T_{q_{|Q|}}$. In addition, we define $T^j_{q_i}$ as the jth heaviest maximal subtree rooted at a child of q_i not in Q (Figure 8.1). Note that these definitions are slightly different from the ones presented in the previous part of this work. Finally, let n_i denote the number of children of q_i

which do not belong to Q and define e_i^j as the arc of T connecting the node q_i to the subtree $T_{q_i}^j$.

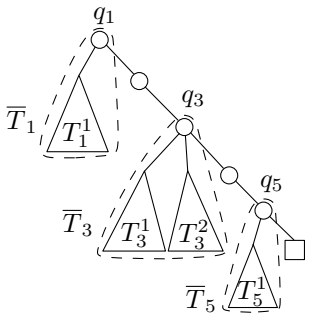

Figure 8.1: Example of structures $\overline{T}_{q_i}$ and $T_{q_i}^j$.

Now we explain the high level structure of the solution returned by the algorithm. The main observation is that we can break the task of finding the marked node in three stages: first finding the node q_i of Q such that $\overline{T}_{q_i}$ contains the marked node, then querying the arcs $\{e_i^j\}_j$ to discover which tree $T_{q_i}^{j'}$ contains the marked node or that the marked node is q_i, and finally (if needed) locate the marked node in $T_{q_i}^{j'}$. The algorithm follows this reasoning: it computes a decision tree for the heavy path Q, then a decision tree for querying the arcs $\{e_i^j\}$ and recursively computes a decision tree for each tree $T_{q_i}^j$.

Now we present the algorithm itself, which consists of the following five steps:

(i) Find a heavy path Q of T and then for each $q_i \in Q$ define $w'(q_i) = w(\overline{T}_{q_i})/w(T)$.

(ii) Calculate a decision tree D_Q for the instance (Q, w') using the approximation algorithm presented in [PS93].

(iii) Calculate recursively a decision tree D_i^j for each instance $(T_{q_i}^j, w)$.

(iv) Build a decision tree D_i for each $\overline{T}_{q_i}$ as follows. The leftmost path of D_i consists of nodes corresponding to the arcs $e_i^1, \ldots, e_i^{n_i}$, with a node u_{q_i} appended at the end. In addition, for every j, D_i^j is the right child of the node corresponding to e_i^j in D_i (Figure 8.2).

(v) Construct the decision tree D for T by replacing the leaf u_{q_i} of D by D_i, for each $q_i \in Q$ (Figure 8.3).

It is not difficult to check that the decision tree D computed by the algorithm is a valid decision tree for T.

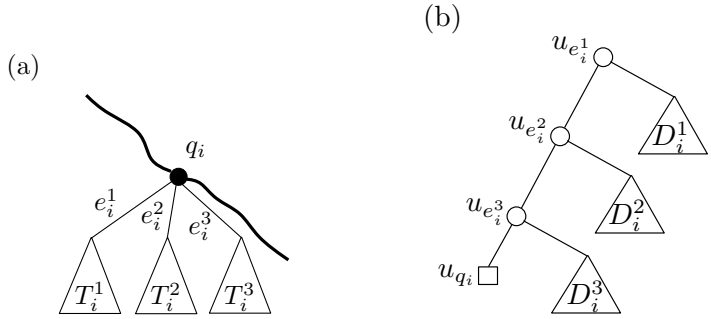

Figure 8.2: Illustration of Step (iv) of the algorithm. (a) Tree T with heavy path in bold. (b) Decision tree D_i for $\overline{T}_i$.

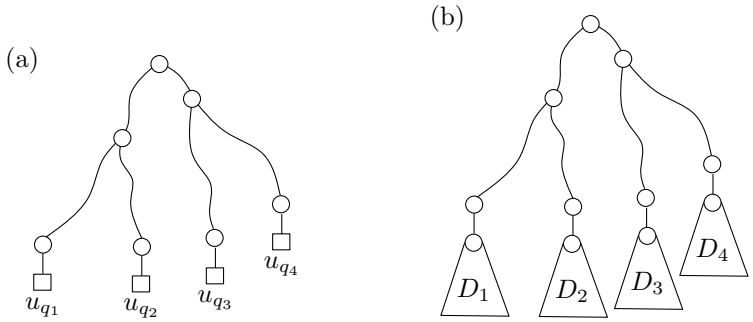

Figure 8.3: Illustration of Step (v) of the algorithm. (a) Decision tree D_Q built at Step (ii). (b) Decision tree D constructed by replacing the leaves $\{u_{q_i}\}$ by the decision trees $\{D_i\}$.

8.1 Upper Bound

As the trees $\{T_{q_i}^j\}$ and Q form a partition of the nodes of T, we analyze the distance of the root of D to the nodes in each of these structures separately in order to upper bound the cost of D.

First, consider a tree $T_{q_i}^j$ and let x be a node in $T_{q_i}^j$. Noticing that u_x is a leaf of the tree D_i^j, the path from $r(D)$ to u_x in D must contain the node $r(D_i^j)$. Then, by the construction made in Step (iv), the path from $r(D)$ to u_x in D has the form $(r(D) \rightsquigarrow u_{e_i^1} \rightarrow u_{e_i^2} \rightarrow \ldots \rightarrow u_{e_i^j} \rightsquigarrow r(D_i^j) \rightsquigarrow u_x)$. Notice that the path $(r(D) \rightsquigarrow u_{e_i^1})$ in D is the same as the path $(r(D) \rightsquigarrow u_{q_i})$ in D_Q. In addition, the path from $r(D_i^j)$ to u_x is the same in D and in D_i^j. Employing the previous observations, we have that the length of the path $(r(D) \rightsquigarrow u_{e_i^1} \rightarrow u_{e_i^2} \rightarrow \ldots \rightarrow u_{e_i^j} \rightsquigarrow r(D_i^j) \rightsquigarrow u_x)$ is:

$$d(r(D), u_x, D) = d(r(D), u_{q_i}, D_Q) + j + d(r(D_i^j), u_x, D_i^j)$$

Now we consider a node $q_i \in Q$. Again due to the construction made in Step (iv) of the algorithm, it is not difficult to see that the path from $r(D)$ to u_{q_i} in D traverses the leftmost path of D_i, that is, this path has the form

$(r(D) \rightsquigarrow u_{e_i^1} \rightarrow u_{e_i^2} \rightarrow \ldots \rightarrow u_{e_i^{n_i}} \rightarrow u_{q_i})$. Because the path $(r(D) \rightsquigarrow u_{e_i^1})$ in D is the same as the path $(r(D) \rightsquigarrow u_{q_i})$ in D_Q, it follows that the length of the path from the root of D to u_{q_i} is $d(r(D), u_{q_i}, D_Q) + n_i$.

Weighting the distance to reach nodes in $\{T_{q_i}^j\}$ and in Q, we find the cost of D:

$$\begin{aligned}
\text{cost}(D,w) &= \sum_{q_i \in Q} \sum_j d(r(D), u_{q_i}, D_Q) w(T_{q_i}^j) + \sum_{q_i \in Q} \sum_j j \cdot w(T_{q_i}^j) \\
&\quad + \sum_{q_i \in Q} \sum_j \sum_{x \in T_{q_i}^j} d(r(D_i^j), u_x, D_i^j) w(x) \\
&\quad + \sum_{q_i \in Q} d(r(D), u_{q_i}, D_Q) w(q_i) + \sum_{q_i \in Q} n_i w(q_i) \\
&= \sum_{q_i \in Q} d(r(D), u_{q_i}, D_Q) w(\overline{T}_{q_i}) + \sum_{q_i \in Q} \sum_j j \cdot w(T_{q_i}^j) \\
&\quad + \sum_{q_i \in Q} \sum_j \text{cost}(D_i^j, w) + \sum_{q_i \in Q} n_i w(q_i) \\
&= w(T) \sum_{q_i \in Q} d(r(D), u_{q_i}, D_Q) w'(q_i) + \sum_{q_i \in Q} \sum_j j \cdot w(T_{q_i}^j) \\
&\quad + \sum_{q_i \in Q} \sum_j \text{cost}(D_i^j, w) + \sum_{q_i \in Q} n_i w(q_i) \\
&= w(T) \cdot \text{cost}(D_Q, w') + \sum_{q_i \in Q} \sum_j j \cdot w(T_{q_i}^j) \\
&\quad + \sum_{q_i \in Q} \sum_j \text{cost}(D_i^j, w) + \sum_{q_i \in Q} n_i w(q_i) \qquad (1)
\end{aligned}$$

Now all we need is to upper bound the first term of the right-hand side of previous equality. Notice that $\text{cost}(D_Q, w')$ is exactly the cost of the approximation computed at Step (ii) of the algorithm. As mentioned previously, we can use the algorithm of [PS93] in this step to find a decision tree D_Q with cost at most $H(\{w'(q_i)\}) + 2$. Substituting this bound on equality (1) and observing that $w(T) \cdot H(\{w'(q_i)\}) = H(\{w(\overline{T}_{q_i})\})$, we conclude the upper bound:

$$\begin{aligned}
\text{cost}(D,w) &\leq H(\{w(\overline{T}_{q_i})\}) + 2w(T) + \sum_{q_i \in Q} \sum_j j \cdot w(T_{q_i}^j) \\
&\quad + \sum_{q_i \in Q} \sum_j \text{cost}(D_i^j, w) + \sum_{q_i \in Q} n_i w(q_i) \qquad (2)
\end{aligned}$$

8.2 Entropy Lower Bound

In this section we present a lower bound on the cost of an optimal decision tree for (T, w). Hence, let D^* be a minimum cost decision tree for (T, w), and let r^* be the root of D^*.

Consider a tree $T^j_{q_i}$ and let x be a node in $T^j_{q_i}$. By definition, the representative of $T^j_{q_i}$ in D^* (the node $u(T^j_{q_i})$) is an ancestor of the node u_x in D^*. Notice that the representative of $T^j_{q_i}$ is a node in D^* that corresponds to some arc (i', j') of $T^j_{q_i}$ (that is $u(T^j_{q_i}) = u_{(i',j')}$) and that (i', j') is also an arc of $\overline{T}_{q_i}$. Therefore, the definition of representative again implies that $u(T^j_{q_i})$ is a descendant of $u(\overline{T}^j_{q_i})$. Combining the previous observations, we have that the path in D^* from r^* to u_x has the form $(r^* \rightsquigarrow u(\overline{T}_{q_i}) \rightsquigarrow u(T^j_{q_i}) \rightsquigarrow u_x)$.

Now consider a node $q_i \in Q$; again by the definition of representative, the path from r^* to u_{q_i} can be written as $(r^* \rightsquigarrow u(\overline{T}_{q_i}) \rightsquigarrow u_{q_i})$. Adding the weighted paths for nodes in $\{T^j_{q_i}\}$ and in Q, we can write the cost of D^* as:

$$\begin{aligned} \mathrm{OPT}(T, w) &= \sum_{q_i} d(r^*, u(\overline{T}_{q_i}))w(\overline{T}_{q_i}) + \sum_{q_i}\sum_{j} d(u(\overline{T}_{q_i}), u(T^j_{q_i}))w(T^j_{q_i}) \\ &\quad + \sum_{q_i}\sum_{j}\sum_{x \in T^j_{q_i}} d(u(T^j_{q_i}), u_x)w(x) \\ &\quad + \sum_{q_i} d(u(\overline{T}_{q_i}), u_{q_i})w(q_i) \end{aligned} \tag{3}$$

Now we lower bound each term of the last equation. The idea to analyze the first term is the following: it can be seen as the cost of the decision tree D^* under a cost function where the representative of $\overline{T}_{q_i}$, for every i, has weight $w(\overline{T}_{q_i})$ and all other nodes have weight zero. Since D^* is a binary tree, we can use Shannon's Coding Theorem to guarantee that the first term of (3) is lower bounded by $\approx H(\{w(\overline{T}_{q_i})\})/c$ for some constant c. Then we have the following lemma, which is proved more formally in the appendix:

Lemma 14 $\sum_{q_i \in Q} d(r^*, u(\overline{T}_{q_i}), D^*)w(\overline{T}_{q_i}) \geq H(\{w(\overline{T}_{q_i})\})/\log 3 - w(T)$

Now we bound the second term of (3). Fix a node $q_i \in Q$; consider two different trees $T^j_{q_i}$ and $T^{j'}_{q_i}$ such that $d(r^*, u(T^j_{q_i})) = d(r^*, u(T^{j'}_{q_i}))$. We claim that $u(T^j_{q_i})$ and $u(T^{j'}_{q_i})$ are siblings in D^*. Because their distances to r^* are the same, $u(T^j_{q_i})$ cannot be a neither a descendant nor an ancestor of $u(T^{j'}_{q_i})$. Therefore, they have a common ancestor, say $u_{(v,z)}$, and one of these node is in the right subtree of $u_{(v,z)}$ and the other in the left subtree of $u_{(v,z)}$. It is not difficult to see that $u_{(v,z)}$ can only correspond to either (q_i, j) or (q_i, j'). Without loss of generality suppose the latter; then the right child of $u_{(v,z)}$ corresponds to an

arc/node of $T^{j'}_{q_i}$, and therefore $u(T^{j'}_{q_i})$ must be this child. Due to their distance to r^*, it follows that $u(T^{j}_{q_i})$ must be the other child of $u_{(v,z)}$.

As a consequence, we have that for any level ℓ of D^*, there are at most two representatives of the trees $T^j_{q_i}$ located at ℓ. This together with the fact that $T^j_{q_i}$ is the jth heaviest tree rooted at a child of q_i guarantees that

$$\begin{aligned}\sum_{j=1}^{n_i} d(u(\overline{T}_{q_i}), u(T^j_{q_i}))w(T^j_{q_i}) &\geq \sum_{j=1}^{n_i} \lfloor (j-1)/2 \rfloor w(T^j_{q_i}) \\ &\geq \sum_{j=1}^{n_i} \left(\frac{j}{2} - \frac{3}{2}\right) w(T^j_{q_i}) \qquad (4)\end{aligned}$$

and the last inequality gives a bound for the second term of (3).

Directly employing Lemma 13, we can lower bound the third term of (3) by $\sum_{q_i,j} \mathrm{OPT}(T^j_{q_i}, w)$.

Finally, for the fourth term we fix q_i and note that the path in D^* connecting $u(\overline{T}_{q_i})$ to u_{q_i} must contain the nodes corresponding to arcs $e^1_i, \ldots, e^{n_i}_i$ (otherwise when traversing D^* and reaching u_{q_i} we would not have enough knowledge to infer that q_i is the marked node, contradicting the validity of D^*). Applying this reasoning for each q_i, we conclude that last term of (3) is lower bounded by $\sum_{q_i} n_i \cdot w(q_i)$.

Therefore, applying the previous discussion to lower bound the terms of (3) we obtain that:

$$\begin{aligned}\mathrm{OPT}(T, w) \geq\ & \frac{H(\{w(\overline{T}_{q_i})\})}{c} - \frac{5w(T)}{2} + \sum_{q_i,j} \frac{j \cdot w(T^j_{q_i})}{2} \\ & + \sum_{q_i,j} \mathrm{OPT}(T^j_{q_i}, w) + \sum_{q_i} n_i w(q_i) \qquad (5)\end{aligned}$$

8.3 Alternative Lower Bounds

As in the case of the k-Hotlink Assignment Problem presented in previous sections, when the value of the entropy $H(\{w(\overline{T}_{q_i})\})$ is large enough, it dominates the term $-5w(T)/2$ in inequality (5) and leads to a sufficiently strong lower bound. However, when this entropy assumes a small value we need to adopt a different strategy to devise an effective bound.

First alternative lower bound. In order to reduce the additive factor that appears in (5), we use almost the same derivation that leads from (3) to (5); however, we simply lower bounded the first summation of (3) by zero.

This gives:

$$\begin{aligned}\mathrm{OPT}(T,w) \;\geq\; & -\frac{3w(T)}{2}+\sum_{q_i,j}\frac{j\cdot w(T^j_{q_i})}{2} \\ & +\sum_{q_i,j}\mathrm{OPT}(T^j_{q_i},w)+\sum_{q_i\in Q}n_i w(q_i) \qquad (6)\end{aligned}$$

Second alternative lower bound. Now we devise a lower bound without an additive factor; however it also does not contain the important term $\sum_{q_i\in Q} n_i w(q_i)$. Consider a tree $T^j_{q_i}$ and a node $v \in T^j_{q_i}$. By the definition of representative, $u(T^j_{q_i})$ is an ancestor of u_v in D^*, thus $d(r^*,u_v,D^*) = d(r^*,u(T^j_{q_i}),D^*) + d(u(T^j_{q_i}),u_v,D^*)$. Because the trees $\{T^j_{q_i}\}$ and Q form a partition of nodes of T, the cost $\mathrm{OPT}(T,w)$ can be written as:

$$\mathrm{OPT}(T,w)=\sum_{q_i,j} d(r^*,u(T^j_{q_i}))w(T^j_{q_i})+\sum_{q_i\in Q} d(r^*,u_{q_i})w(q_i)+\sum_{q_i,j}\sum_{v\in T^j_{q_i}} d(u(T^j_{q_i}),u_v)w(v)$$

First, as the trees $\{T^j_{q_{|Q|}}\}$ are single nodes and hence do not contain any arcs, $\{u(T^j_{q_{|Q|}})\}$ and $\{u_{q_i}\}$ cannot be the root of D^*. Therefore, at most one distance $d(r^*,u(T^j_{q_i}))$ (and with $q_i \neq q_{|Q|}$) of the first two summations of the previous inequality can be equal to zero, and all others must have value of at least one. But the construction of the heavy path guarantees that for $q_i \neq q_{|Q|}$ the weight of each of the trees $\{T^j_{q_i}\}$ is at most $w(T)/2$. As a consequence, the first two summations of the inequality can be lower bounded by $w(T)/2$. Combining this fact with a lower bound for the last summation provided by Lemma 13 (with $T' = T^j_{q_i}$) we have:

$$\mathrm{OPT}(T,w) \geq \frac{w(T)}{2}+\sum_{q_i,j}\mathrm{OPT}(T^j_{q_i},w) \qquad (7)$$

8.4 Approximation Guarantee

We proceed by induction over the number of nodes of T, where the base case is the trivial one when T has only one node. Notice that because each tree $T^j_{q_i}$ is properly contained in T, the inductive hypothesis asserts that D^j_i is an approximate decision tree for $T^j_{q_i}$, namely $\mathrm{cost}(D^j_i,w) \leq \alpha\mathrm{OPT}(T^j_{q_i},w)$ for some constant α. The analysis needs to be carried out in two different cases depending on the value of the entropy (henceforth we use H as a shorthand for $H(\{w(\overline{T}_{q_i})\})$.

Case 1: $H/\log 3 \geq 3w(T)$. Applying this bound to inequality (5) we have:

$$\mathrm{OPT}(T,w) \geq \frac{H}{6\log 3} + \sum_{q_i,j} \frac{j \cdot w(T^j_{q_i})}{2} + \sum_{q_i,j} \mathrm{OPT}(T^j_{q_i}, w) + \sum_{q_i} n_i w(q_i) \quad (8)$$

Employing the entropy hypothesis to the first term of inequality (2) and using the inductive hypothesis we have:

$$\mathrm{cost}(D,w) \leq \frac{H(3\log 3+2)}{3\log 3} + \sum_{q_i,j} \left(j \cdot w(T^j_{q_i})\right) + \sum_{q_i,j} \alpha\mathrm{OPT}(T^j_{q_i}, w) + \sum_{q_i} n_i w(q_i)$$

Setting $\alpha \geq 2(3\log 3+2)$ it follows from the previous inequality and from inequality (8) that $\mathrm{cost}(D,w) \leq \alpha\mathrm{OPT}(T,w)$.

Case 2: $H/\log 3 \leq 3w(T)$. Applying the entropy hypothesis and the inductive hypothesis to inequality (2) we have:

$$\mathrm{cost}(D,w) \leq (3\log 3+2)w(T) + \sum_{q_i,j} \left(j \cdot w(T^j_{q_i})\right) + \sum_{q_i,j} \alpha\mathrm{OPT}(T^j_{q_i}, w) + \sum_{q_i} n_i w(q_i)$$

Adding $(\alpha-1)$ times inequality (7) to inequality (6) we have:

$$\alpha\mathrm{OPT}(T,w) \geq \frac{(\alpha-4)w(T)}{2} + \sum_{q_i,j} \frac{j \cdot w(T^j_{q_i})}{2} + \sum_{q_i,j} \alpha\mathrm{OPT}(D^j_i, w) + \sum_{q_i} n_i w(q_i)$$

Setting $\alpha \geq 2(3\log 3+2)+4$ we have that $\mathrm{cost}(T,w) \leq \alpha\mathrm{OPT}(T,w)$. Therefore, the inductive step holds for both cases when $\alpha \geq 2(3\log 3+2)+4$.

8.5 Efficient Implementation

Notice that during the presentation and analysis of the algorithm, we have assumed that the trees $\{T^j_{q_i}\}$ are ordered by their weights. However, in order to actually implement Step (iv) of the algorithm, one needs to sort these trees. As a heavy path decomposition can be computed in linear time [DL06], it is easy to see that all steps of the steps of the algorithm, besides this sorting at Step (iv), can be implemented in linear time. Hence, we resort to an 'approximate sorting' to provide a linear time implementation for the algorithm.

Fix a node q_i in Q. Without loss of generality, we assume there are more than three trees $\{T^j_{q_i}\}$ (that is $n_i > 3$), otherwise one could sort them in constant time. In order to simplify the analysis, we use $w_j = w(T^j_{q_i})$ and $W = \max_j\{w_j\}$. Then we have the sequence $WQ = (w_1, w_2, \ldots, w_{n_i})$, which is non-decreasing by the definition of the trees $\{T^j_{q_i}\}$. Now we partition WQ

into blocks with similar weights: for $1 \leq j < n_i$, the j-block contains all elements of WQ with weights in the interval $(W/2^j, W/2^{j-1}]$ and the n_i-block contains all elements of WQ with weights in $[0, W/2^{n_i-1}]$. Due to its order, WQ is the concatenation of these blocks. Defining s_j as the index in WQ of the first element of the j-block[1], we have that the for $j < n_i$ the j-block is the sequence $(w_{s_j}, w_{s_j+1}, \ldots, w_{s_{j+1}-1})$ and the n_i-block is the sequence $(w_{s_{n_i}}, w_{s_{n_i}+1}, \ldots, w_{n_i})$.

Now we say that WQ' is an *approximate sorting* of the weights $\{w_j\}$ if WQ' contains first all elements 1-block of WQ, then all elements of the 2-block of WQ and so on. The sequence WQ' can be thought as a permutation of WQ where only elements *on the same j-block* can be permuted. Defining $WQ' = (w_{\sigma(1)}, w_{\sigma(2)}, \ldots, w_{\sigma(n_i)})$, it follows that the subsequence $(w_{\sigma(s_j)}, w_{\sigma(s_j+1)}, \ldots, w_{\sigma(s_{j+1})})$ contains the same elements as the j-block of WQ. As there are only $n_i \leq n$ blocks, we can find an approximate sorting in linear time using a bucketing strategy.

We claim that $\sum_{j=1}^{n_i} j \cdot w_{\sigma(j)} \leq 3 \sum_{j=1}^{n_i} j \cdot w_j$. For every $1 \leq j < n_i$, it follows from the definition of j-block and the relationship between the elements of WQ' and the blocks of WQ that:

$$\sum_{k=s_j}^{s_{j+1}-1} k \cdot w_k \geq \frac{W}{2^j} \sum_{k=s_j}^{s_{j+1}-1} k = \frac{1}{2} \left(\frac{W}{2^{j-1}} \sum_{k=s_j}^{s_{j+1}-1} k \right) \geq \frac{1}{2} \sum_{k=s_j}^{s_{j+1}-1} k \cdot w_{\sigma(k)} \qquad (9)$$

In addition, because for every $k \geq s_{n_i}$ we have $w_{\sigma(k)} \leq W/2^{n_i}$, the sum $\sum_{k=s_{n_i}}^{n_i} k \cdot w_{\sigma(k)}$ can be upper bounded by $n_i^2(W/2^{n_i})$. Therefore, for $n_i > 3$ this term can be upper bounded by W. Combining with the fact that $\sum_{k=1}^{n_i} k \cdot w_k \geq W$ we have that $\sum_{k=s_{n_i}}^{n_i} k \cdot w_{\sigma(k)} \leq \sum_{k=1}^{n_i} k \cdot w_k$.

Using the previous inequality together with inequality (9), we have:

$$\begin{aligned} \sum_{k=1}^{n_i} k \cdot w_{\sigma(k)} &= \sum_{j=1}^{n_i-1} \sum_{k=s_j}^{s_{j+1}-1} k \cdot w_{\sigma(k)} + \sum_{k=s_{n_i}}^{n_i} k \cdot w_{\sigma(j)} \\ &\leq 2 \sum_{j=1}^{n_i-1} \sum_{k=s_j}^{s_{j+1}-1} k \cdot w_k + \sum_{k=1}^{n_i} k \cdot w_k \leq 3 \sum_{k=1}^{n_i} k \cdot w_k \end{aligned}$$

Now suppose that instead of using an exact sorting our algorithm uses an *approximate sorting* of the trees $\{T_{q_i}^j\}$, by means of permutations σ_{q_i}. It is easy to see that the only impact this modification has to the upper bound of the algorithm occurs in the second term of the right-hand side of inequality (2), with $\sum_j j \cdot w(T_{q_i}^j)$ being replaced by $\sum_j j \cdot w(T_{q_i}^{\sigma_{q_i}(j)})$. Then the previous

[1]In order to avoid a heavier notation, we assume that each j-block is nonempty.

argument implies that this *approximate sorting* introduces a multiplicative factor of three in second term of (2) and it is straight forward to check this does not alter the guarantee of the algorithm.

Theorem 6 *There is a linear time algorithm which provides a constant factor approximation for the problem of binary searching in trees.*