# Distribution algorithms for the all-nearest-foreign-neighbors problem in arbitrary $L^t$-metrics

Thorsten Graf, Klaus Hinrichs

Institut für numerische und instrumentelle Mathematik -INFORMATIK-
Westfälische Wilhelms-Universität Münster
Einsteinstr. 62 D-48149 Münster e-mail: graf@math.uni-muenster.de khh@math.uni-muenster.de

EXTENDED ABSTRACT

**Abstract.** Given a dynamic set $S$ of $n$ colored points in the plane. In the *all-nearest-foreign-neighbors* problem (ANFN problem) one has to find for each point in $S$ a nearest neighbor with a different color. We describe a *distribution* technique which reduces the ANFN problem to a number of bichromatic ANFN problems which can be solved in a total worst case time of $O(n \log n)$. In this paper we present two simple algorithms based on the distribution technique which solve the ANFN problem with respect to an arbitrary $L^t$-metric in optimal $O(n \log n)$ time and $O(n)$ space.

## 1 Solving the ANFN problem by distribution

In this paper we consider the *all-nearest-foreign-neighbors* (ANFN) problem which is defined as follows:

> Given a finite set $S$ of points in the plane $\mathbb{R}^2$, $|S| = n$, $S = \cup_{i=1}^{N} S_i$ with $S_i \cap S_j = \emptyset$ for $i, j \in \{1, \ldots, N\}$, $i \neq j$, and a fixed $L^t$-metric $d_t$ $(1 \leq t \leq \infty)$. For each $i \in \{1, \ldots, M\}$ and each $p \in S_i$ determine a point $q \in S \setminus S_i$ with $d_t(p, q) = \min\{d_t(p, r) : r \in S \setminus S_i\}$.

We assign each of the sets $S_i$ a unique color and reformulate the problem as follows: Determine for each point $p \in S$ a nearest neighbor in $S$ having a color different from $p$'s color $c(p)$. $\Omega(n \log n)$ is a lower bound for the ANFN problem in the algebraic decision tree model of computation ([1, 2]).

In this section we present a *distribution mechanism* which can be used to solve the ANFN problem with respect to an arbitrary Minkowski-metric. The method reduces the ANFN problem for the given configuration to a number of bichromatic ANFN problems such that the total time needed to solve these problems does not exceed the optimal time bound $O(n \log n)$. Our algorithms for the ANFN problem are based on this distribution technique. [1] describe an algorithm for the Euclidean ANFN problem which uses a similar distribution mechanism.

Each set $S_i$ is assigned an initially empty *candidate set* $C_i$. The points in $S$ are *distributed* among the candidate sets, i.e. each point is assigned to some of the candidate sets such that the total number of points contained in the candidate sets is linear, i.e. $\sum_{i=1}^{m} |C_i| \in O(n)$. After the distribution process a nearest foreign neighbor of a point is found in the candidate set of its color. Hence the ANFN problem for $S$ has been reduced to the problem of finding nearest neighbors for the points in $S_i$ among the points in $C_i$. This can be interpreted as a bichromatic ANFN problem for the set $S_i \cup C_i$ where nearest foreign neighbors have to be computed only for the points in $S_i$.
The bichromatic ANFN problem for a set $S_i$ and its candidate set $C_i$ can be solved by the algorithm [3] which also works for arbitrary $L^t$-metrics in

$$\sum_{i=1}^{m} O((|C_i| + |S_i|) \log |C_i| + |S_i| \log |S_i|) = O(n \log n)$$

time and linear space ([3]). In this paper we present two algorithms for distributing the points among the candidate sets in $O(n \log n)$ time and linear space. The analysis of our algorithms is fairly straightforward, and we therefore omit details.

## 2 The first distribution algorithm

In a preprocessing step we use the algorithm [2] to solve the $L^\infty$-ANFN problem for $S$. This preprocessing step requires $O(n \log n)$ time and returns a $L^\infty$-nearest foreign neighbor $\tilde{p}$ of each point $p \in S$.

The algorithm performs four sweeps: a left-to-right, a right-to-left, a top-to-bottom, and a bottom-to-top sweep. We will show that if a nearest foreign neighbor $q$ of $p$ is contained in $QL(p) := \{r | r.x \leq p.x \wedge |p.y - r.y| \leq |p.x - r.x|\}$ then $q$ will be inserted into $C_{c(p)}$ during the left-to-right sweep. Together with similar arguments for the three remaining sweeps and the corresponding quadrants this ensures the correctness of the algorithm. It therefore suffices to describe the left-to-right sweep.

Fix a point $p \in S$. Let $p_1, \ldots, p_8$ be the points which we obtain by mirroring the point $\tilde{p}$ cyclically along the diagonals (with slopes $\pm 1$) and the horizontal and vertical lines through $p$ (see Figure 1). We will use these points to select a set of

points to be considered when processing the insertion event of $p$. In the left-to-right sweep we only use the points $p_1$ and $p_2$, the other points $p_3, \ldots, p_8$ are used in the other sweeps analogously.

An *insertion* event for $p \in S$ has to be processed when the sweep-line reaches $p$. Insertion events which coincide in space can be processed in arbitrary order. To process an insertion event we do the following: First insert $p$ into the sweep-line structure $SLS$. Then we have to find the nearest foreign neighbor candidates of $p$ lying in $QL(p)$.

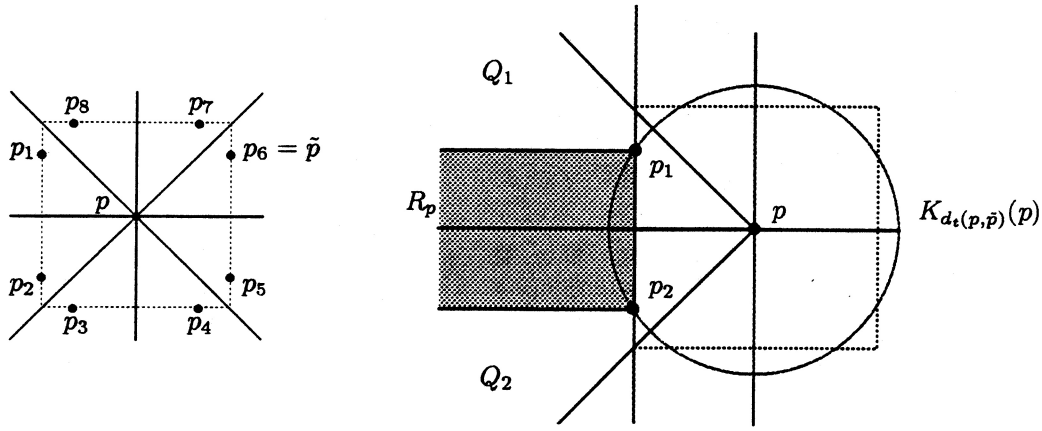


**Figure 1.** Points $p_1, \ldots, p_8$ and region $R_p \subset QL(p)$ which may contain nearest foreign neigbors of $p$

Lemma 2.1 restricts the area where to search for nearest foreign neighbors of $p$ with a distance to $p$ strictly smaller than $d_t(p, \tilde{p})$:

**Lemma 2.1.** *Nearest foreign neighbors of $p$ in $QL(p)$ with $d_t(p,q) < d_t(p,\tilde{p})$ are contained in the region $R_p := \{(x,y) | x \leq p_1.x, p_2.y < y < p_1.y\}$.*

**Proof:** There is no nearest foreign neighbor of $p$ in $QL(p)$ with a $x$-coordinate greater than $p_1.x = p_2.x$. Since the $L^t$-circle $K_{d_t(p,\tilde{p})}(p)$ with center $p$ and radius $d_t(p,\tilde{p})$ does not intersect the north-west quadrant $Q_1 := \{(x,y) | x < p_1.x, y \geq p_1.y\}$ of $p_1$ and the south-west quadrant $Q_2 := \{(x,y) | x < p_2.x, y \leq p_2.y\}$ of $p_2$ this completes the proof. Figure 1 gives an illustration for the Euclidean metric. □

Denote by $QL_a(p) := \{r \in QL(p) | r.y \geq p.y\}$ and $QL_b(p) := \{r \in QL(p) | r.y \leq p.y\}$. $SLS$ consists of two components $SLS_a$ and $SLS_b$ each of which holds a set of points and is implemented by a *quadrant priority search tree* ([2]) choosing appropriate keys. When a point $p$ is encountered by the sweep-line $p$ is inserted into both $SLS_a$ and $SLS_b$. Then we remove from $SLS_a$ the points contained in $R_p \cap QL_a(p)$ and from $SLS_b$ the points in $R_p \cap SLS_b$ and add all of these points which have a different color than $c(p)$ to the candidate set $C_{c(p)}$.

## 2.1 Correctness

As above we restrict ourselves to the left-to-right sweep and the case that a nearest foreign neighbor $q$ of $p$ is contained in $QL(p)$. Before we prove that $q$ will become a member of $C_{c(p)}$ we give two lemmas:

**Lemma 2.2.** *Let $r, q \in \mathbb{R}^2$ such that $q \in QL_a(r)$ and $r.y \neq q.y$. Then the half-line $h$ starting at point $(r.x, r.y + r.x - q.x)$ with slope $+1$ is part of the $L^\infty$-bisector $B_\infty(q,r)$ for $x \geq r.x$. For $1 \leq t < \infty$ the $L^t$-bisector $B_t(q,r)$ extends for $x \geq r.x$ above $h$.*

**Proof:** Since $L^t$-bisectors are invariant under translations we may assume that $q.x = 0$ and $r.y = 0$.

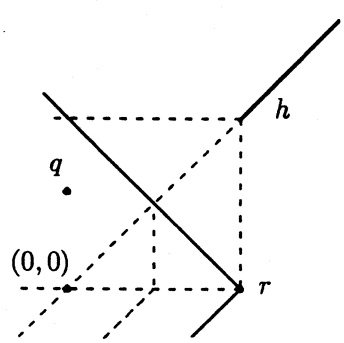


**Figure 2:** Extension of $L^t$-bisectors for $x \geq r.x$

For points $u = (\eta, \eta)$ on the diagonal $y(x) = x$ with $\eta \geq r.x$ we have $d_\infty(u, q) = \eta = d_\infty(u, r)$ which proves the first part of the lemma. Fix an arbitrary $1 \leq t < \infty$. The bisector $B_t(r, q)$ is given in implicit form by

$$B_t(r,q): \quad B(x,y) = |x - q.x|^t + |y - q.y|^t - |x - r.x|^t - |y - r.y|^t = 0$$

Since $q.y \leq r.x$ an easy computation shows that $B(\eta, \eta) \geq 0$ for all $\eta \geq r.x$ which implies that for $x \geq r.x$ the halfline $h$ is on the same side of $B_t(r, q)$ as $r$. Hence $B_t(r, q)$ extends above $h$. □

**Lemma 2.3.** *Let $p, q, r \in \mathbb{R}^2$ such that $q \in QL_a(p) \cap QL_a(r)$ and $r.x \leq p.x$. Then for all $1 \leq t \leq \infty$ we have $d_t(p, r) \leq d_t(p, q)$.*

**Proof:** Again we may assume that $q.x = 0$ and $r.y = 0$. If $q.y = r.y$ then the vertical line crossing point $(\frac{1}{2}(r.x + q.x), q.y)$ is part of the bisector $B_t(q, r)$ ([4]) which implies that $d_t(p, r) \leq d_t(p, q)$. We may therefore assume in the following that $q.y > r.y$. The $L^t$-bisector is a function which is non-decreasing ([4]) since $|q.x - r.x| \leq |q.y - r.y|$.

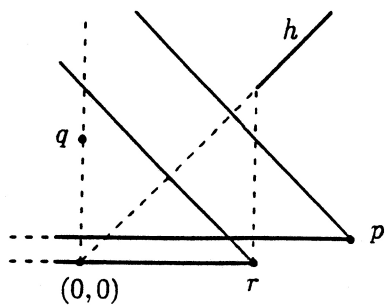


**Figure 3:** Point $q$ cannot be nearest neighbor of $p$

By Lemma 2.2 the part of the bisector $B_t(r, q)$ to the right of $r.x$ extends above the half-line $h$ which for $x \geq r.x$ falls onto the diagonal $y(x) = x$. Since $q \in QL_a(p)$ and $q.y > r.y$ this implies that for $x \geq r.x$ the bisector $B_t(r, q)$ extends above $q.y$ and strictly above $q.y$ for $x > r.x$. By the condition $q \in QL_a(p)$ we have $q.y \geq p.y$. Hence $p$ is on the same side of $B_t(q, r)$ as $r$ which completes the proof. □

Now fix an arbitrary point $p \in S$. W.l.o.g. we assume that a nearest foreign neighbor $q$ of $p$ is contained in $QL_a(p)$. Let $q$ be chosen such that $|p.x - q.x|$ is minimal. If $d_t(p, q) = d_t(p, \tilde{p})$ a nearest foreign neighbor of $p$ has already been found during the preprocessing step and nothing remains to be shown. We therefore assume that $d_t(p, q) < d_t(p, \tilde{p})$.
If $q$ has not yet been removed from $SLS_a$ when the insertion event for $p$ is processed then $q$ is inserted into $C_{c(p)}$ by construction, if $q$ has already been removed from $SLS_a$ then Theorem 2.4 guarantees that $q$ has been inserted into the candidate set $C_{c(p)}$.

**Theorem 2.4.** *Let $q$ be a nearest foreign neighbor of $p$ in $QL(p)$ such that $|p.x - q.x|$ is minimal. If $q$ has been removed from $SLS_a$ before the insertion event of $p$ is processed then $q$ has already been inserted into $C_{c(p)}$.*

**Proof:** The assumption that $q$ has already been removed from $SLS_a$ implies that there exists a point $r \in S$ which has been processed before $p$, i.e. for which $r.x \leq p.x$, such that $q \in QL_a(r)$ and $q$ has been removed from $SLS_a$ when processing $r$. If $c(p) = c(r)$ then $q$ has been inserted into $C_{c(p)}$. We may therefore assume that $c(p) \neq c(r)$.
The points $p, q, r$ satisfy the conditions of Lemma 2.3 which implies that $d_t(p, r) \leq d_t(p, q)$. If $p.x = r.x$ then $d_t(p, r) = d_\infty(p, r) \geq d_t(p, \tilde{p})$ and $d_t(p, q) < d_\infty(p, \tilde{p})$ by assumption which implies $d_t(p, r) > d_t(p, q)$. This leads to a contradiction since either $|p.y - q.y| \geq |p.y - r.y|$ if $r.y > p.y$ or $|p.x - q.x| \geq |p.y - r.y|$ if $r.y < p.y$.
Hence we may assume $p.x > r.x$. In this case Lemma 2.3 shows that $d_t(p, r) \leq d_t(p, q)$ which is a contradiction to the choice of $q$. This completes the proof. □

## 3 The second distribution algorithm

The sweep algorithm in section 2 determines for each point $p$ the candidates among which we have to search for the nearest foreign neighbors of $p$. For the algorithm presented in this section we take a different approach: Determine for each point $p$ the candidate points which may have $p$ as nearest foreign neighbor.
This algorithm also performs four sweeps: a left-to-right, a right-to-left, a top-to-bottom, and a bottom-to-top sweep. During a sweep we maintain for each $p \in S$ the minimal distance $\delta(p)$ between $p$ and a nearest foreign neighbor found so far. We initialize $\delta(p)$ with $\infty$.
We will show that if there exists a nearest foreign neighbor $p$ of $q \in QL(p)$ then after the left-to-right sweep either $\delta(q) = d_t(p, q)$ or $p$ has been inserted into the candidate set $C_{c(q)}$. Together with similar arguments for the three remaining sweeps

and the corresponding quadrants this ensures the correctness of the algorithm. It therefore suffices to describe the left-to-right sweep.

Among the points which have already been encountered by the sweep-line a point $p$ is called *active* if $p.x + \delta(p)$ is to the right of the sweep-line, otherwise the point $p$ is called *deactivated*. The active points are exactly those of the points seen so far which may have a nearest foreign neighbor among the points to the right of the sweep-line. During the sweep we maintain the following sweep-invariant:

The $y$-table stores the active points w.r.t. their $y$-coordinate and their color.

Note that the $y$-table may contain more than one point with the same $y$-coordinate and the same color. Therefore the $y$-table stores the active points lexicographically.

There are two types of events: *insertions* and *deletions*: An insertion event for $p \in S$ has to be processed when the sweep-line reaches $p$. Insertion events which coincide in space can be processed in an arbitrary order. A deletion event for a point $p$ contained in the $y$-table has to be processed when $p.x + \delta(x)$ is at or to the left of the sweep-line. This may happen either if the sweep-line proceeds to the right or the $\delta$-value of $p$ becomes smaller.

Let the operation $succ - col(p)$ ($pred - col(p)$) return the successor (resp. the predecessor) of $p$ in the $y$-table with a color different from $c(p)$. These operations can be performed in $O(\log n)$ time by implementing the $y$-table as a *colored balanced binary search tree* (see section 4).

A deletion event of a point $p$ is processed by removing $p$ from the $y$-table. An insertion event of $p$ is processed by first inserting $p$ into the $y$-table. Then we search for those points $q$ which can have $p$ as nearest foreign neighbor by determining the following two points

$$r := col - succ(p) \qquad s := col - succ(r)$$

We compute intermediate distances of the points $p$, $r$ and $s$ and update their $\delta$-values. We maintain the sweep invariant by removing those points of $\{r, s\}$ from the $y$-table that are no longer active, i.e. we process deactivation events if necessary. We continue this process until either both points $r$ and $s$ remain active or at least one of them is the *nil*-point. After the process has stopped we insert the point $p$ into the set $C_{c(r)}$ (resp. $C_{c(s)}$) if the points $r$ (resp. $s$) is non-*nil*. Note that by the definition of $col - succ()$ we have $c(r) \neq c(s)$. The predecessor points of $p$ in the $y$-table are processed analogously. The sweep is complete after all insertion events have been processed.

## 3.1 Correctness

We have to show that the algorithm computes for all points a nearest foreign neighbor directly, or the distribution process guarantees that we find a nearest foreign neighbor when solving the bichromatic ANFN problems for the sets $S_i \cup C_i$. W.l.o.g. we restrict ourselves to the left-to-right sweep. The correctness follows from Theorem 3.1:

**Theorem 3.1.** *Let $q$ be a point for which $p$ is a nearest foreign neighbor such that $q \in QL(p)$. After processing the insertion event of $p$ either we have $\delta(q) = d_t(p, q)$ or $p$ has been inserted into the candidate set $C_{c(q)}$.*

**Proof:** W.l.o.g. we may assume that $q.y \geq p.y$. Consider the moment immediately after inserting the new point $p$ into the $y$-table. If $\delta(q) = d_t(p, q)$ then a nearest foreign neighbor of $q$ has already been found and nothing remains to be shown. We therefore assume that $\delta(q) > d_t(p, q)$. This implies $q.x + \delta(q) > q.x + d_t(p, q) \geq p.x$ since $q \in QL(p)$ which shows that $q$ is still active and therefore contained in the $y$-table.

If at any time during the examination of the successor points of $p$ either of the points $r$ or $s$ equals $q$ nothing remains to be shown since then $p$ has been found as a nearest foreign neighbor of $q$ and $\delta(q) = d_t(p, q)$. We therefore assume that this never happens during the process and that after the process has finished neither $r$ nor $s$ is the point $q$. Of course $r$ is not the *nil*-point and is below $q$, i.e. $r.y \leq q.y$. Therefore $p$ is inserted into the set $C_{c(r)}$. If $c(r) = c(q)$ this completes the proof. We therefore assume $c(r) \neq c(q)$. This implies that $s$ cannot be the *nil*-point and $s$ is below $q$, i.e. $s.y \leq q.y$. Again, if $c(s) = c(q)$ then the point $p$ is inserted into the set $C_{c(s)}$ and the theorem follows. Hence we additionally assume that $c(s) \neq c(q)$.

This leads to a contradiction: Since $p$ is a nearest foreign neighbor of $q$ the rectangle

$$R := \{(x, y) : |x - q.x| < |q.x - p.x| \wedge p.y < y < q.y\}$$

which is part of the $L^t$-circle with center $q$ and radius $d_t(p, q)$ cannot contain points of $S$ with a color different from $c(q)$.

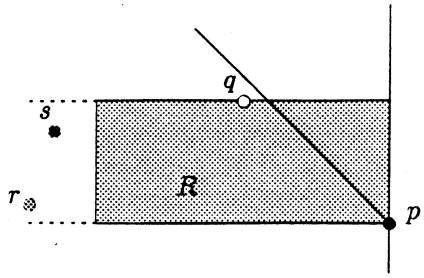


**Figure 4:** If $c(r) \neq c(q) \neq c(s)$ then $r$ and $s$ are outside of $R$

This shows that $r$ and $s$ are outside the rectangle $R$. But then an easy computation shows that

$$\min\{r.x, s.x\} + d_t(r,s) \leq \min\{r.x, s.x\} + d_1(r,s) \leq \max\{r.x, s.x\} + |p.y - q.y| \leq p.x$$

which implies that at least one of the points $r$ or $s$ has already been deactivated. This is a contradiciton since $r$ and $s$ are assumed to be active which completes the proof. □

## 4 Coloring a binary search tree

In section 3 we use a *colored balanced binary search tree* to implement the $y$-table in order to perform the operations *col-pred* and *col-succ* in logarithmic time. In this section we extend any arbitrary (not necessarily balanced) binary search tree structure to the *colored binary search tree* (CBST).

For a node $k$ of a binary search tree $T$ we denote by $T(k)$ the subtree of $T$ rooted by $k$ and by $LST(k)$ and $RST(k)$ the left respectively right subtree of $k$. The key-values of all nodes in $LST(k)$ are equal to or smaller than the key-value of $k$ which itself is smaller than any key-value of a node in $RST(k)$.

The CBST which is based on a binary search tree stores colored elements with respect to their key-value and supports the following extended query operations: When searching for the predecessor (respectively successor) of an element the CBST allows to restrict the search to those elements which do not have a certain *forbidden* color; these elements are called *admissible*. The operations supported by the CBST are the following:

1. *insert*$(e)$: Insert the element $e$ into the CBST.
2. *delete*$(e)$: Delete the element $e$ from the CBST.
3. $ad - pred(e,c)$: Return the admissible predecessor of $e$ in the CBST where $c$ is the forbidden color.
4. $ad - succ(e,c)$: Return the admissible successor of $e$ in the CBST where $c$ is the forbidden color.

The operations *col-pred* and *col-succ* in section 3 can be realized by choosing the query element's color as forbidden color in the query operations *ad-pred* and *ad-succ*, respectively. In the following we extend the domain of the color function $c()$ to the nodes of the CBST, i.e. each node of the CBST inherits the color from the element it stores.

We call a subtree $T$ of the CBST *homogeneous*, if all elements stored in $T$ have the same color. In each node $k$ we store:

$Hom(k)$ = true iff $T(k)$ is homogeneous with color $c(k)$.

The $Hom()$-information can be maintained in optimal $\Theta(h(n))$ time ($h(n)$ denotes the height of the underlying binary search tree) during an insertion operation and a bottom-up walk after a deletion operation. In the special case that all key values are known in advance we can use a *skeleton tree* to avoid rebalancing and achieve an optimal update time of $O(\log n)$: Build a complete binary search tree for the key-values. This is the *skeleton* which is filled dynamically during the sequence of insertions and deletions. This is the case in our application to the ANFN problem presented in section 3. For details about skeleton structures see e.g. [5]. However, it is easy to see that the $Hom()$-information can also be maintained in balanced trees during rebalancing operations in optimal time.

It remains to show how to perform the operations $ad - pred(e,c)$ and $ad - succ(e,c)$ for an element $e$ and a forbidden color $c$. These two operations do not differ in principle and we therefore restrict ourselves to the description of the operation $ad - pred(e,c)$:

Call an element stored in the CBST *proper* (with respect to $e$) if its key-value is smaller than the key-value of $e$. Now $ad - pred(e,c)$ asks for the admissible proper element with largest key-value, i.e. the admissible predecessor of $e$. First perform a binary search for the node $k_s$ which stores the element $e$ by walking a path $K := \{r = k_1, \ldots, k_s\}$ from the root $r$ of the CBST to the node $k_s$. During the walk consider all elements stored along $K$. It is easy to verify that if the path branches off to the left in $k_i \in K$ then the nodes in $RST(k_i)$ need not be considered. Furthermore it is easy to see that the admissible predecessor is either stored in a node $k_\nu$ of path $K$ or in $LST(k_i)$ of a node $k_i \in K$ such that $k_{i+1} \in RST(k_i)$

and $LST(k_i)$ is not homogeneous w.r.t. the forbidden color $c$. Hence we consider these subtrees bottom-up stopping as soon as the first admissible proper element has been found. Finding the admissible predecessor in a subtree $T(k)$ can be done as follows: If $RST(k)$ is not empty and is not homogeneous w.r.t. the color $c$ then $RST(k)$ contains at least one admissible proper element and we continue the search in the right son node of $k$. If either $RST(k)$ is empty or is homogeneous w.r.t. the color $c$ then we proceed as follows: If the color of the element stored in $k$ is $c$ and $LST(k)$ is not homogeneous w.r.t. color $c$ then we continue in $LST(k)$.
The process stops if we drop out of the tree (i.e. after processing a leaf node), if we cannot continue in $RST(k)$ and $k$ stores an admissible element, or if we cannot choose a subtree which is not homogeneous w.r.t. the color $c$.

A straightforward analysis shows that coloring a binary search tree does not increase the asymptotic time complexities of the operations of the underlying binary search tree.

## References


1 A.Aggarwal, H.Edelsbrunner, P.Raghavan and P.Tiwari: Optimal time bounds for some proximity problems in the plane. *Information Processing Letters* 42, 55-60, (1992).

2 Th. Graf, K. Hinrichs: Algorithms on colored point sets, *Proceedings of the 5th Canadian Conference on Computational Geometry CCCG '93*, (1993).

3 K.Hinrichs, J.Nievergelt, P.Schorn: An all-round sweep algorithm for 2-dimensional nearest-neighbor problems, *Acta Informatica*, 29(4), 383-394 (1992).

4 D. T. Lee: Two-Dimensional Voronoi Diagrams in the $L^p$-metric, *Journal of the ACM* **27**(4), 604-618 (1980).

5 Th. Ottmann and P. Widmayer: *Algorithmen und Datenstrukturen*, Reihe Informatik, Band 70, BI Wissenschaftsverlag, Mannheim, 1993.