Color-Distance Oracles and Snippets

Tsvi Kopelowitz[*1] **and Robert Krauthgamer**[†2]

1 **University of Michigan, Ann Arbor, Michigan, USA**
kopelot@gmail.com
2 **Weizmann Institute of Science, Rehovot, Israel**
robert.krauthgamer@weizmann.ac.il

Abstract

In the snippets problem we are interested in preprocessing a text T so that given two pattern queries P_1 and P_2, one can quickly locate the occurrences of the patterns in T that are the closest to each other. A closely related problem is that of constructing a color-distance oracle, where the goal is to preprocess a set of points from some metric space, in which every point is associated with a set of colors, so that given two colors one can quickly locate two points associated with those colors, that are as close as possible to each other.

We introduce efficient data structures for both color-distance oracles and the snippets problem. Moreover, we prove conditional lower bounds for these problems from both the 3SUM conjecture and the Combinatorial Boolean Matrix Multiplication conjecture.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Snippets, Text Indexing, Distance Oracles, Near Neighbor Search

Digital Object Identifier 10.4230/LIPIcs.CPM.2016.24

1 Introduction

We introduce and study the following problem: preprocess a text T so that given two pattern queries, P_1 and P_2, one can quickly locate the occurrences of the two patterns in T that are closest to each other, or report the distance between these occurrences. This natural task arises in many common indexing applications, for example, when searching a corpus of documents for two query keywords, the relevance of a document may be measured by the two keywords' proximity inside the document. Web search engines often use this notion of relevance by providing with each result a *snippet* — a subtext from the corresponding webpage in which the two keywords appear close to each other, which is very useful to assess the relevance of that result. This problem, which we call the *snippets* problem, turns out to be a special case of a more general problem, which we define next, and deals with colored points in a metric space.

Colored Points.

Let M be a metric space with distance function $d(\cdot,\cdot)$. Each point $p \in M$ may have an associated color $c_p \in [\ell]$, in which case we say that the point is *colored*. Let $S \subset M$ be a set of N points. We call S a *colored set* if every point $p \in S$ is colored. For a color $c \in [\ell]$, let $P(c)$ denote the set of points in S which have color c. The distance between a point $p \in M$ and a color $c \in [l]$ is defined as $d(p,c) := \min\{d(p,q) : \ q \in P(c)\}$. The distance between

* Work supported in part by NSF grants CCF-1217338, CNS-1318294, and CCF-1514383.
† Work supported in part by an Israel Science Foundation grant #897/13.

two colors, called the *color-distance*, of $c, c' \in [\ell]$ is defined as $d(c, c') := \min\{d(q, q') : q \in P(c), q' \in P(c')\}$.

One application for computing the distance of two colors arises in navigational tools. For example, consider a user who is interested in visiting both a postoffice and a pharmacy. One can color, in advance, all of the pharmacies with one color and all of the postoffices with another color. The distance between the two colors corresponds to the closest pair of a postoffice and a pharmacy. This leads to the following problem.

▶ Problem 1.1 (Color-Distance Oracle). The color-distance oracle problem asks to preprocess a colored set S, so that given a query of two colors $c, c' \in [\ell]$, one can quickly report $d(c, c')$.

Multi-colored points.

A natural generalization of color-distance oracles is to let each point have several colors. For example, a single location in a map could be both a postoffice and a pharmacy. A point $p \in M$ is said to be *multi-colored* if p has an associated nonempty set of colors $C(p) \subseteq [\ell]$. A set of points $S \subset M$ is a *multi-colored set* if each point $p \in S$ is multi-colored.

▶ Problem 1.2 (Multi-Color-Distance Oracle). The multi-color-distance oracle problem asks to preprocess a multi-colored set S, so that given a query of two colors $c, c' \in [\ell]$, one can quickly report $d(c, c')$.

One straightforward way for solving the multi-color-distance oracle problem is to create $|C(p)|$ copies of each point, one for each color, and apply a solution for the color-distance oracle problem (such as Theorem 2). The size of the instance of the newly created instance is $\sum_{p \in S} |C(p)|$, which could be much larger than $N = |S|$. Notice that this quantity is actually the size of the input for the multi-color-distance oracle problem, since generally speaking, each point may need to have its colors listed explicitly. Nevertheless, there are interesting cases in which the list of colors for each point need not be given explicitly. One such example is in the snippets problem, which falls under the notion of a color hierarchy, described next.

Color hierarchies.

We say that a multi-colored set S admits a *color hierarchy* if for every two colors $c, c' \in [l]$, either one of the sets $P(c)$ and $P(c')$ contains the other, or the two sets are disjoint (a formal terminology is that $\{P(c)\}_{c \in [l]}$ is a laminar family). It is easy to see that a color hierarchy can be represented by a rooted forest (i.e., each tree has a root) T_S of size $O(\ell)$, where each color c is associated with a vertex u_c in this forest, such that the descendants of u_c are exactly all the vertices $u_{c'}$ whose color c' satisfies $P(c') \subseteq P(c)$. We convert the forest T_S to a rooted tree by adding a dummy root vertex and making it the parent of all of the roots of the trees in the forest. With the aid of T_S, a multi-colored set S that admits a color hierarchy can be represented using only $O(N + \ell)$ machine words, because it suffices to describe the tree and associate with each point just one color (the color with the lowest corresponding vertex in T_S); the other colors of this point are implicit from T_S (the colors on the path to the root of T_S). This leads us to the following problem.

▶ Problem 1.3 (Multi-Color-distance Oracle with a Color Hierarchy). The multi-color-distance oracles with a color hierarchy problem asks to preprocess a multi-colored set S that admits a color hierarchy, so that given a query of two colors $c, c' \in [\ell]$, one can quickly report $d(c, c')$.

1.1 Our Results

Our main result is a data structure for the snippets problem, summarized as follows.

▶ **Theorem 1.** *For every fixed $\epsilon > 0$, there is a data structure for the snippets problem with preprocessing time $O(N^{1.5}\log^{\epsilon} N)$, query time $O(|P_1|+|P_2|+\sqrt{N}\log^{\epsilon} N))$, and space usage $O(N)$ words.*

To prove Theorem 1 we solve the more general problems of colored and multi-color-distance oracles. These data structures use (in a black-box manner) an algorithm (data structure) for nearest neighbor search.

Nearest Neighbor Search (NNS).

In the Nearest Neighbor Search (NNS) problem, the goal is to preprocess a set of points $S \subset M$ (recall M is a metric space), so that given a point $p \in M$ one can quickly report $\text{argmin}_{q\in S}\{d(p,q)\}$. Given an NNS algorithm for $k = |S|$ points, we denote its preprocessing time by $t_{\text{NNS}}(k)$, its query time by $q_{\text{NNS}}(k)$, and its space usage by $s_{\text{NNS}}(k)$. Once the nearest neighbor is found, one can evaluate the distance between p and its nearest neighbor by invoking the function d (we assume such evaluation takes $O(1)$ time, for simplicity), thereby obtaining $d(p,S)$.

With the aid of an NNS data structure for point sets in a metric M, we prove the following theorem in Section 2.

▶ **Theorem 2** (Color-distance Oracle)**.** *Assume there is a data structure that supports NNS queries on a set of k points from M using preprocessing time $t_{\text{NNS}}(k)$, query time $q_{\text{NNS}}(k)$, and space usage $s_{\text{NNS}}(k)$ words. Then for every $0 < \tau \le N$, there exists a color-distance oracle for N-point sets in M, that has preprocessing time $O(t_{\text{NNS}}(N) + N\cdot\tau\cdot q_{\text{NNS}}(N))$, query time $O(\tau\cdot q_{\text{NNS}}(N))$, and space usage $O(s_{\text{NNS}}(N) + (\frac{N}{\tau})^2)$ words.*

Our solution for the multi-color-distance oracles with a color hierarchy problem is based on the notion of range NNS, which is defined as follows.

▶ Problem 1.4 (Range NNS). In the range NNS problem the goal is to preprocess an array A of N points from a metric M, so that given two indices $1 \le i \le j \le N$ and a point $p \in M$, one can quickly find the NNS of p in the set $\{A[i], A[i+1], \ldots, A[j]\}$.

We prove the following theorem in Section 3.

▶ **Theorem 3** (Multi-Color-distance Oracle with a Color Hierarchy)**.** *Assume there is a range NNS algorithm for k-point sets in a metric M that uses preprocessing time $t_{\text{RNNS}}(k)$, query time $q_{\text{RNNS}}(k)$, and space usage $s_{\text{RNNS}}(k)$ words. Then for every $0 < \tau \le N$, there exists a multi-color-distance oracle for N-point sets in M that admit a color hierarchy, (specifically, the multi-coloring of the input S is given implicitly via a tree T_S), that has preprocessing time $O(t_{\text{RNNS}}(N) + \frac{N^2}{\tau}\cdot(q_{\text{RNNS}}(N) + \log\log\log\frac{N}{\tau}))$, query time $O(\tau\cdot q_{\text{RNNS}}(N))$, and space usage $O(s_{\text{RNNS}}(N) + (\frac{N}{\tau})^2)$ words.*

Conditional Lower Bounds.

Solving the multi-color-distance oracle problem with poly-logarithmic query time and non-trivial preprocessing time seems to be extremely difficult, leading to the question of finding a polynomial time lower bound. Polynomial (unconditional) lower bounds for data structure problems are considered beyond the reach of current techniques. Thus, it is common to

prove conditional lower bounds (CLBs) based on the *conjectured* hardness of some "basic" problem. One of the most popular conjectures for CLBs is that the 3SUM problem (given n integers determine if any three sum to zero) cannot be solved in truly subquadratic time, where truly subquadratic time is $O(n^{2-\Omega(1)})$ time. This conjecture is reasonable even if the algorithm is allowed to use randomization, see e.g. [30, 1, 23, 15].

Another popular conjecture is the *combinatorial Boolean matrix multiplication* (BMM) conjecture. In the BMM problem we are given two $n \times n$ Boolean matrices A and B and the task is to compute the Boolean product of the two matrices. The combinatorial BMM conjecture states that combinatorial algorithms for computing this Boolean product require runtime $\Omega(n^{3-o(1)})$, see [1].

In Section 5 we prove CLBs for the color-distance oracle problem based on the 3SUM and combinatorial BMM conjectures. Moreover, these CLBs hold also for approximate versions of the color-distance oracle problem, where the answer to a color-distance query between two colors c and c' is required to be between $d(c, c')$ and $\alpha \cdot d(c, c')$, where $\alpha \geq 1$ is a *stretch* parameter. The CLBs are summarized as follows.

▶ **Theorem 4.** *Assume the 3SUM conjecture holds. Then for every fixed $0 < \gamma < 1$ and fixed $\alpha \geq 1$, every data structure for the color-distance oracle problem with stretch α for points on the line, that has preprocessing time t_{CDO} and query time q_{CDO}, must satisfy*

$$t_{CDO} + N^{\frac{1+\gamma}{2-\gamma}} q_{CDO} = \Omega\left(N^{\frac{2}{2-\gamma}-o(1)}\right).$$

Notice that by taking γ arbitrarily close to 0, a linear preprocessing time implies an $\Omega(N^{0.5-o(1)})$ query time. This is line with other conditional lower bounds based on the 3SUM conjecture [23].

▶ **Theorem 5.** *Assume the combinatorial BMM conjecture holds. Then every combinatorial data structure for the color-distance oracle problem with constant stretch $\alpha \geq 1$ for points on the line, that has preprocessing time t_{CDO} and query time q_{CDO}, must satisfy*

$$t_{CDO} + N \cdot q_{CDO} = \Omega\left(N^{1.5-o(1)}\right).$$

Comparing Theorem 5 with Theorem 1 and assuming the combinatorial BMM conjecture holds, it is impossible to obtain a polynomial speedup in both the preprocessing and query time of Theorem 1 via combinatorial algorithms. However, it might be possible to obtain a polynomial speedup in one of them. We emphasize that the proofs of Theorems 4 and 5 are for the one dimensional case, and thus the conditional lower bounds apply to the special case of the snippets problem.

1.2 Related Work

Perhaps the most related problem color-distance oracles is the *(approximate) vertex-labeled distance oracles for graphs problem*, where we are interested in preprocessing a colored graph G so that given a query of a vertex q and a color c we can return $d(q, c)$ (or some approximation thereof). Hermelin, Levy, Weimann and Yuster [16] introduced this problem and provided, amongst other results, a data structure using $O(kn^{1+1/k})$ expected space with stretch $(4k-5)$ and $O(k)$ query time. In another result they showed how to reduce the space usage to $O(kN\ell^{1/k})$ at the expense of an exponential stretch $(2^k - 21)$. Chechik [8] showed how to reduce this stretch back down to $(4k - 5)$.

Two pattern document retrieval problems.

Another related body of work are document retrieval problems on two patterns. In the *Document Retrieval problem* [28] we are interested in preprocessing a collection of documents $X = \{D_1, \cdots, D_k\}$ where $N = \sum_{D \in X} |D|$, so that given a pattern P we can quickly report all of the documents that contain P. Typically, we are interested in run time that depends on the number of documents that contain P and not in the total number of occurrences of P in the entire collection of documents. In the *Two Patterns Document Retrieval problem* we are given two patterns P_1 and P_2 during query time, and wish to report all of the documents that contain both P_1 and P_2. In the *Forbidden Pattern Document Retrieval problem* [14] we are also interested in preprocessing the collection of documents but this time given a query P^+ and P^- we are interested in reporting all of the documents that contain P^+ and do not contain P^-.

All known solutions for the Two Patterns Document Retrieval problem or the Forbidden Pattern Document Retrieval problem with non trivial preprocessing use at least $\Omega(\sqrt{N})$ time per query [28, 11, 17, 18, 14]. In a recent paper, Larsen, Munro, Nielsen, and Thankachan [25] show lower bounds for these problems conditioned on the hardness of BMM. More recently some CLBs for both problems were shown from the 3SUM conjecture as well [23].

Nearest Neighbor Search.

The NNS problem has been studied intensively for many metric spaces M, due to its numerous applications. The literature on both theoretical and practical aspects is very extensive, and we provide below only a brief overview of leading theoretical approaches.

In the classical setting where M is a D-dimensional Euclidean space, the standard algorithm is to preprocess the point set by computing a Voronoi diagram, which has a fast query time. However, the Voronoi diagram requires $O(n^{\lceil D/2 \rceil})$ time and space, which is prohibitive unless D is rather small. Several algorithms are known to achieve $(1+\epsilon)$-approximate NNS in R^D (often under any ℓ_p norm) by employing various space partitions. Specifically, Arya, Mount, Netanyahu, Silverman, and Wu [5] achieve preprocessing time that is linear in the number of points k (but exponential in D), which is quite effective when the dimension D is not too large. Locality Sensitive Hashing (LSH), which was introduced by Indyk and Motwani [19] and further refined later, see e.g. [3, 4], is an alternative approach that is often preferred for high dimension D, because its performance is polynomial in D (although its query time is typically polynomial, and not logarithmic, in k).

In general metric spaces (i.e., not of the form R^D), NNS is considered a very difficult problem. But under certain "bounded growth" conditions on the data, one can obtain performance that is similar to, or even better than, the low-dimensional Euclidean case, see e.g. [10, 20, 24, 12], and the survey [9].

2 Color-distance Oracle

Proof of Theorem 2. We begin by preprocessing each set $P(c)$ with a NNS data structure. This takes a total of $O(s_{\text{NNS}}(N))$ words of space and $O(t_{\text{NNS}}(N))$ time. A color c is said to be *light* if $|P(c)| < \tau$. If color c is not light then it is *heavy*. Notice that there can be at most N/τ heavy colors. For each pair of heavy colors we precompute and store their distances in a lookup table using $O((\frac{N}{\tau})^2)$ words. The computation of the entries for this table is done directly using $O(N\tau)$ NNS queries. To answer a color-distance query on two colors c and c', if both colors are heavy then we use the lookup table for their precomputed distance.

Otherwise, assume without loss of generality that c is light. We then execute $|P(c)| < \tau$ NNS queries on $P(c')$, one for each of the points in $P(c)$, and the distance between c and c' is the smallest distance found by any of these NNS queries.

To summarize, the preprocessing time is $O(t_{\text{NNS}}(N) + N \cdot \tau \cdot q_{\text{NNS}}(N))$, the query time is $O(\tau \cdot q_{\text{NNS}}(N))$, and the space usage is $O(s_{\text{NNS}}(N) + (\frac{N}{\tau})^2)$ words. ◀

Since we may assume that $s_{\text{NNS}}(k) = \Omega(k)$ (as we need to store all of the points in S), picking $\tau = \sqrt{N}$ the preprocessing time becomes $O(t_{\text{NNS}}(N) + N^{1.5} q_{\text{NNS}}(N))$, the space usage becomes $O(s_{\text{NNS}}(N))$ and the query time is $O(\sqrt{N} \cdot q_{\text{NNS}}(N))$.

3 Multi-color-distance Oracle

Proof of Theorem 3. Assume without loss of generality that T_S is an ordinal tree (the children of each vertex are ordered). For every point $p \in S$ we create a new vertex u_p and add it to T_S as a child of the single vertex representing the color set $C(p)$. This process adds N leaves to T_S and now each leaf is associated with a unique point. With the aid of T_S we embed the set S in an array A, where the order is determined by the order in which the leaves (corresponding to points in S) are encountered during a pre-order traversal of T_S. After the construction of A, by the properties of ordered traversals on trees, each color $c \in [\ell]$ is associated with a range in A. We preprocess A using a RNNS data structure.

Next, we partition A into blocks of size τ. For each pair of blocks we precompute and store the two closest points, one from each block, together with their distances in a $\frac{N}{\tau} \times \frac{N}{\tau}$ matrix B. The entry $B[i][j]$ corresponds to the two closest points between the ith and jth blocks. It is straightforward to compute each entry in $O(\tau \cdot q_{\text{RNNS}}(N))$ time, for a total of $O(\frac{N^2}{\tau} \cdot q_{\text{RNNS}}(N))$ time to precompute the B. Next, we preprocess B using a 2D Range Minimum Query (2DRMQ) data structure [2]. Such data structures preprocess a matrix of values (in our case these are the distances that are stored in B) so that given a rectangle in the matrix, defined by its corners, we can quickly return the entry with the smallest value. The 2DRMQ data structure uses $O((\frac{N}{\tau})^2)$ space and $O((\frac{N}{\tau})^2 \log\log\log \frac{N}{\tau})$ preprocessing times, and the query time is constant.

Answering a query.

To answer a multi-color-distance oracle query between two colors c and c', let $[x_c, y_c]$ and $[x_{c'}, y_{c'}]$ be the ranges in A that are associated with c and c' respectively. Each interval can be partitioned into three parts, based on the block partitioning. The first (last) part is a suffix (prefix) of some block that starts from the left (ends at the right) endpoint of the interval. The middle part is everything else, which completely spans some consecutive blocks. Notice that the first and last part have size at most τ. For the two middle parts (one for each color) we find the two closest points by invoking the 2DRMQ data structure on B, since the two contiguous ranges define a natural rectangle in B. This covers all of the possible combinations of two points from the two middle parts and takes constant time. Now to the remaining parts. Each of the $O(\tau)$ points in the first and last parts of $[x_c, y_c]$ is queried against the entire range $[x_{c'}, y_{c'}]$ with the range NNS data structure. Similarly, each of the $O(\tau)$ points in the first and last parts of $[x_{c'}, y_{c'}]$ is queried against the entire range $[x_c, y_c]$ with the range NNS data structure. This costs $O(\tau \cdot q_{\text{RNNS}}(N))$ time. Finally, we take the minimum over all of the answers to the queries. Since the queries together cover all of the possible pairs of points, the minimum over all queries is the distance between the colors. ◀

4 Back to Snippets

A multi-color-distance oracle with a color hierarchy can be used to solve the snippets problem as follows. During the preprocessing phase we construct the suffix tree for the text T. Every internal node in the suffix tree defines a unique color. Location i in T is colored with all of the colors defined by nodes on the path from the root of the suffix tree to the leaf corresponding to the suffix at location i. Thus, the suffix tree represents the color hierarchy of the set of colors.

Given two query patterns, P_1 and P_2, we first locate their corresponding vertices in the suffix tree. This takes $O(|P_1| + |P_2|)$ time. These two nodes define the two colors that we give as input to the multi-color-distance oracle query. It is straightforward to see that the answer to this color-distance oracle query is exactly the distance of the two patterns in T.

In order to use Theorem 3 we need to specify a range NNS data structure that works in a metric defined by locations of an array. For this we can use the range predecessor data structures [13, 29, 26, 31, 27, 6, 7, 22, 21]. In these data structures the goal is to preprocess an array A of n elements from integer universe $[u]$ so that given a query $range\_pred(x, y, p)$ where $1 \leq x \leq y \leq n$ and $p \in [u]$ the data structure quickly returns $\text{argmax}_{q \in \{A[x], A[x+1], \ldots, A[y]\}}\{q < p\}$. Using, for example, the data structure of [29] the preprocessing time is $O(n \log n)$, the query time is $O(\log^{\epsilon} n)$ and the space usage is $O(n)$ words, where $\epsilon > 0$ is an arbitrarily small constant. Plugging these runtimes into Theorem 3 and setting $\tau = \sqrt{N}$ completes the proof of Theorem 1.

Notice that if the multi-color-distance oracle would return the two points that are closest, and not just their distance, then we could also report the two occurrences of the patterns that are closest to each other. Our implementations of multi-color-distance oracle do in fact allow for this information to be returned.

5 Conditional Lower Bounds

Offline SetDisjointness.

Both the 3SUM problem and the combinatorial BMM problem can be reduced to the SetDisjointness data structure problem. In this problem we wish to preprocess a family of sets F, all from universe U, with total size $N = \sum_{S \in F} |S|$ so that given a query of pointers to two sets $S, S' \in F$, one can quickly determine if $S \cap S' = \emptyset$. For a SetDisjointness data structure let t_p denote the preprocessing time and let t_q denote query time. The following theorems summarize the best known CLBs for the SetDisjointness data structure problem from the 3SUM and combinatorial BMM conjectures.

▶ **Theorem 6** ([23]). *Assume the 3SUM conjecture holds. For every fixed $0 < \gamma < 1$, any data structure for SetDisjointness has $t_p + N^{\frac{1+\gamma}{2-\gamma}} t_q = \Omega\left(N^{\frac{2}{2-\gamma} - o(1)}\right)$.*

▶ **Theorem 7** (Folklore). *Assume the combinatorial BMM conjecture holds. Any combinatorial data structure for SetDisjointness has $t_p + N \cdot t_q = \Omega\left(N^{1.5-o(1)}\right)$.*

SetDisjointness via color-distance oracles

We prove next that the SetDisjointness data structure problem can be reduced to the color-distance oracle problem. Combining this reduction with Theorems 6 and Theorem 7 we obtain CLBs for the the color-distance oracle problem from both the 3SUM conjecture and the combinatorial BMM conjecture. Moreover, this reduction also holds for approximate

versions of the color-distance oracle problem. In these versions the answer to a color-distance query between two colors c and c' can be as large as $\alpha \cdot d(c, c')$, where $\alpha \geq 1$ is a constant stretch parameter.

▶ **Theorem 8.** *If there exists a color-distance oracle with constant stretch $\alpha \geq 1$ for points on the line with t_{CDO} preprocessing time and q_{CDO} query time, then there exists a data structure for online* SetDisjointness *where $t_p = O(t_{CDO} \log k)$ and $t_q = O(q_{CDO} \log k)$.*

Proof. We reduce the SetDisjointness problem to the color-distance problem as follows. Let $F = \{S_1, \cdots, S_k\}$. For each S_i we define a unique color c_i. For an element $e \in U$ let $|e|$ denote the number of subsets containing e. Since each element in U appears in at most $O(k)$ subsets, we partition U into $\Theta(\log k)$ parts where the i^{th} part P_i contains all of the elements $e \in U$ such that $2^{i-1} < |e| \leq 2^i$. An array X_i is constructed from $P_i = \{e_1, \cdots e_{|P_i|}\}$ by assigning an interval $I_j = [f_j, \ell_j]$ in X_i to each $e_j \in P_i$ such that no two intervals overlap. Every interval I_j contains a list of all of the colors of sets in F that contain e_j. This implies that $|I_j| = |e_j| \leq 2^i$. Furthermore, for each e_j and e_{j+1} we separate I_j from I_{j+1} with a dummy color d listed $2^i + 1$ times at locations $[\ell_j + 1, f_{j+1} - 1]$. Finally, we pad each X_i so that its size is exactly N. This is always possible since $\sum_{e \in U} |e| = N$ (so the array is never of size more than N).

We now simulate a SetDisjointness query on subsets $(S_i, S_j) \in F$ by performing a color-distance query on colors c_i and c_j in each of the $\Theta(\log k)$ arrays. There exists a P_i for which the two points returned from the query are at distance strictly less than $2^i + 1$ if and only if there is an element in U that is contained in both S_i and S_j. Thus, using $O(\log k)$ color-distance queries we solve the SetDisjointness query.

Finally, notice that the reduction also holds for the approximate case, as for any constant α the reduction can overcome the α approximation by separating intervals using $2^i\alpha + 1$ instances of the dummy color d. ◀

Acknowledgments. We thank Sharma Thankachan for suggesting to consider range predecessor data structures, thereby significantly simplifying our earlier, more complicated, solution.

References

1 A. Abboud and V. Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 434–443, 2014. doi:10.1109/FOCS.2014.53.

2 A. Amir, J. Fischer, and M. Lewenstein. Two-dimensional range minimum queries. In *Combinatorial Pattern Matching, 18th Annual Symposium, CPM*, pages 286–294, 2007. doi:10.1007/978-3-540-73437-6_29.

3 A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *47th Annual IEEE Symposium on Foundations of Computer Science*, pages 459–468. IEEE, 2006. doi:10.1109/FOCS.2006.49.

4 A. Andoni and I. Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *47th Annual ACM on Symposium on Theory of Computing*, STOC'15, pages 793–801. ACM, 2015. doi:10.1145/2746539.2746553.

5 S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. *J. ACM*, 45(6):891–923, 1998. doi:10.1145/293347.293348.

6 M. A. Babenko, P. Gawrychowski, T. Kociumaka, and T. A. Starikovskaya. Wavelet trees meet suffix trees. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 572–591, 2015. doi:10.1137/1.9781611973730.39.

7 D. Belazzougui and S. J. Puglisi. Range predecessor and lempel-ziv parsing. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 2053–2071, 2016. doi:10.1137/1.9781611974331.ch143.

8 S. Chechik. Improved distance oracles and spanners for vertex-labeled graphs. In *ESA*, pages 325–336, 2012. doi:10.1007/978-3-642-33090-2_29.

9 K. Clarkson. Nearest-neighbor searching and metric space dimensions. In G. Shakhnarovich, T. Darrell, and P. Indyk, editors, *Nearest-Neighbor Methods for Learning and Vision: Theory and Practice*, pages 15–59. MIT Press, 2006. URL: http://kenclarkson.org/nn_survey/p.pdf.

10 K. L. Clarkson. Nearest neighbor queries in metric spaces. *Discrete Comput. Geom.*, 22(1):63–93, 1999.

11 H. Cohen and E. Porat. Fast set intersection and two-patterns matching. *Theor. Comput. Sci.*, 411(40-42):3795–3800, 2010.

12 R. Cole and L.-A. Gottlieb. Searching dynamic point sets in spaces with bounded doubling dimension. In *38th annual ACM symposium on Theory of computing*, pages 574–583. ACM, 2006. doi:10.1145/1132516.1132599.

13 M. Crochemore, C. S. Iliopoulos, M. Kubica, M. S. Rahman, G. Tischler, and T. Walen. Improved algorithms for the range next value problem and applications. *Theor. Comput. Sci.*, 434:23–34, 2012.

14 J. Fischer, T. Gagie, T. Kopelowitz, M. Lewenstein, V. Mäkinen, L. Salmela, and N. Välimäki. Forbidden patterns. In *LATIN*, 2012. doi:10.1007/978-3-642-29344-3_28.

15 A. Grønlund and S. Pettie. Threesomes, degenerates, and love triangles. In *Proceedings of the 55th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 621–630, 2014.

16 D. Hermelin, A. Levy, O. Weimann, and R. Yuster. Distance oracles for vertex-labeled graphs. In *Automata, Languages, and Programming - 38th International Colloquium, ICALP (2)*, 2011. doi:10.1007/978-3-642-22012-8_39.

17 W. Hon, R. Shah, S. V. Thankachan, and J. S. Vitter. String retrieval for multi-pattern queries. In *SPIRE*, 2010.

18 W. Hon, R. Shah, S. V. Thankachan, and J. S. Vitter. Document listing for queries with excluded pattern. In *CPM*, 2012.

19 P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *30th Annual ACM Symposium on Theory of Computing*, pages 604–613, May 1998.

20 D. Karger and M. Ruhl. Finding nearest neighbors in growth-restricted metrics. In *34th Annual ACM Symposium on the Theory of Computing*, pages 63–66, 2002.

21 O. Keller, T. Kopelowitz, S. Landau Feibish, and M. Lewenstein. Generalized substring compression. *Theor. Comput. Sci.*, 525:42–54, 2014. doi:10.1016/j.tcs.2013.10.010.

22 O. Keller, T. Kopelowitz, and M. Lewenstein. Range non-overlapping indexing and successive list indexing. In *Algorithms and Data Structures, 10th International Workshop, WADS*, pages 625–636, 2007. doi:10.1007/978-3-540-73951-7_54.

23 T. Kopelowitz, S. Pettie, and E. Porat. Higher lower bounds from the 3SUM conjecture. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1272–1287, 2016. doi:10.1137/1.9781611974331.ch89.

24 R. Krauthgamer and J. R. Lee. Navigating nets: Simple algorithms for proximity search. In *15th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 791–801, January 2004.

25 K. Green Larsen, J. I. Munro, J. Sindahl Nielsen, and S. V. Thankachan. On hardness of several string indexing problems. In *CPM*, 2014.

26 M. Lewenstein. Orthogonal range searching for text indexing. In *Space-Efficient Data Structures, Streams, and Algorithms - Papers in Honor of J. Ian Munro on the Occasion of His 66th Birthday*, pages 267–302, 2013.

27 J. I. Munro, Y. Nekrich, and J. Scott Vitter. Fast construction of wavelet trees. In *String Processing and Information Retrieval - 21st International Symposium, SPIRE*, pages 101–110, 2014.

28 S. Muthukrishnan. Efficient algorithms for document retrieval problems. In *SODA*, pages 657–666. ACM/SIAM, 2002.

29 Y. Nekrich and G. Navarro. Sorted range reporting. In *Algorithm Theory - SWAT 2012 - 13th Scandinavian Symposium and Workshops*, pages 271–282, 2012.

30 M. Pătraşcu. Towards polynomial lower bounds for dynamic problems. In *STOC*, pages 603–610. ACM, 2010. doi:10.1145/1806689.1806772.

31 C. Yu, W. Hon, and B. Wang. Improved data structures for the orthogonal range successor problem. *Comput. Geom.*, 44(3):148–159, 2011.