Fast algorithm for two-dimensional pattern matching with k mismatches

Adam Górkiewicz[1], Paweł Gawrychowski[2], Tatiana Starikovskaya[3], and Jonas Ellert[4]

[1]?
[2]?
[3]?
[4]?

Abstract

1 Introduction

All-substring Hamming distance (1DASHD)
INPUT: String T of length n and string P of length $m \leq n$.
OUTPUT: The Hamming distance between P and every substring of T that has length m.

Pattern matching with k mismatches (1DPMWM)
INPUT: String T of length n, string P of length $m \leq n$, integer $k \geq 0$.
OUTPUT: The same as for 1DASHD, but if any distance is larger than k, the result for that substring should equal ∞.

Fact 1. *Pattern matching with k mismatches can be solved in $\tilde{\mathcal{O}}((m + k\sqrt{m})n/m)$ time.*

Two-dimensional all-substring Hamming distance (2DASHD)
INPUT: 2D string T of size $n \times n$, 2D string P of size $m \times m$ where $m \leq n$.
OUTPUT: The Hamming distance between P and every $m \times m$ substring of T.

Two-dimensional pattern matching with k mismatches (2DPMWM)
INPUT: 2D string T of size $n \times n$, 2D string P of size $m \times m$ where $m \leq n$, integer $k \geq 0$.
OUTPUT: The same as for 2DASHD, but if any distance is larger than k, the result for that substring should equal ∞.

Theorem 1. *Two-dimensional pattern matching with k mismatches can be solved in $\tilde{\mathcal{O}}((m^2 + mk^{5/4})n^2/m^2)$ time.*

2 One-dimensional generalizations

In this section we explore some of the methods used for one-dimensional strings. Specifically, as our goal is to generalise the solution for pattern matching with k mismatches described in [2], we are especially interested in two-dimensional variants of the techniques used to solve the one-dimensional case. We consider two simple reduction methods.

Wildcard padding Consider the input strings for the two aforementioned two-dimensional problems – an $n \times n$ text T and $m \times m$ pattern P. We construct one-dimensional strings T' and P' by "flattening" T and P with some extra padding characters. Specifically:

$$T' = T[0] \; ?^m \; T[1] \; ?^m \; \dots \; ?^m \; T[n-1] \; ?^m,$$
$$P' = P[0] \; ?^n \; P[1] \; ?^n \; \dots \; ?^n \; P[m-1] \; ?^n$$

where $T[0], \dots, T[n-1]$ and $P[0], \dots, P[m-1]$ represent subsequent rows of T and P and ? is the **wildcard** symbol that matches with every single character.

Observation 1. *The Hamming distance between $T[i \mathinner{.\,.} i+m-1, j \mathinner{.\,.} j+m-1]$ and P is equal to the Hamming distance between $T'[i(n+m)+j \mathinner{.\,.} (i+m)(n+m)+j-1]$ and P'. As a result, any solution for* 1DASHD / 1DPMWM, *which allows the text and pattern to contain wildcard symbols, can be easily generalized to solve* 2DASHD / 2DPMWM, *respectively.*

Unfortunately, the most effective known algorithms for 1DPMWM rely on periodicity ([1], [2]) and do not allow wildcard symbols, thus, they cannot be easily generalized. There are however two useful solutions for the 1DASHD problem, which can.

Fact 2. *There exists an algorithm which solves* 1DASHD *in $\tilde{\mathcal{O}}(n|\Sigma|)$ time. [maybe a reference?] It allows wildcard symbols in T and P.*

Fact 3. *There exists a $(1+\varepsilon)$-approximate algorithm which solves* 1DASHD *in $\tilde{\mathcal{O}}(n)$ time. It was first introduced in [3]. It allows wildcard symbols in T and P.*

Corollary 2. *By Observation 1. and Fact 2., there exists an algorithm which solves* 2DASHD *in $\tilde{\mathcal{O}}(n^2|\Sigma|)$ time.*

Corollary 3. *By Observation 1. and Fact 3., there exists a $(1+\varepsilon)$-algorithm which solves* 2DASHD *in $\tilde{\mathcal{O}}(n^2)$ time.*

Column compression Kangaroo jumps go here.

3 Preliminaries

Definition 1 (Two-dimensional string)**.** We define a **string** S as an ordered pair $(S^{\mathbf{d}}, S^{\mathbf{f}})$ where $S^{\mathbf{d}} \subseteq \mathbb{Z}^2$ is a finite set of two-dimensional integer vectors and $S^{\mathbf{f}} : S^{\mathbf{d}} \to \Sigma$ is a function mapping the vectors to characters. For simplicity we will sometimes write $S(u)$ to denote $S^{\mathbf{f}}(u)$ for $u \in S^{\mathbf{d}}$. We will also sometimes write $u \in S$ to denote that $u \in S^{\mathbf{d}}$. We define a **substring** of S as a string R such that $R^{\mathbf{d}} \subseteq S^{\mathbf{d}}$ and $R(u) = S(u)$ for all $u \in R$. We say that a string S is **partitioned** into its substrings $R_1, \dots, R_\ell$ when the sets $R_1^{\mathbf{d}}, \dots, R_\ell^{\mathbf{d}}$ partition $S^{\mathbf{d}}$. We call a string S **monochromatic** if $S^{\mathbf{f}}[S^{\mathbf{d}}] = \{ \alpha \}$ for some $\alpha \in \Sigma$.

Definition 2 (Shifting)**.** For a set of vectors V and a vector u, we denote $V+u$ as $\{ v+u : v \in V \}$. For a string S and a vector u, we denote $S+u$ as a string R such that $R^{\mathbf{d}} = S^{\mathbf{d}} + u$ and $R^{\mathbf{f}}(v) = S^{\mathbf{f}}(v-u)$ for $v \in R^{\mathbf{d}}$. Intuitively, we shift the set of vectors while maintaining their character values.

Definition 3 (Hamming distance)**.** Consider two strings S, R. We define

$$\mathrm{Ham}(S,R) = |\{ u : u \in S, u \in R, S(u) \neq R(u) \}|.$$

Consider a text string T and a pattern string P where

$$T^{\mathbf{d}} = \left\{ 0, \ldots, \frac{3}{2}n - 1 \right\}^2$$
$$P^{\mathbf{d}} = \{ 0, \ldots, n - 1 \}^2$$

for some integer n such that $4|n$. There exists an algorithm which for a given integer k calculates $\mathrm{Ham}(P + q, T)$

Definition 4 (Vector operators)**.** For a vector $u \in \mathbb{Z}^2$ we refer to its coordinates as $x(u), y(u)$. For $u, v \in \mathbb{Z}^2$ we denote $u \cdot v = x(u) \cdot x(v) + y(u) \cdot y(v)$ and $u \times v = x(u) \cdot y(v) - y(u) \cdot x(v)$. Note that alternatively $u \cdot v = |u||v| \cos \alpha$ and $u \times v = |u||v| \sin \alpha$ where α is the angle between u and v.

Definition 5 (Quadrants)**.** We define the four **quadrants** as

$$\mathcal{Q}_1 = (0, +\infty) \times [0, +\infty), \quad \mathcal{Q}_2 = (-\infty, 0] \times (0, +\infty), \quad \mathcal{Q}_3 = (-\infty, 0) \times (-\infty, 0], \quad \mathcal{Q}_4 = [0, +\infty) \times (-\infty, 0).$$

4 Main Algorithm

Consider a text string T and a pattern string P where

$$T^{\mathbf{d}} = \left\{ 0, \ldots, \frac{3}{2}n - 1 \right\}^2$$
$$P^{\mathbf{d}} = \{ 0, \ldots, n - 1 \}^2$$

for some integer n such that $4|n$. There exists an algorithm which for a given integer k calculates $\mathrm{Ham}(P + q, T)$

Theorem 4 (Main result)**.** *for such $q \in \{ 0, \ldots, \frac{1}{2}n \}^2$ for which the result is at most k and returns ∞ for the rest. The time complexity of the algorithm is $\tilde{\mathcal{O}}(n^2 + nk^{5/4})$.*

5 Description of Algorithm 4.

Firstly, we run a two-dimensional variant of Karloff's $(1 + \varepsilon)$-algorithm with $\varepsilon = 1$ matching the pattern with the text. We find the set Q of vectors q for which the estimated value of $\mathrm{Ham}(P + q, T)$ is at most $2k$. We return ∞ for $q \notin Q$ and to calculate the exact result for $q \in Q$ we distinguish two cases depending on the size of Q.

5.1 Solution for few queries

Assume that $|Q| \leq 2n + n^2/k$. For every $q \in Q$ we explicitly detect all mismatches using the "kangaroo jumps" technique in $\mathcal{O}(k)$ operations. In total, the algorithm takes $\tilde{\mathcal{O}}(n^2 + nk)$ time.

5.2 Solution for many queries

Assume that $|Q| > 2n + n^2/k$. We take advantage of the fact that some occurrences of the pattern in the text must have a large overlap, and thus the pattern must be periodic. Consequently, it can be decomposed into some regularly structured monochromatic substrings. We employ a similar decomposition approach for the text and then calculate the result by summing the contributions of every pair of pattern and text substrings.

We start by defining two-dimensional periodicity and finding suitable periods of the pattern.

Definition 6 (Periodicity)**.** Consider any vector $\delta \in \mathbb{Z}^2$. We say that a string S has an ℓ-period δ if

$$\mathrm{Ham}(S+\delta, S) \le \ell.$$

Lemma 1. *For every $u, v \in Q$, a vector $u - v$ is an $8k$-period of P.*

Theorem 5. *For an integer $\ell > 0$ and a set of vectors $U \subseteq \{0, \dots, \ell\}^2$ such that $|U| > 4\ell$ there exist $s, t, s', t' \in U$ such that $w = t - s$ and $w' = t' - s'$ hold the following conditions:*

- *$w, w' \neq \vec{0}$,*
- *$|w||w'| \le 22\frac{\ell^2}{|U|}$,*
- *$|\sin\alpha| \ge \frac{1}{2}$ where α is the angle between w and w',*
- *$w, w', -w, -w'$ are all contained in different quadrants.*

There exists an algorithm which finds such w, w' in $\tilde{\mathcal{O}}(|U|)$ operations.

By running Algorithm 5. on the set Q we obtain vectors $\varphi \in \mathcal{Q}_4$ and $\psi \in \mathcal{Q}_1$ which by Lemma 1. are $\mathcal{O}(k)$-periods of P. We use them as constants throughout the rest of the description along with $m = \varphi \times \psi$. Note that because $|Q| > n + n^2/k$, we have $m \le |\varphi||\psi| = \mathcal{O}(\min\{n, k\})$.

We will abuse the notation and for $u \in \mathbb{Z}^2$ write $\varphi(u), \psi(u)$ to denote values $\varphi \times u$ and $\psi \times u$.

Definition 7 (Lattice function)**.** We call $\mathcal{F} : \mathbb{Z}^2 \to \{1, \dots, m\}$ a lattice function if

$$\mathcal{F}(u) = \mathcal{F}(v) \iff \exists_{s,t \in \mathbb{Z}}\; u = v + s\varphi + t\psi.$$

Lemma 2. *There exists a lattice function (note that the values are from 1 to m).*

We choose any lattice function $\mathcal{L}$ and use it consistently throughout the description, as a way to help with the notation. We will now show how to decompose the pattern and the text and how to calculate the result with the help of some auxiliary algorithms.

Definition 8 (Parquet)**.** Consider a set $U \subseteq \mathbb{Z}^2$. We call U a **parquet** if there exist some values (restrictions) $x_0, x_1, y_0, y_1, \varphi_0, \varphi_1, \psi_0, \psi_1 \in \mathbb{Z}$ such that

$$U = \{u : u \in \mathbb{Z}^2, x_0 < x(u) \le x_1, y_0 < y(u) \le y_1, \varphi_0 < \varphi(u) \le \varphi_1, \psi_0 < \psi(u) \le \psi_1\}.$$

If some existing restrictions hold additional conditions, we classify U as a special case of parquet:

a) if $x_1 - x_0 \ge |x(\varphi)| + |x(\psi)|$ and $y_1 - y_0 \ge |y(\varphi)| + |y(\psi)|$, then we call U a **spacious** parquet,

b) if $x_0, y_0 = -\infty$ and $x_1, y_1 = +\infty$, then we call U a **simple** parquet,

c) if $x_0, y_0, \varphi_0, \psi_0 = -\infty$ and $x_1, y_1 = +\infty$, then we call U a **primitive** parquet.

Note that every primitive parquet is simple and every simple parquet is spacious.

Definition 9 (Subparquet)**.** Consider a set $V \subseteq \mathbb{Z}^2$. We call V a **subparquet** if there exist a parquet U and a value $\gamma \in \{1, \dots, m\}$ such that

$$V = \{u : u \in U, \mathcal{L}(u) = \gamma\}.$$

We call V a spacious/simple/primitive subparquet when there exists U which is (correspondingly) a spacious/simple/primitive parquet. We will abuse the notation and for non-empty V write $\mathcal{L}(V)$ to (unambiguously) denote the value γ.

Definition 10 (Parquet string)**.** For a string S, if $S^{\mathbf{d}}$ is a spacious/simple (sub-)parquet, then we call S a spacious/simple (sub-)parquet string. Note that since primitive (sub-)parquets are infinite, we do not extend their notion to strings.

Definition 11 (Active text)**.** Consider a set $U = \bigcup_{q \in Q} P^{\mathbf{d}} + q$. We define substrings $T_a = (U, T^{\mathbf{f}})$ and $T_b = (T^{\mathbf{d}} \setminus U, T^{\mathbf{f}})$. We will call T_a the **active text** and T_b the **inactive text**. For every $u \in \mathbb{Z}^2$ we define its **border distance** as

$$\min \{ \|u - v\|_\infty : v \in (\mathbb{Z}^2 \setminus U) \}.$$

Lemma 3. *For every $q \in Q$ we have*

$$\mathrm{Ham}(P + q, T) = \mathrm{Ham}(P + q, T_a).$$

Theorem 6 (Periodic parquet decomposition)**.** *Consider a spacious/simple parquet string R with $\mathcal{O}(k)$-periods φ and ψ. It can be partitioned into $\mathcal{O}(k)$ monochromatic spacious/simple subparquet substrings, correspondingly. There exists an algorithm which finds those partitionings in $\tilde{\mathcal{O}}(|R^{\mathbf{d}}|)$ operations.*

Theorem 7 (Active text decomposition)**.** *There exists an algorithm which for any $\ell = \mathcal{O}(n)$ partitions T_a into a set of monochromatic simple subparquet substrings $\mathcal{U}$ and a substring F, such that $|\mathcal{U}| = \mathcal{O}(\min\{ n^2, \ell k \})$ and for every $u \in F$ its border distance is $\mathcal{O}(n/\ell)$. It does so in $\tilde{\mathcal{O}}(n^2)$ operations.*

Theorem 8 (Sparse Hamming)**.** *There exists an algorithm which for a set of monochromatic simple subparquet strings $\mathcal{U}$ and a set of monochromatic subparquet strings $\mathcal{V}$ calculates*

$$\sum_{U \in \mathcal{U}} \sum_{V \in \mathcal{V}} \mathrm{Ham}(U + q, V)$$

for any $q \in \mathbb{Z}^2$ in $\tilde{\mathcal{O}}(1)$ operations after $\tilde{\mathcal{O}}(\ell^2 + |\mathcal{U}||\mathcal{V}|)$ preprocessing time assuming that all strings are defined for vectors with coordinate values from $\{0, \dots, \ell\}$.

Theorem 9 (Dense Hamming)**.** *Consider a substring F of T_a such that for every $u \in F$ its border distance is less than ℓ for some integer ℓ. There exists an algorithm which calculates $\mathrm{Ham}(P + q, F)$ for every $q \in Q$ in total time $\tilde{\mathcal{O}}(n^2 + n\ell k^{1/2})$.*

As P is a spacious parquet string, we partition it using Algorithm 6. into a set of subparquet substrings $\mathcal{V}$. Subsequently we partition T_a using Algorithm 7. with $\ell = nk^{-3/4}$ into a set of simple subparquet substrings $\mathcal{U}$ and a substring F. For every $q \in Q$ we then have

$$\mathrm{Ham}(P + q, T) = \mathrm{Ham}(P + q, T_a) = \mathrm{Ham}(P + q, F) + \sum_{U \in \mathcal{U}} \sum_{V \in \mathcal{V}} \mathrm{Ham}(U - q, V)$$

which we calculate by summing the results of Algorithm 9 and Algorithm 8

6 Description of Algorithm 5.

Firstly, we find any closest pair of vectors $s, t \in U$ by running the standard $\tilde{\mathcal{O}}(|U|)$ time algorithm and denote $w = t - s$. We define a partial order $\leq_w$ where $v \leq_w u$ for some $u, v \in U$ when at least one condition holds:

(a) $u = v$,

(b) $u - v$ and w belong to the same quadrant,

(c) $\alpha \in (-\frac{\pi}{6}, \frac{\pi}{6})$ where α is the angle between w and $u - v$.

We find the longest chain C and the longest antichain A using dynamic programming in $\tilde{\mathcal{O}}(|U|)$ operations. We then find any closest pair of vectors $s', t' \in A$ and denote $w' = t' - s'$. We have the following inequalities:

(i) $|U| \leq |C||A|$ (by Dilworth's theorem),

(ii) $(|C| - 1)|w| \leq (1 + \sqrt{3})\ell$ (roughly by the fact that vectors in C must be increasing in a certain direction),

(iii) $(|A| - 1)|w'| \leq 2\ell$ (by using a similar argument for vectors in A).

By considering the assumpion $|U| > 4\ell$ it can be proven that $|w||w'| \leq 22\frac{\ell^2}{|U|}$ and the other conditions also hold.

7 Description of Algorithm 6.

Definition 12 (Lattice graph)**.** For a set $U \subseteq \mathbb{Z}^2$ we define its **lattice graph** $G_U = (U, E_U)$ where

$$E_U = \left\{ \{ u, u + \delta \} : \delta \in \{ \varphi, \psi \}, u \in U, u + \delta \in U \right\}$$

so every vector is connected with its translations by $\varphi, \psi, -\varphi, -\psi$.

Lemma 4. *If U is a spacious subparquet, then G_U is connected.*

Firstly, we partition R into a set of subparquet substrings $\mathcal{S}$. For every non-empty $S \in \mathcal{S}$ we consider a lattice graph $G_{S^{\mathbf{d}}}$. If S is not monochromatic, then since $G_{S^{\mathbf{d}}}$ is connected, there must exist a pair of neighbouring vectors v, w such that $S(v) \neq S(w)$. We select any such pair and partition S into spacious (or simple if S is simple) subparquet substrings S' and S'' such that $v \in S'$ and $w \in S''$. For example if $v = w + \varphi$, then $S' = \{ u : u \in S, \psi(u) \leq \psi(v) \}$ and $S'' = \{ u : u \in S, \psi(u) > \psi(v) \}$. In the cases when $v = w + \delta$ for $\delta \in \{ -\varphi, \psi, -\psi \}$ the construction in similar.

We can recursively partition S' and S'' further until we obtain monochromatic substrings. Because R has $\mathcal{O}(k)$-periods φ and ψ, the total number of neighbor pairs v, w such that $S(v) \neq S(w)$ is $\mathcal{O}(k)$ throughout all $S \in \mathcal{S}$. Thus the total number of recursive calls is $\mathcal{O}(k)$ and because $|\mathcal{S}| = \mathcal{O}(k)$, the total number of constructed substrings is $\mathcal{O}(k)$. The algorithm can be implemented to work in time $\tilde{\mathcal{O}}(|R^{\mathbf{d}}|)$.

8 Description of Algorithm 7.

We assume ℓ to be an even number smaller than $\frac{n}{4}$. We start by partitioning $T^{\mathbf{d}}$ into **tiles**. We define $\varphi_{\min} = \min \{ \varphi(u) : u \in T^{\mathbf{d}} \}$, analogously $\varphi_{\max}, \psi_{\min}, \psi_{\max}$ and denote $\delta_\varphi = \frac{\varphi_{\max} - \varphi_{\min}}{\ell}$, $\delta_\psi = \frac{\psi_{\max} - \psi_{\min}}{\ell}$. We define a tile with integer coordinates (s, t) as a set of vectors $u \in \mathbb{Z}^2$ such that

$$\varphi_{\min} + s\delta_\varphi < \varphi(u) \leq \varphi_{\min} + (s+1)\delta_\varphi,$$
$$\psi_{\min} + t\delta_\psi < \psi(u) \leq \psi_{\min} + (t+1)\delta_\psi.$$

For a fixed tile U, let's consider $x_{\min} = \min \{ x(u) : u \in U \}$, analogously $x_{\max}, y_{\min}, y_{\max}$ and a set

$$R = \{ u : u \in \mathbb{Z}^2, x_{\min} \leq x(u) \leq x_{\max}, y_{\min} \leq y(u) \leq y_{\max} \}.$$

We classify U into one of three types:

a) if $U \cap T_a = \emptyset$ then U is an inactive tile,

b) if $U \cap T_a \neq \emptyset$, $R \not\subseteq T_a$ then U a border tile,

c) if $U \cap T_a \neq \emptyset$, $R \subseteq T_a$ then U is an active tile.

We define B as a set of all $u \in T_a$ contained in a border tile and construct $F = (B, T^{\mathbf{f}})$. Let us denote $z = \frac{3n-2}{4}$. Consider a family of sets $\mathcal{R} = \{ R_i^1 \} \cup \{ R_i^2 \} \cup \{ R_i^3 \} \cup \{ R_i^4 \}$, where for every active tile U with coordinates (s,t) its members are placed into exactly one subset:

1) R_t^1 if $y_{\min} > z$, $x_{\max} \geq z$,

2) R_s^2 if $x_{\max} < z$, $y_{\max} \geq z$,

3) R_t^3 if $y_{\max} < z$, $x_{\min} \leq z$,

4) R_s^4 if $x_{\min} > z$, $y_{\min} \leq z$.

The number of non-empty sets $R \in \mathcal{R}$ is $\mathcal{O}(\ell)$. For each of them we consider $S = (R, T^{\mathbf{f}})$ which is a simple parquet string with $\mathcal{O}(k)$-periods φ and ψ, and we further partition it using Algorithm 6., thus constructing the set $\mathcal{U}$.

9 Description of Algorithm 8.

For $U \in \mathcal{U}$, $V \in \mathcal{V}$, the value $\mathrm{Ham}(U+q, V)$ either equals $|(U^{\mathbf{d}}+q) \cap V^{\mathbf{d}}|$ if $U[U^{\mathbf{d}}] \neq V[V^{\mathbf{d}}]$ or 0 otherwise. We have

$$\sum_{U \in \mathcal{U}} \sum_{V \in \mathcal{V}} \mathrm{Ham}(U+q, V) = \sum_{(A,B) \in \mathcal{F}} |(A+q) \cap B|$$

where $\mathcal{F} = \{ (U^{\mathbf{d}}, V^{\mathbf{d}}) : U \in \mathcal{U}, V \in \mathcal{V}, U[U^{\mathbf{d}}] \neq V[V^{\mathbf{d}}] \}$. For every $(A,B) \in \mathcal{F}$ we can find primitive subparquets $A_1, \ldots, A_4$ such that for every q we have

$$|(A+q) \cap B| = |(A_1+q) \cap B| - |(A_2+q) \cap B| - |(A_3+q) \cap B| + |(A_4+q) \cap B|$$

thus we will consider four instances of a problem of calculating $\sum_{(A,B) \in \mathcal{F}'} |(A+q) \cap B|$ where A is a primitive subparquet and B is a subparquet for all $(A,B) \in \mathcal{F}'$.

We will write $u \leq_{\varphi\psi} v$ to denote that $\varphi(u) \leq \varphi(v) \wedge \psi(u) \leq \psi(v)$ for some $u, v \in \mathbb{Z}^2$.

Theorem 10. *There exists a data structure which for a given set of vectors U and a set of parquets $\mathcal{S}$ calculates*

$$\sum_{v \in V} |\{ S : S \in \mathcal{S}, v \in S \}|$$

for a given query vector q where $V = \{ v : v \in U, v \leq_{\varphi\psi} q \}$. It requires $\tilde{\mathcal{O}}(|U| + |\mathcal{S}|)$ preprocessing time and $\tilde{\mathcal{O}}(1)$ query time.

We consider an array of data structures $J_1, \ldots, J_m$ described in Algorithm 10. We construct J_γ for a set of points $U_\gamma = \{ u : u \in \mathbb{Z}^2, |x(u)| \leq \ell, |y(u)| \leq \ell, \mathcal{L}(u) = \gamma \}$ and set of parquets $\mathcal{S}_\gamma$. To construct $\mathcal{S}_\gamma$ we consider every pair $(A,B) \in \mathcal{F}'$ and find a vector w and a parquet V such that

$$A = \{ u : u \leq_{\varphi\psi} w, \mathcal{L}(u) = \mathcal{L}(w) \},$$
$$B = \{ u : u \in (V+w), \mathcal{L}(u) = \mathcal{L}(B) \}.$$

The set $\mathcal{S}_\gamma$ contains the parquets V obtained for pairs (A,B) such that $\mathcal{L}(B-w) = \gamma$. We can obtain the result of $\sum_{(A,B) \in \mathcal{F}'} |(A+q) \cap B|$ by making a query to $J_{\mathcal{L}(q)}$ with vector q.

For explanation, if $\mathcal{L}(A+q) \neq \mathcal{L}(B)$, then $(A+q) \cap B = \emptyset$. Otherwise $\mathcal{L}(q) = \mathcal{L}(B-w) = \gamma$ and we have

$$(A+q) \cap B = \{ u : u \leq_{\varphi\psi} w+q, u \in (V+w), \mathcal{L}(u) = \mathcal{L}(B) \} = \{ v : v \leq_{\varphi\psi} q, v \in V, \mathcal{L}(v) = \gamma \}$$

10 Description of Algorithm 9.

To be done.

References

[1] Raphaël Clifford, Allyx Fontaine, Ely Porat, Benjamin Sach, and Tatiana Starikovskaya. The k-mismatch problem revisited. *CoRR*, abs/1508.00731, 2015.

[2] Pawel Gawrychowski and Przemyslaw Uznanski. Optimal trade-offs for pattern matching with k mismatches. *CoRR*, abs/1704.01311, 2017.

[3] Howard J. Karloff. Fast algorithms for approximately counting mismatches. *Inf. Process. Lett.*, 48(2):53–60, 1993.