Near-Optimal-Time Quantum Algorithms for Approximate Pattern Matching

Tomasz Kociumaka[1] **Jakob Nogler**[2] Philip Wellnitz[3]

[1]INSAIT

[2]ETH Zurich

[3]National Institute of Informatics, SOKENDAI

Strings and Similarity Measures Between Strings

- A *string* is a sequence of characters.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
S	a	b	a	a	b	a	a	b	a	a	b	a	a	b	a	a	b

Strings and Similarity Measures Between Strings

- A *string* is a sequence of characters.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
S	a	b	a	a	b	a	a	b	a	a	b	a	a	b	a	a	b

$S[12]$

Strings and Similarity Measures Between Strings

- A *string* is a sequence of characters.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
S	a	b	a	a	b	a	a	b	a	a	b	a	a	b	a	a	b

$S[3..10)$ $S[12]$

Strings and Similarity Measures Between Strings

- A *string* is a sequence of characters.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
S	a	b	a	a	b	a	a	b	a	a	b	a	a	b	a	a	b

$S[3..10)$ $S[12]$

- The *Hamming distance* $\mathrm{HD}(X, Y)$ counts the number of mismatches between X, Y.

X	b	b	a	a	a	b	b	b	a	b	b	a	a
Y	a	b	a	b	a	b	b	b	a	b	b	a	a

$\mathrm{HD}(X, Y) = 2$

Strings and Similarity Measures Between Strings

- A *string* is a sequence of characters.

```
     0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
S    a b a a b a a b a a b  a  a  b  a  a  b
```

$S[3..10)$ $S[12]$

- The *Hamming distance* $\mathrm{HD}(X, Y)$ counts the number of mismatches between X, Y.

```
X    b b a a a b b b a b b a a
Y    a b a b a b b b a b b a a
```

$\mathrm{HD}(X, Y) = 2$

- The *edit distance* $\mathrm{ED}(X, Y)$ measures the minimum number of insertions, deletions, and substitutions of characters to transform X into Y.

```
X    a b a a b a b a b a b b b a
Y    a b a b a b b b a b b a a
```

$\mathrm{ED}(X, Y) = 3$

Pattern Matching (PM)

Text T, $|T| = n$ a b a a b a b a b a b b b a b b a a a b b b a b b a a

Pattern P, $|P| = m$ a b a b a b b b a b b a a

Pattern Matching (PM)

Text T, $|T| = n$ `a b a a b a b a b a b b b a b b a a a b b b a b b a a`

Pattern P, $|P| = m$ `a b a b a b b b a b b a a`

- **Exact PM:** Compute $\mathrm{Occ}(P, T) := \{x \mid T[x\,..\,x+m) = P\}$.

Pattern Matching (PM)

Text T, $|T| = n$ `a b a a b a b a b a b b b a b b a a a b b b a b b a a`

Pattern P, $|P| = m$ `a b a b a b b b a b b a a`

$k = 2$

- **Exact PM:** Compute $\mathrm{Occ}(P, T) := \{x \mid T[x \mathinner{.\,.} x+m) = P\}$.
- **PM with mismatches:** Compute $\mathrm{Occ}^H_k(P, T) := \{x \mid \mathrm{HD}(T[x \mathinner{.\,.} x+m), P) \leq k\}$.

Pattern Matching (PM)

Text T, $|T| = n$ `a b a` ~~`a`~~ `b a b a b a b b b a b b a a a b b b a b b a a`

Pattern P, $|P| = m$ `a b a b a b b b a b b a a`

$k = 3$

- **Exact PM:** Compute $\mathrm{Occ}(P, T) := \{x \mid T[x\,..\,x+m) = P\}$.
- **PM with mismatches:** Compute $\mathrm{Occ}_k^H(P, T) := \{x \mid \mathrm{HD}(T[x\,..\,x+m), P) \leq k\}$.
- **PM with edits:** Compute $\mathrm{Occ}_k^E(P, T) := \{x \mid \exists y\ \mathrm{ED}(T[x\,..\,y), P) \leq k\}$.

The Model

We study pattern matching with mismatches/edits in the quantum setting:

The Model

We study pattern matching with mismatches/edits in the quantum setting:

- Input strings P/T given as oracle where queries can be made in **superposition**

The Model

We study pattern matching with mismatches/edits in the quantum setting:

- Input strings P/T given as oracle where queries can be made in **superposition**
- **Query complexity** $Q(n)$: counts number of queries to oracle

The Model

We study pattern matching with mismatches/edits in the quantum setting:

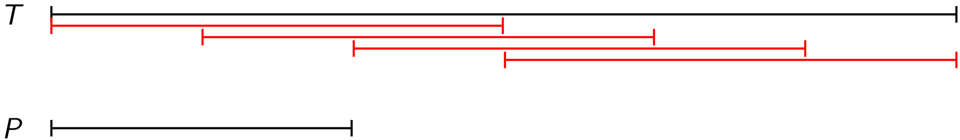
- Input strings P/T given as oracle where queries can be made in **superposition**
- **Query complexity** $Q(n)$: counts number of queries to oracle
- **Time complexity** $T(n)$: also counts the number of elementary gates

Length Assumption

We assume $\boldsymbol{n \leq 3/2 \cdot m}$.

Length Assumption

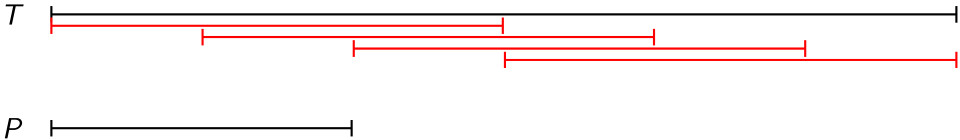
We assume $\boldsymbol{n \leq 3/2 \cdot m}$.



Divide T into $\Theta(n/m)$ blocks of length $n \leq 3/2 \cdot m$, and apply algorithm on each block.

Length Assumption

We assume $\boldsymbol{n \leq 3/2 \cdot m}$.

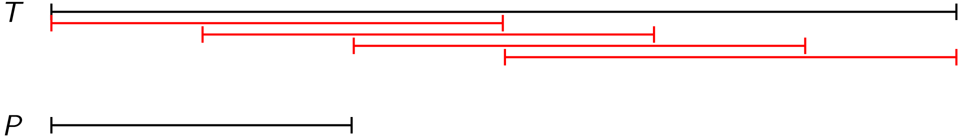


Divide T into $\Theta(n/m)$ blocks of length $n \leq 3/2 \cdot m$, and apply algorithm on each block.

- To compute $\mathrm{Occ}_k^H(P, T)$ (resp. $\mathrm{Occ}_k^E(P, T)$) this incurs a $\mathcal{O}(n/m)$ overhead.

Length Assumption

We assume $\boldsymbol{n \leq 3/2 \cdot m}$.



Divide T into $\Theta(n/m)$ blocks of length $n \leq 3/2 \cdot m$, and apply algorithm on each block.

- To compute $\mathrm{Occ}_k^H(P, T)$ (resp. $\mathrm{Occ}_k^E(P, T)$) this incurs a $\mathcal{O}(n/m)$ overhead.
- To compute $\mathrm{Occ}_k^H(P, T) \neq \emptyset$ (resp. $\mathrm{Occ}_k^H(P, T) \neq \emptyset$) incurs a $\mathcal{O}(\sqrt{n/m})$ overhead.

Our Results

Refererence	Setting	Query Complexity	Time Complexity	Notes
[CKW20]	mismatches	$\widetilde{\mathcal{O}}(k^2\sqrt{n})$	$\widetilde{\mathcal{O}}(k^2\sqrt{n})$	

Our Results

Refererence	Setting	Query Complexity	Time Complexity	Notes
[CKW20]	mismatches	$\widetilde{\mathcal{O}}(k^2\sqrt{n})$	$\widetilde{\mathcal{O}}(k^2\sqrt{n})$	
[JN23]	mismatches	$\widehat{\mathcal{O}}(k^{3/4}\sqrt{n})$	$\widehat{\mathcal{O}}(k\sqrt{n})$	Only verifies $\mathrm{Occ}_k^H(P,T) \neq \emptyset$

Our Results

Refererence	Setting	Query Complexity	Time Complexity	Notes
[CKW20]	mismatches	$\widetilde{\mathcal{O}}(k^2\sqrt{n})$	$\widetilde{\mathcal{O}}(k^2\sqrt{n})$	
[JN23]	mismatches	$\widehat{\mathcal{O}}(k^{3/4}\sqrt{n})$	$\widehat{\mathcal{O}}(k\sqrt{n})$	Only verifies $\mathrm{Occ}^H_k(P, T) \neq \emptyset$
This work	mismatches	$\widetilde{\mathcal{O}}(\sqrt{kn})$	$\widetilde{\mathcal{O}}(\sqrt{kn} + k^2)$	

Our Results

Refererence	Setting	Query Complexity	Time Complexity	Notes
[CKW20]	mismatches	$\widetilde{\mathcal{O}}(k^2\sqrt{n})$	$\widetilde{\mathcal{O}}(k^2\sqrt{n})$	
[JN23]	mismatches	$\widehat{\mathcal{O}}(k^{3/4}\sqrt{n})$	$\widehat{\mathcal{O}}(k\sqrt{n})$	Only verifies $\mathrm{Occ}_k^H(P, T) \neq \emptyset$
This work	mismatches	$\widetilde{\mathcal{O}}(\sqrt{kn})$	$\widetilde{\mathcal{O}}(\sqrt{kn} + k^2)$	
[CKW22]	edits	$\widetilde{\mathcal{O}}(k^{3.5}\sqrt{n})$	$\widetilde{\mathcal{O}}(k^{3.5}\sqrt{n})$	

Our Results

Refererence	Setting	Query Complexity	Time Complexity	Notes
[CKW20]	mismatches	$\widetilde{\mathcal{O}}(k^2\sqrt{n})$	$\widetilde{\mathcal{O}}(k^2\sqrt{n})$	
[JN23]	mismatches	$\widehat{\mathcal{O}}(k^{3/4}\sqrt{n})$	$\widehat{\mathcal{O}}(k\sqrt{n})$	Only verifies $\mathrm{Occ}_k^H(P,T) \neq \emptyset$
This work	mismatches	$\widetilde{\mathcal{O}}(\sqrt{kn})$	$\widetilde{\mathcal{O}}(\sqrt{kn} + k^2)$	
[CKW22]	edits	$\widetilde{\mathcal{O}}(k^{3.5}\sqrt{n})$	$\widetilde{\mathcal{O}}(k^{3.5}\sqrt{n})$	
[KNW24]	edits	$\widehat{\mathcal{O}}(\sqrt{kn})$	$\widehat{\mathcal{O}}(\sqrt{kn} + k^{3.5})$	

Our Results

Refererence	Setting	Query Complexity	Time Complexity	Notes
[CKW20]	mismatches	$\widetilde{\mathcal{O}}(k^2\sqrt{n})$	$\widetilde{\mathcal{O}}(k^2\sqrt{n})$	
[JN23]	mismatches	$\widehat{\mathcal{O}}(k^{3/4}\sqrt{n})$	$\widehat{\mathcal{O}}(k\sqrt{n})$	Only verifies $\mathrm{Occ}_k^H(P,T) \neq \emptyset$
This work	mismatches	$\widetilde{\mathcal{O}}(\sqrt{kn})$	$\widetilde{\mathcal{O}}(\sqrt{kn} + k^2)$	
[CKW22]	edits	$\widetilde{\mathcal{O}}(k^{3.5}\sqrt{n})$	$\widetilde{\mathcal{O}}(k^{3.5}\sqrt{n})$	
[KNW24]	edits	$\widehat{\mathcal{O}}(\sqrt{kn})$	$\widehat{\mathcal{O}}(\sqrt{kn} + k^{3.5})$	
This work	edits	$\widehat{\mathcal{O}}(\sqrt{kn})$	$\widehat{\mathcal{O}}(\sqrt{kn} + k^{3.5})$	

Our Results

Refererence	Setting	Query Complexity	Time Complexity	Notes
[CKW20]	mismatches	$\widetilde{\mathcal{O}}(k^2\sqrt{n})$	$\widetilde{\mathcal{O}}(k^2\sqrt{n})$	
[JN23]	mismatches	$\widehat{\mathcal{O}}(k^{3/4}\sqrt{n})$	$\widehat{\mathcal{O}}(k\sqrt{n})$	Only verifies $\mathrm{Occ}_k^H(P,T) \neq \emptyset$
This work	mismatches	$\widetilde{\mathcal{O}}(\sqrt{kn})$	$\widetilde{\mathcal{O}}(\sqrt{kn} + k^2)$	
[CKW22]	edits	$\widetilde{\mathcal{O}}(k^{3.5}\sqrt{n})$	$\widetilde{\mathcal{O}}(k^{3.5}\sqrt{n})$	
[KNW24]	edits	$\widehat{\mathcal{O}}(\sqrt{kn})$	$\widehat{\mathcal{O}}(\sqrt{kn} + k^{3.5})$	
This work	edits	$\widehat{\mathcal{O}}(\sqrt{kn})$	$\widehat{\mathcal{O}}(\sqrt{kn} + k^{3.5})$	

- Query complexity is optimal for $k = o(n)$.

Our Results

Refererence	Setting	Query Complexity	Time Complexity	Notes
[CKW20]	mismatches	$\widetilde{\mathcal{O}}(k^2\sqrt{n})$	$\widetilde{\mathcal{O}}(k^2\sqrt{n})$	
[JN23]	mismatches	$\widehat{\mathcal{O}}(k^{3/4}\sqrt{n})$	$\widehat{\mathcal{O}}(k\sqrt{n})$	Only verifies $\mathrm{Occ}_k^H(P,T) \neq \emptyset$
This work	mismatches	$\widetilde{\mathcal{O}}(\sqrt{kn})$	$\widetilde{\mathcal{O}}(\sqrt{kn}+k^2)$	
[CKW22]	edits	$\widetilde{\mathcal{O}}(k^{3.5}\sqrt{n})$	$\widetilde{\mathcal{O}}(k^{3.5}\sqrt{n})$	
[KNW24]	edits	$\widehat{\mathcal{O}}(\sqrt{kn})$	$\widehat{\mathcal{O}}(\sqrt{kn}+k^{3.5})$	
This work	edits	$\widehat{\mathcal{O}}(\sqrt{kn})$	$\widehat{\mathcal{O}}(\sqrt{kn}+k^{3.5})$	

- Query complexity is optimal for $k = o(n)$.
- We need the same query complexity to compute HD/ED between n-length strings.

Our Results

Refererence	Setting	Query Complexity	Time Complexity	Notes
[CKW20]	mismatches	$\widetilde{\mathcal{O}}(k^2\sqrt{n})$	$\widetilde{\mathcal{O}}(k^2\sqrt{n})$	
[JN23]	mismatches	$\widehat{\mathcal{O}}(k^{3/4}\sqrt{n})$	$\widehat{\mathcal{O}}(k\sqrt{n})$	Only verifies $\mathrm{Occ}_k^H(P,T) \neq \emptyset$
This work	mismatches	$\widetilde{\mathcal{O}}(\sqrt{kn})$	$\widetilde{\mathcal{O}}(\sqrt{kn} + k^2)$	
[CKW22]	edits	$\widetilde{\mathcal{O}}(k^{3.5}\sqrt{n})$	$\widetilde{\mathcal{O}}(k^{3.5}\sqrt{n})$	
[KNW24]	edits	$\widehat{\mathcal{O}}(\sqrt{kn})$	$\widehat{\mathcal{O}}(\sqrt{kn} + k^{3.5})$	
This work	edits	$\widehat{\mathcal{O}}(\sqrt{kn})$	$\widehat{\mathcal{O}}(\sqrt{kn} + k^{3.5})$	

- Query complexity is optimal for $k = o(n)$.
- We need the same query complexity to compute HD/ED between n-length strings.
- Time complexity is optimal for $k \leq n^{1/3}$ for mismatches and for $k \leq n^{1/6}$ for edits.

Our Results

Refererence	Setting	Query Complexity	Time Complexity	Notes
[CKW20]	mismatches	$\widetilde{\mathcal{O}}(k^2\sqrt{n})$	$\widetilde{\mathcal{O}}(k^2\sqrt{n})$	
[JN23]	mismatches	$\widehat{\mathcal{O}}(k^{3/4}\sqrt{n})$	$\widehat{\mathcal{O}}(k\sqrt{n})$	Only verifies $\mathrm{Occ}_k^H(P,T) \neq \emptyset$
This work	mismatches	$\widetilde{\mathcal{O}}(\sqrt{kn})$	$\widetilde{\mathcal{O}}(\sqrt{kn}+k^2)$	
[CKW22]	edits	$\widetilde{\mathcal{O}}(k^{3.5}\sqrt{n})$	$\widetilde{\mathcal{O}}(k^{3.5}\sqrt{n})$	
[KNW24]	edits	$\widehat{\mathcal{O}}(\sqrt{kn})$	$\widehat{\mathcal{O}}(\sqrt{kn}+k^{3.5})$	
This work	edits	$\widehat{\mathcal{O}}(\sqrt{kn})$	$\widehat{\mathcal{O}}(\sqrt{kn}+k^{3.5})$	

- Query complexity is optimal for $k = o(n)$.
- We need the same query complexity to compute HD/ED between n-length strings.
- Time complexity is optimal for $k \leq n^{1/3}$ for mismatches and for $k \leq n^{1/6}$ for edits.
- Offer advantage over classical algorithms for $k \leq n^{1/4}$ for mismatches and for $k \leq n^{1/7}$ for edits.

Previous Approach

[CKW20]: Find a candidate set $\mathrm{Occ}_k^H(P,T) \subseteq C$ in one of two forms.

$|C| = \mathcal{O}(k)$

C forms an arithmetic progression and $C \subseteq \mathrm{Occ}_{5k}^H(P,T)$

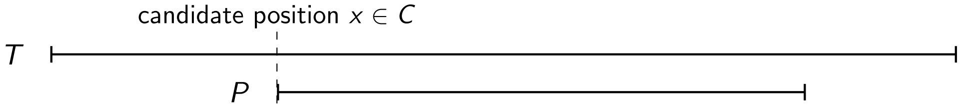
Previous Approach

[CKW20]: Find a candidate set $\mathrm{Occ}_k^H(P, T) \subseteq C$ in one of two forms.

- $|C| = \mathcal{O}(k)$

  bottleneck

- C forms an arithmetic progression and $C \subseteq \mathrm{Occ}_{5k}^H(P, T)$



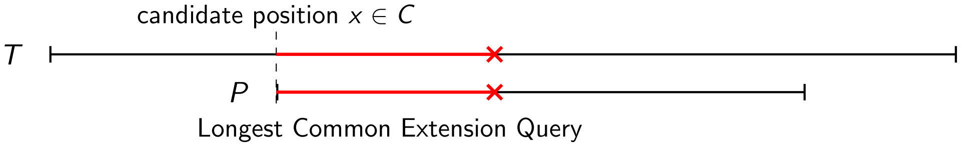
Previous Approach

[CKW20]: Find a candidate set $\mathrm{Occ}^H_k(P, T) \subseteq C$ in one of two forms.

$|C| = \mathcal{O}(k)$

bottleneck

C forms an arithmetic progression and $C \subseteq \mathrm{Occ}^H_{5k}(P, T)$



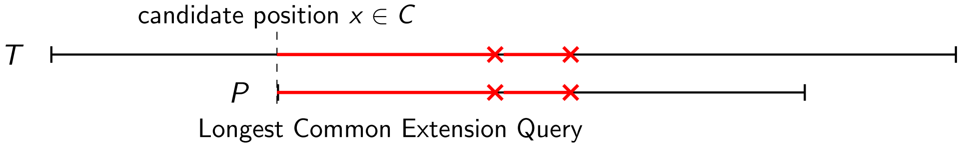
Previous Approach

[CKW20]: Find a candidate set $\mathrm{Occ}_k^H(P, T) \subseteq C$ in one of two forms.

- $|C| = \mathcal{O}(k)$

  bottleneck

- C forms an arithmetic progression and $C \subseteq \mathrm{Occ}_{5k}^H(P, T)$



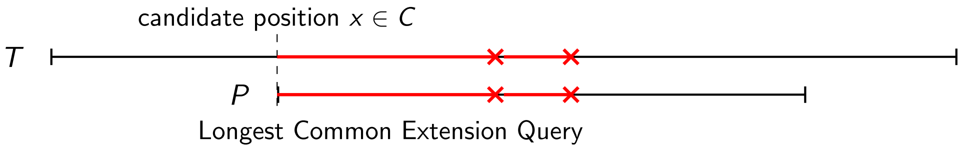
Previous Approach

[CKW20]: Find a candidate set $\mathrm{Occ}_k^H(P, T) \subseteq C$ in one of two forms.

$|C| = \mathcal{O}(k)$

bottleneck

C forms an arithmetic progression and $C \subseteq \mathrm{Occ}_{5k}^H(P, T)$



Problem: $\mathcal{O}(k)$ are too many candidate positions!

Workaround: Communication Complexity

[CKP19]: A subset $S \subseteq \mathrm{Occ}_k^H(P, T)$ of size $|S| = \mathcal{O}(\log n)$ suffices to encode $\mathrm{Occ}_k^H(P, T)$.

Workaround: Communication Complexity

[CKP19]: A subset $S \subseteq \mathrm{Occ}_k^H(P, T)$ of size $|S| = \mathcal{O}(\log n)$ suffices to encode $\mathrm{Occ}_k^H(P, T)$.

Q: Can we avoid verifying $\mathcal{O}(k)$ candidate positions, and only work with $\mathcal{O}(\log n)$ candidate positions?

Workaround: Communication Complexity

[CKP19]: A subset $S \subseteq \mathrm{Occ}^H_k(P, T)$ of size $|S| = \mathcal{O}(\log n)$ suffices to encode $\mathrm{Occ}^H_k(P, T)$.

Q: Can we avoid verifying $\mathcal{O}(k)$ candidate positions, and only work with $\mathcal{O}(\log n)$ candidate positions?

Theorem [CKP19]

There exists a subset $S \subseteq \mathrm{Occ}^H_k(P, T)$ of size $|S| = \mathcal{O}(\log n)$ such that the mismatch information for all $x \in S$, defined as

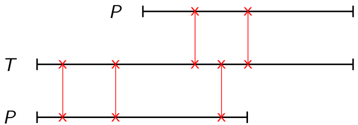
$$\mathrm{MI}(x) := \{(i, P[i], T[x+i]) \mid i \in [0..m) \text{ and } P[i] \neq T[x+i]\},$$

provides enough information to construct two strings $P^{\#}$ and $T^{\#}$ satisfying

$$\mathrm{Occ}^H_k(P, T) = \mathrm{Occ}^H_k(P^{\#}, T^{\#}).$$

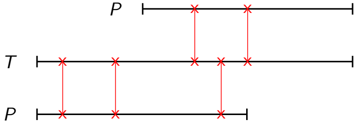
Construction of $P^{\#}$ and $T^{\#}$

Select a subset $\{0, n-m\} \subseteq S \subseteq \mathrm{Occ}_k^H(P, T)$.



Construction of $P^{\#}$ and $T^{\#}$

Select a subset $\{0, n-m\} \subseteq S \subseteq \mathrm{Occ}_k^H(P, T)$.

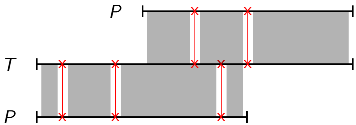


From $\{\mathrm{MI}(x) \mid x \in S\}$ we can infer a set $\mathcal{E}(S)$ of equalities of the form:

1. $T[i] = \sigma$
2. $P[j] = \sigma'$

Construction of $P^{\#}$ and $T^{\#}$

Select a subset $\{0, n-m\} \subseteq S \subseteq \mathrm{Occ}_k^H(P, T)$.

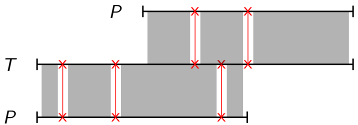


From $\{\mathrm{MI}(x) \mid x \in S\}$ we can infer a set $\mathcal{E}(S)$ of equalities of the form:

1. $T[i] = \sigma$
2. $P[j] = \sigma'$
3. $P[x..x') = T[y..y')$

Construction of $P^{\#}$ and $T^{\#}$

Select a subset $\{0, n-m\} \subseteq S \subseteq \mathrm{Occ}_k^H(P, T)$.



From $\{\mathrm{MI}(x) \mid x \in S\}$ we can infer a set $\mathcal{E}(S)$ of equalities of the form:

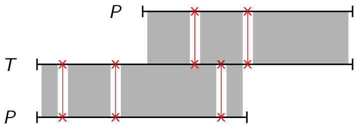
1. $T[i] = \sigma$
2. $P[j] = \sigma'$
3. $P[x..x') = T[y..y')$

Definition

$P^{\#}$, $T^{\#}$ are strings of the same length of P, T, and $T^{\#}[i] = \sigma$, $P^{\#}[j] = \sigma'$, and $P^{\#}[j] = T^{\#}[i]$ if and only if such equalities can be inferred from $\mathcal{E}(S)$.

Construction of $P^{\#}$ and $T^{\#}$

Select a subset $\{0, n-m\} \subseteq S \subseteq \mathrm{Occ}_k^H(P, T)$.



From $\{\mathrm{MI}(x) \mid x \in S\}$ we can infer a set $\mathcal{E}(S)$ of equalities of the form:

1. $T[i] = \sigma$
2. $P[j] = \sigma'$
3. $P[x..x') = T[y..y')$

Definition

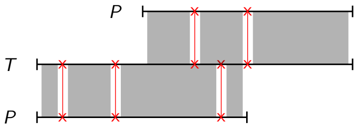
$P^{\#}, T^{\#}$ are strings of the same length of P, T, and $T^{\#}[i] = \sigma$, $P^{\#}[j] = \sigma'$, and $P^{\#}[j] = T^{\#}[i]$ if and only if such equalities can be inferred from $\mathcal{E}(S)$.

Theorem [CKP19]

For all $x \in [0..n-m]$ we have $\mathrm{HD}(P^{\#}, T^{\#}[x..x+m)) \geq \mathrm{HD}(P, T[x..x+m))$.
Moreover, if x divides $\gcd(S)$, then equality holds.

Construction of $P^{\#}$ and $T^{\#}$

Select a subset $\{0, n-m\} \subseteq S \subseteq \mathrm{Occ}_k^H(P, T)$.



From $\{\mathrm{MI}(x) \mid x \in S\}$ we can infer a set $\mathcal{E}(S)$ of equalities of the form:

1. $T[i] = \sigma$
2. $P[j] = \sigma'$
3. $P[x..x') = T[y..y')$

Definition

$P^{\#}, T^{\#}$ are strings of the same length of P, T, and $T^{\#}[i] = \sigma$, $P^{\#}[j] = \sigma'$, and $P^{\#}[j] = T^{\#}[i]$ if and only if such equalities can be inferred from $\mathcal{E}(S)$.

Theorem [CKP19]

For all $x \in [0..n-m]$ we have $\mathrm{HD}(P^{\#}, T^{\#}[x..x+m)) \geq \mathrm{HD}(P, T[x..x+m))$.
Moreover, if x divides $\gcd(S)$, then equality holds.

By choosing S s.t. $\gcd(S) = \gcd(\mathrm{Occ}_k^H(P, T))$ we obtain $P^{\#}, T^{\#}$ s.t. $\mathrm{Occ}_k^H(P, T) = \mathrm{Occ}_k^H(P^{\#}, T^{\#})$.

Constructing S through a Candidate Set

[CKW20]: Find a candidate set $\mathrm{Occ}_k^H(P, T) \subseteq C$ in one of two forms.

- $|C| = \mathcal{O}(k)$
- C forms an arithmetic progression and $C \subseteq \mathrm{Occ}_{5k}^H(P, T)$

Constructing S through a Candidate Set

[CKW20]: Find a candidate set $\mathrm{Occ}^H_k(P,T) \subseteq C$ in one of two forms.

$|C| = \mathcal{O}(k)$

For each $x \in C$, distinguish between $x \in \mathrm{Occ}^H_k(P,T)$ and $x \notin \mathrm{Occ}^H_{5k}(P,T)$

C forms an arithmetic progression and $C \subseteq \mathrm{Occ}^H_{5k}(P,T)$

Constructing S through a Candidate Set

[CKW20]: Find a candidate set $\mathrm{Occ}^H_k(P,T) \subseteq C$ in one of two forms.

$|C| = \mathcal{O}(k)$

For each $x \in C$, distinguish between $x \in \mathrm{Occ}^H_k(P,T)$ and $x \notin \mathrm{Occ}^H_{5k}(P,T)$

C forms an arithmetic progression and $C \subseteq \mathrm{Occ}^H_{5k}(P,T)$

The set C satisfies $\mathrm{Occ}^H_k(P,T) \subseteq C \subseteq \mathrm{Occ}^H_{5k}(P,T)$.
Choose $\{0, n-m\} \subseteq S \subseteq C$ s.t. $\gcd(S) = \gcd(C)$ and $|S| = \mathcal{O}(\log n)$.

Construct compressed $P^{\#}$ and $T^{\#}$ and compute $\mathrm{Occ}^H_k(P^{\#}, T^{\#})$.

Construction of $P^{\#}$ and $T^{\#}$

Theorem [KNW25]

Given S and $\mathrm{MI}(x)$ for all $x \in S$, we can construct a grammar-like representation $P^{\#}$ and $T^{\#}$ of size $\widetilde{\mathcal{O}}(k)$ in time $\widetilde{\mathcal{O}}(k^2)$. The grammar supports $\widetilde{\mathcal{O}}(1)$ time `PILLAR` operations.

Construction of $P^{\#}$ and $T^{\#}$

Theorem [KNW25]

Given S and $\mathrm{MI}(x)$ for all $x \in S$, we can construct a grammar-like representation $P^{\#}$ and $T^{\#}$ of size $\widetilde{\mathcal{O}}(k)$ in time $\widetilde{\mathcal{O}}(k^2)$. The grammar supports $\widetilde{\mathcal{O}}(1)$ time `PILLAR` operations.

- `PILLAR` operations: longest common prefix, internal pattern matching queries, etc...

Construction of $P^\#$ and $T^\#$

Theorem [KNW25]

Given S and $\mathrm{MI}(x)$ for all $x \in S$, we can construct a grammar-like representation $P^\#$ and $T^\#$ of size $\widetilde{\mathcal{O}}(k)$ in time $\widetilde{\mathcal{O}}(k^2)$. The grammar supports $\widetilde{\mathcal{O}}(1)$ time PILLAR operations.

- PILLAR operations: longest common prefix, internal pattern matching queries, etc...
- **[CKW20]**: output (representation of) $\mathrm{Occ}_k^H(P, T)$ using $\mathcal{O}(k^2)$ PILLAR operations.

Construction of $P^\#$ and $T^\#$

Theorem [KNW25]

Given S and $\mathrm{MI}(x)$ for all $x \in S$, we can construct a grammar-like representation $P^\#$ and $T^\#$ of size $\widetilde{\mathcal{O}}(k)$ in time $\widetilde{\mathcal{O}}(k^2)$. The grammar supports $\widetilde{\mathcal{O}}(1)$ time `PILLAR` operations.

- `PILLAR` operations: longest common prefix, internal pattern matching queries, etc...
- **[CKW20]**: output (representation of) $\mathrm{Occ}_k^H(P, T)$ using $\mathcal{O}(k^2)$ `PILLAR` operations.
- This means:
  1. We can construct a grammar for $P^\#$ and $T^\#$ in $\widetilde{\mathcal{O}}(k^2)$ time, and

Construction of $P^\#$ and $T^\#$

Theorem [KNW25]

Given S and $\mathrm{MI}(x)$ for all $x \in S$, we can construct a grammar-like representation $P^\#$ and $T^\#$ of size $\widetilde{\mathcal{O}}(k)$ in time $\widetilde{\mathcal{O}}(k^2)$. The grammar supports $\widetilde{\mathcal{O}}(1)$ time `PILLAR` operations.

- `PILLAR` operations: longest common prefix, internal pattern matching queries, etc...
- **[CKW20]**: output (representation of) $\mathrm{Occ}_k^H(P, T)$ using $\mathcal{O}(k^2)$ `PILLAR` operations.
- This means:
  1. We can construct a grammar for $P^\#$ and $T^\#$ in $\widetilde{\mathcal{O}}(k^2)$ time, and
  2. We can compute $\mathrm{Occ}_k^H(P^\#, T^\#) = \mathrm{Occ}_k^H(P, T)$ in $\widetilde{\mathcal{O}}(k^2)$ time.

Construction of $P^{\#}$ and $T^{\#}$

Theorem [KNW25]

Given S and $\mathrm{MI}(x)$ for all $x \in S$, we can construct a grammar-like representation $P^{\#}$ and $T^{\#}$ of size $\widetilde{\mathcal{O}}(k)$ in time $\widetilde{\mathcal{O}}(k^2)$. The grammar supports $\widetilde{\mathcal{O}}(1)$ time `PILLAR` operations.

- `PILLAR` operations: longest common prefix, internal pattern matching queries, etc...
- **[CKW20]**: output (representation of) $\mathrm{Occ}_k^H(P, T)$ using $\mathcal{O}(k^2)$ `PILLAR` operations.
- This means:
  1. We can construct a grammar for $P^{\#}$ and $T^{\#}$ in $\widetilde{\mathcal{O}}(k^2)$ time, and
  2. We can compute $\mathrm{Occ}_k^H(P^{\#}, T^{\#}) = \mathrm{Occ}_k^H(P, T)$ in $\widetilde{\mathcal{O}}(k^2)$ time.

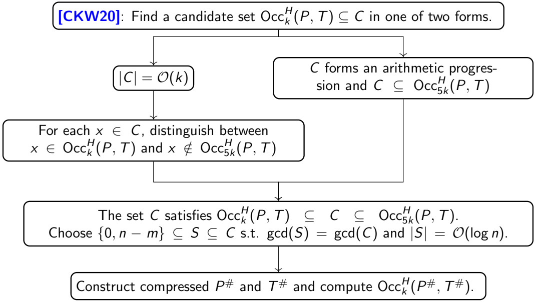
Theorem [KNW25]

Given N equality equations of the form $X[i..j) = X[i'..j')$ on a length-n string X, we can construct in time $\widetilde{\mathcal{O}}(N^2)$ a grammar-like representation of size $\widetilde{\mathcal{O}}(N)$ of a strings Y which:

1. satisfies all N equations, and
2. $Y[i] = Y[j]$ only when dictated by the equations.

Pattern Matching with Edits

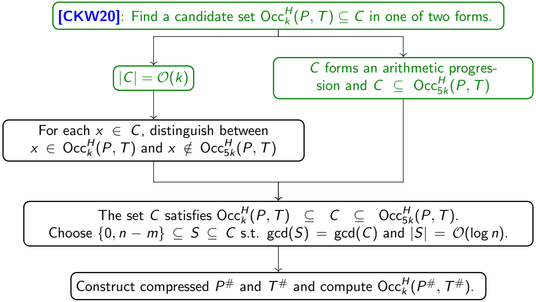
The edit case has similar properties to the hamming case:



Pattern Matching with Edits

The edit case has similar properties to the hamming case:

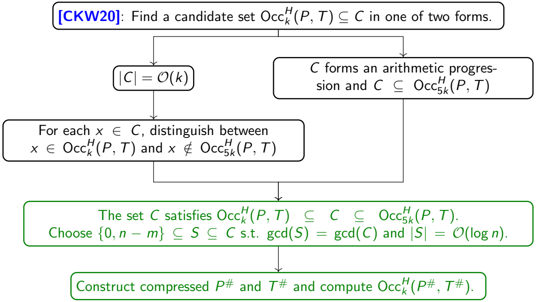
- **[CKW20]**: There is a candidate set that either has a small size or can be decomposed in arithmetic progressions.



Pattern Matching with Edits

The edit case has similar properties to the hamming case:

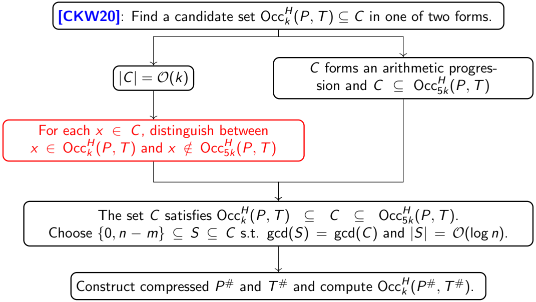
- **[CKW20]**: There is a candidate set that either has a small size or can be decomposed in arithmetic progressions.
- **[KNW24]**: To encode $\mathrm{Occ}^H_E(P, T)$ it suffices to consider a set S of size $|S| = \mathcal{O}(\log n)$ of k-edit occurrences + we can translate the encoded information to string equations.



Pattern Matching with Edits

The edit case has similar properties to the hamming case:

- **[CKW20]**: There is a candidate set that either has a small size or can be decomposed in arithmetic progressions.
- **[KNW24]**: To encode $\mathrm{Occ}_E^H(P, T)$ it suffices to consider a set S of size $|S| = \mathcal{O}(\log n)$ of k-edit occurrences + we can translate the encoded information to string equations.
- **[KNW24]**: Adapt to the quantum setting the classical algorithm from **[GKKS22]** for the GAP EDIT DISTANCE problem, i.e., distinguish between $\mathrm{ED}(X, Y)$ "small" and $\mathrm{ED}(X, Y)$ "large".



Search on Bounded-Error and Neutral Inputs

The quantum algorithm for the GAP EDIT DISTANCE good query complexity but not good time complexity.

Search on Bounded-Error and Neutral Inputs

The quantum algorithm for the Gap Edit Distance good query complexity but not good time complexity.

To address this, we consider the following search problem:

Search on Bounded-Error and Neutral Inputs

The quantum algorithm for the Gap Edit Distance good query complexity but not good time complexity.

To address this, we consider the following search problem:

- The universe $U = [0..n)$ is partitioned into three sets: $U = I^{+} \cup I^{\sim} \cup I^{-}$.

Search on Bounded-Error and Neutral Inputs

The quantum algorithm for the Gap Edit Distance good query complexity but not good time complexity.

To address this, we consider the following search problem:

- The universe $U = [0..n)$ is partitioned into three sets: $U = I^+ \cup I^\sim \cup I^-$.
- An oracle $f : U \to \{0, 1\}$ is provided with the following properties:
  - $\Pr[f(i) = 1] \geq 9/10$ if $i \in I^+$,
  - $\Pr[f(i) = 1] \leq 1/10$ if $i \in I^-$,
  - No guarantees on $\Pr[f(i) = 1]$ if $i \in I^\sim$.

Search on Bounded-Error and Neutral Inputs

The quantum algorithm for the Gap Edit Distance good query complexity but not good time complexity.

To address this, we consider the following search problem:

- The universe $U = [0..n)$ is partitioned into three sets: $U = I^{+} \cup I^{\sim} \cup I^{-}$.
- An oracle $f : U \to \{0, 1\}$ is provided with the following properties:
  - $\Pr[f(i) = 1] \geq 9/10$ if $i \in I^{+}$,
  - $\Pr[f(i) = 1] \leq 1/10$ if $i \in I^{-}$,
  - No guarantees on $\Pr[f(i) = 1]$ if $i \in I^{\sim}$.
- The goal is to output an element $i \in I^{+} \cup I^{\sim}$ or confirm that $I^{+} = \emptyset$.

Search on Bounded-Error and Neutral Inputs

The quantum algorithm for the GAP EDIT DISTANCE good query complexity but not good time complexity.

To address this, we consider the following search problem:

- The universe $U = [0..n)$ is partitioned into three sets: $U = I^+ \cup I^\sim \cup I^-$.
- An oracle $f : U \to \{0, 1\}$ is provided with the following properties:
  - $\Pr[f(i) = 1] \geq 9/10$ if $i \in I^+$,
  - $\Pr[f(i) = 1] \leq 1/10$ if $i \in I^-$,
  - No guarantees on $\Pr[f(i) = 1]$ if $i \in I^\sim$.
- The goal is to output an element $i \in I^+ \cup I^\sim$ or confirm that $I^+ = \emptyset$.

We give a $\widetilde{\mathcal{O}}(\sqrt{n})$-time algorithm, adapting the quantum algorithm for bounded-error inputs from **[HMDW03]**.

Open Questions

Open Questions

- Can we solve N equality equations on a string in time $\widetilde{\mathcal{O}}(N)$?

Open Questions

- Can we solve N equality equations on a string in time $\widetilde{\mathcal{O}}(N)$?
- Can we verify whether there is a k-mismatch occurrence in time $\widetilde{\mathcal{O}}(\sqrt{kn})$?

Open Questions

- Can we solve N equality equations on a string in time $\widetilde{\mathcal{O}}(N)$?
- Can we verify whether there is a k-mismatch occurrence in time $\widetilde{\mathcal{O}}(\sqrt{kn})$?

Thank you!