A Sublinear Algorithm for Weakly Approximating Edit Distance

Batu, Ergun, Kilian et al.

Presentation by Yoav Artzi
Today

- Present an interesting solution to the substring problem
- Show the power of the substring problem by examining weak approximation of edit distance
The Substring Problem

- Two strings: A and B
- Is B a continuous substring of A

A

B

ABCDEFGHJKLMNOPQR

EFGHIJK
The Substring Problem

1st Iteration: The Simple Solution

- Try all relevant indices in A
- At each index, try to check if B starts at that index
- \( O(|B| \cdot |A|) \)
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple</td>
<td>$O(nb)$</td>
</tr>
</tbody>
</table>

$|A| = n$

$|B| = b$
Random Sampling

- Suppose element $S (|S| = n)$ has a property $Z$ with probability $p$
- We would like to estimate $p$
- For any positive $\epsilon$ and $c$, there exists $d$ such that for $d \log n$ randomly sampled from $S$:
  - The fraction $p'$ of these samples with property $Z$ satisfies:
    $$ p - \epsilon/2 \leq p' \leq p + \epsilon/2 $$
  - with probability:
    $$ 1 - 1 / n^c $$
Random Sampling

Hoeffding bounds: \( P(|p' - p| \leq \frac{\varepsilon}{2}) \geq 1 - 2e^{-2m(\frac{\varepsilon}{2})^2} \)

We want: \( 1 - 2e^{-2m(\frac{\varepsilon}{2})^2} = 1 - \frac{1}{n^c} \)

After some arithmetic we get:

\( m \approx \frac{2c \log n}{\varepsilon^2} \Rightarrow d = \frac{2c}{\varepsilon^2} \Rightarrow m = d \log n \)
Starting with a Small Problem

The Substring Problem: Probabilistic Version

- Two strings: $A$ and $B$
- Accuracy parameter: $\frac{1}{c} < 1$
- $B$ is a continuous substring of $A$ with a hamming distance of at most $\frac{b}{c}$ with high probability
The Substring Problem

2nd Iteration: Probabilistic Solution

- Try all relevant indices on A
- For each such index randomly sample A
- Answer is correct with high probability
- Observation: We try each shift $t$ exactly once

Random shifts: \{1,4,5\}
### The Substring Problem

#### Solutions Runtime

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple</td>
<td>$O(nb)$</td>
</tr>
<tr>
<td>Probabilistic</td>
<td>$O(n \log b)$</td>
</tr>
</tbody>
</table>

| $|A| = n$ | $|B| = b$ |

$|A| = n$

$|B| = b$
The Substring Problem

3rd Iteration: The “Ruler”

- We would like to check each $t$ once
- But we would like to minimize the number of queries we do
- Intuition: until now we only moved on string $A$, maybe we can do better if we utilize both strings
- Goal: we would like a set of indices in $A$ and $B$ which will provide us with $i$ in $A$ and $j$ in $B$ s.t. $i-j=t$ for each $t$
- Actually we’ll implement something stronger: return all the shifts in $A$ in which $B$ starts
The Substring Problem

3rd Iteration: The “Ruler” – Check Each $t$ Once

- Let $u$ be the number of shifts allowed in $A$
- Define progressions of indices $A_i$ and $B_j$
- For each shift $t$ there exist a pair of $A_i$ and $B_j$ s.t. $i-j=t$

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>$b$</td>
</tr>
</tbody>
</table>

$u = n - b$

Assume $b >>> \sqrt{u}$

$A_i = \sqrt{u}, 2\sqrt{u}, 3\sqrt{u}, ... u$

$B_j = 1, 2, 3, \ldots \sqrt{u}$

$\forall t \in \{1 \ldots u\} \exists i, j : i - j = t$
The Substring Problem

3rd Iteration: The “Ruler” – Finding $i$ and $j$ s.t. $i-j=t$

Let $cen = \left\lfloor \frac{t}{\sqrt{u}} \right\rfloor + \sqrt{u}$

and $mil = t \mod \sqrt{u}$

Observe $A_i = A[cen \times \sqrt{u}]$ and $B_j = B[\sqrt{u} - mil]$

\[ i - j = cen \times \sqrt{u} - (\sqrt{u} - mil) = \left\lfloor \frac{t}{\sqrt{u}} \right\rfloor \times \sqrt{u} + \sqrt{u} - \sqrt{u} + t \mod \sqrt{u} = t \]

\[ A_i = \sqrt{u}, 2\sqrt{u}, 3\sqrt{u}\ldots u \]

\[ B_j = 1, 2, 3\ldots \sqrt{u} \]
The Substring Problem

3rd Iteration: The “Ruler”

- The indices form a ruler on A and B

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>n = 19,</td>
<td>b = 10</td>
</tr>
<tr>
<td>u = n - b = 9,</td>
<td>\sqrt{u} = 3</td>
</tr>
<tr>
<td>A_{1} = {3,6,9}</td>
<td></td>
</tr>
<tr>
<td>B_{1} = {1,2,3}</td>
<td></td>
</tr>
</tbody>
</table>

A

B
The Substring Problem

3rd Iteration: The “Ruler”

- Pick \( l \) random numbers
- For each ‘tick’ construct a fingerprint
- Now we can detect with high probability if there’s a match with shift \( t \)

Let \( l = \Theta(\log n) \)
Pick \( m_1, m_2 \ldots m_l \in [0, b - \sqrt{u}] \)
For example, let the fingerprint for \( A_i \) be:
\[ (A[i + m_1], A[i + m_2], \ldots, A[i + m_l]) \]
The Substring Problem

3rd Iteration: The “Ruler”

Return the group \( \{2\} \)

\( |A| = n = 9, |B| = b = 6 \)

\( u = n - b = 3, \sqrt{u} = \sqrt{3} \approx 2 \)

\( A_i = \{2,4\} \)

\( B_j = \{1,2\} \)

\( m_i \in [0, b - \sqrt{u}] \Rightarrow m_i - \{2,4\} \)

<table>
<thead>
<tr>
<th>Signature</th>
<th>A-list</th>
<th>B-list</th>
</tr>
</thead>
<tbody>
<tr>
<td>D,A</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>A,C</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>E,F</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>
The Substring Problem

3rd Iteration: The “Ruler”

- For each ‘tick’ we sample $\log b$ characters
- Query complexity is: $O(\max\{\sqrt{u}, u/b\} \log n)$
- Time complexity still not good enough

If $b \leq \sqrt{u}$ we’ll create an asymmetric ruler. We can only have $O(b)$ millimeter marks, so we only need $\Omega(n/b)$ centimeter marks.
### The Substring Problem

**Solutions Runtime**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple</td>
<td>$O(nb)$</td>
</tr>
<tr>
<td>Probabilistic</td>
<td>$O(n \log b)$</td>
</tr>
<tr>
<td>Basic “Ruler”</td>
<td>Query: $O(\max{\sqrt{u},u/b}\log b)$&lt;br&gt;Time: $O(u \log b)$</td>
</tr>
</tbody>
</table>

$|A| = n$

$|B| = b$

$u = n - b$
The Substring Problem

4th Iteration: Optimizing the “Ruler”

- Need an efficient way to couple the matched fingerprints and collect the valid shifts
- Checking all possible shifts still takes $O(\log n)$ time
The Substring Problem

4th Iteration: Optimizing the “Ruler”

- Hash the fingerprints:
  - The keys of the hash are the fingerprints themselves
  - The values for each fingerprints are the shifts for it in A and B stored separately in lists: A-list and B-list respectively
  - Populating the tree takes \( O(\max\{\sqrt{u}, u/b\}\log b) \)
- Can collect all shifts by going over the fingerprints and for each fingerprint return all the combination \( i - j \) when \( i \) is from the A-list and \( j \) from the B-list
- Total running time: \( O(\max\{\sqrt{u}, u/b\}\log b + u) \)
The Substring Problem

4th Iteration: Quantizing the “Ruler”

- If we allow the values in the hash table to be only multiples of $Q$, we will lower the number of values and will achieve:
  \[ O(\max\{\sqrt{u}, u/b\} \log n + u/Q) \]

- However, the group of $t$’s that we will receive in the end won’t be accurate, but will have a maximum deviation of $Q/2$

- For the substring problem we only care if the group of $t$’s is empty or not, which means we can push to:
  \[ O(\max\{\sqrt{u}, u/b\} \log n) \]

- However, later it will matter.
## The Substring Problem

### Solutions Runtime

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple</td>
<td>$O(nb)$</td>
</tr>
<tr>
<td>Probabilistic</td>
<td>$O(n \log b)$</td>
</tr>
<tr>
<td>Basic “Ruler”</td>
<td>Query: $O(\max{\sqrt{u}, u/b} \log b)$</td>
</tr>
<tr>
<td></td>
<td>Time: $O(u \log b)$</td>
</tr>
<tr>
<td>Quantized “Ruler”</td>
<td>Query: $O(\max{\sqrt{u}, u/b} \log b)$</td>
</tr>
<tr>
<td></td>
<td>Time: $O(\max{\sqrt{u}, u/b} \log b + u/Q)$</td>
</tr>
</tbody>
</table>

$|A| = n$

$|B| = b$

$u = n - b$
The Substring Problem

Summary

- Probabilistic algorithm
- Group of shifts returned has a worst case deviation of $Q/2$
- Complexity: $O(\max\{\sqrt{u}, u/b\} \log n + u/Q)$
The Edit Distance

- Two string A and B
- Minimum number of changes to make them equal
- Limited set of operation

\[ D(A,B) = 4 \]
The Problem

- Given $n$-character strings $A$ and $B$, $0 < \alpha < 1$ and $C > 1$
- If $D(A, B) \leq n^\alpha$, output “CLOSE” with probability of at least $2/3$
- If $D(A, B) > n/C$, output “FAR” with probability at least $2/3$
Motivation

- Computational biology
- Text processing
- Web searching
Importance of Sublinear Solution

- Traditional solution is a dynamic programming algorithm running in quadratic time
- Not good enough for the bioinformatics community
- Often, the actual distance doesn’t matter
- Can serve as part of an exact algorithm that will run in good amortized time
An Observation

- If two strings are close they will have many identical substrings whose locations in the respective strings is similar
General Idea

- Compare $B$ against $A$
- At each stage divide $B$ into $k$ intervals and assume the edit distance divides between these intervals
- Compare recursively the a random selection of these segment against $A$ with a bounded shift in $A$
- Once the edit distance is smaller than 1, treat the problem as a substring problem
Matching

An interval $I = B[s \ldots e]$ has a $(t, E)$ matching with respect to $A$ if for some interval $A[s' \ldots e']$, $s' = s + t$ and $D(A[s' \ldots e'], I) \leq E$. 

I has a $(16,2)$ Matching with respect to $A$
Coordinated Matching

Given a collection of intervals $I = I_1, \ldots, I_k$, we say that $I$ has a $(t, \sigma, E, D)$ coordinated matching with $A$ if all but $D$ of the intervals $I_i \in I$, $I_i$ has a $(t_i, E)$ matching with $A$, where $|t - t_i| \leq \sigma$.

Pay attention that two segments can “step” on each other.

$I$ has a $(4, 2, 1, 1)$ coordinated matching with $A$. 

$E = 1$
$t - t_i = 1$

$E = 0$
$t - t_i = 2$

$E = 1$
$t - t_i = 2$
Coordinated Matching to a matching

Let $A, I$ and $I_1, \ldots, I_k$ be as aforementioned.
Let $S = |I|$ and $S' = S / k$.
If $I_1, \ldots, I_k$ have a $(t, \sigma, \varepsilon S', \delta k)$ coordinated matching with $A$, then $I$ has a $(t, \beta S)$ matching with $A$, where:
\[ \beta = (2\sigma / S' + \varepsilon + \delta). \]

*This transformation is worst case*
Matching to a coordinated matching

Let $A, I$ and $I = I_1, \ldots, I_k$ be as aforementioned. Let $S = |I|$ and $S' = S/k$. Let $c > 1$ and $S > cE$. If $I$ has a $(t, E)$ matching with $A$ then $I$ has a $(t, E, cE/k, k/c)$ coordinated matching with $A$.

*This transformation is worst case*
Matching to a coordinated matching

A Special Case

Let $E, I, I_1, \ldots, I_k, S$ and $S'$ be as aforementioned.

If $I$ has a $(t, E)$ matching with $A$, and $k \geq E$, then $I_1, \ldots, I_k$ have a $(t, E, 0, E)$ coordinated matching with $A$.

This case will end the recursion and call the “ruler”
Where have we been and where we are going from here?

- What have we been doing so far?
  - We have a solution the substring problem.
  - We can have a way to deduce a match from a matches over subintervals and vice versa.

- What’s next?
  - We will combine what we learned about matches and coordinated matches to divide and conquer the problem.
  - Our stopping rule will be based on the $k \geq E$ special case.
  - The base problem will be treated as a substring problem.
The Problem

- Given $n$-character strings $A$ and $B$, $0<\alpha<1$ and $C>1$
- If $D(A,B) \leq n^\alpha$, output "CLOSE" with probability of at least $2/3$
- If $D(A,B) > n/C$, output "FAR" with probability at least $2/3$
The Code: DECIDE()

The strings to compare

DECIDE(A,B,α,C)

1. Set global constants:
   1. Choose small $\epsilon$ and large $c$
   2. Choose the quantization parameter
      \[ Q = \epsilon \min\{n^{1-\alpha}, n^{\alpha/2}\} \]

2. \( T = \text{MATCHES}(A,B,n^\alpha) \)

3. If \( T \) is nonempty, return “CLOSE” else return “FAR”
The Code: MATCHES()

MATCHES(A, I, E)
1. If $E < 1$ then:
   1. return SHIFTS(A, I)
2. Else:
   1. $k = \min\{\varepsilon n^{1-\alpha}, 2cE\}$
   2. Break I into k contiguous disjoint segments
   3. return COORD-MATCHES(A, \{I_i\}, E, cE/k, k/c)
The Code: COORD-MATCHES()

- Sample the subintervals.
- Find shifts that are good for each subinterval through calls to MATCHES.
- From the groups of shifts returned from the calls to MATCHES find the shifts which are good for enough subintervals.

Think of it like a common denominator
The Code: COORD-MATCHES()

COORD-MATCHES(A,\{I_1,\ldots,I_k\},\sigma,E,D)

1. Let \(d\) be s.t. \(d\log n\) samples provide a good answer with high probability and let \(l=d\log n\)
2. Choose samples \(i_1,\ldots,i_l\) in \([1\ldots k]\) uniformly
3. For each sample \(j\) \(T_j=\text{MATCHES}(A,I_j,E)\)
4. Let \(\Delta=(D/k+\varepsilon/2)l\)
5. Return the set \(\{t: T_j\cap[t-\sigma\ldots t+\sigma] \text{ is empty for at most } \Delta T_j\text{s}\}\)
The Code: COORD-MATCHES()

COORD-MATCHES(A,\{I_1,\ldots,I_k\},\sigma,E,D)

1. Let d be s.t. $d\log n$ samples provide a good answer with high probability and let $l=d\log n$
2. Choose samples $i_1,\ldots,i_l$ in $[1\ldots k]$ uniformly
3. For each sample $j$ $T_j=$MATCHES(A,$I_j$,E)
4. Let $\Delta=(D/k+\varepsilon/2)l$
5. Return the set $\{t : T_j\cap[t-\sigma\ldots t+\sigma]$ is empty for at most $\Delta T_j's\}$

The string to compare against
The subintervals to compare
Max deviation for each subinterval
Max edit dist. for each subinterval
Max # of subintervals with no match in A
Integrating the “Ruler”

Taking Quantization Deviation into Account

- We gave an efficient probabilistic solution to the substring problem with the $Q$ quantized “ruler”, but with a deviation of $Q/2$.
- At the last stage of the recursion we use SHIFTS which implements a this method.
- This deviation should be taken into account in the coordinated matching composed of these matchings.
Integrating the “Ruler”

Taking Quantization Deviation into Account

1. Interval $I$ has a $(t, \sigma, E, D)$ co-matching with $A$

$\Rightarrow$ It has a $(t[Q], \sigma + \frac{Q}{2}, E, D)$ quantized co-matching

2. $I$ has a $(t, \sigma, E, D)$ quantized co-matching with $A$

$\Rightarrow$ It has a $(t, \sigma + \frac{Q}{2}, E, D)$ co-matching.
Integrating the “Ruler” Arguments

We pass SHIFTS two arguments:
- The current ‘area’ of string A
- String I in length $n^{1-\alpha}$
Integrating the “Ruler”

Time Complexity

- The range of shifts available in SHIFTS is limited by the total edit distance of the two strings, hence:
  \[ u = 2n^\alpha \]

- The quantization factor is determined in the beginning:
  \[ Q = \varepsilon \min\{n^{1-\alpha}, n^{\alpha/2}\} \]

- Therefore, the time complexity is:
  \[ O(\max\{n^{\alpha/2}, n^{2\alpha-1}\} \log n^{1-\alpha} + n^\alpha / \min\{n^{1-\alpha}, n^{\alpha/2}\}) \]
The Code: \texttt{SHIFTS()}

\begin{itemize}
  \item The portion of $A$ to compare against
  \item The subinterval of $B$ to compare
  \item Quantization parameter
  \item Parameter that controls the allowed number of mismatches
\end{itemize}

\texttt{SHIFTS (A', I, Q, c)}

1. Initialize $u$ to create the “ruler”
2. Pick $O(d\log n)$ random valid shifts
3. Go over $A_i$’s and insert the fingerprints of $A$ into the hash
4. Go over $B_i$’s and insert the fingerprints of $B$ into the hash
5. Quantize the B-lists and A-lists for each fingerprint
6. For each fingerprint return the group of $t$’s
The Recursion Tree

COORD-MATCHES calls MATCHES on subintervals. However, MATCHES is not called on all the subintervals.

Looking for a match with a shift only
The Code: MATCHES()

MATCHES(A,I,E)

1. If $E < 1$ then:
   1. return SHIFTS(A,I)
2. Else:
   1. $k = \min\{\varepsilon n^{1-\alpha}, 2cE\}$
   2. Break $I$ into $k$ contiguous disjoint segments
   3. return COORD-MATCHES(A,$\{I_i\}$,$E$,$cE/k$, $k/\tilde{c}$)
The Code: COORD-MATCHES()

COORD-MATCHES(A,\{I_1,\ldots,I_k\},\sigma,E,D)

1. Let d be s.t. dlogn samples provide a good answer with high probability and let l=dlogn
2. Choose samples i_1,\ldots,i_l in [1\ldots k] uniformly
3. For each sample j T_j=MATCHES(A,I_j,E)
4. Let \Delta=(D/k+\epsilon/2)l
5. Return the set \{t : T_j \cap [t-\sigma..t+\sigma] is empty for at most \Delta T_j\}
The Recursion Depth

- The depth arises from the division into $k$ subintervals

$$k = \min\{\varepsilon n^{1-\alpha}, 2cE\}$$
The Recursion Depth

$k = 2c_1 E$

- Observe the call in MATCHES:
  
  $$COORD - MATCHES(A, \{I_{1..k}\}, E, \frac{cE}{k}, \frac{k}{c})$$

- The number of edit distances per matched subinterval is passed to the next MATCHES call:
  
  $$k = 2cE \Rightarrow \frac{cE}{k} = \frac{1}{2c} < 1$$

- During the next MATCHES call SHIFTS will be called and the recursion will end

- Therefore, this value is possible only at the end
The Recursion Depth

$k = \epsilon n^{1-\alpha}$

- At every stage $n$ is multiplied by $\Omega(n^{\alpha-1})$:
  \[ n \cdot n^{\alpha-1} = n^{1+\alpha-1}, n \cdot (n^{\alpha-1})^r = n^{1+(\alpha-1)r} \]
  Note that $\alpha < 1 \Rightarrow \alpha - 1 < 0$
- Therefore, it will take a constant (dependent on $\alpha$ only) number of steps to reach $n^{1-\alpha}$
- There are $n^{\alpha}$ segments of size $n^{1-\alpha}$ segments in $n$: \[ \frac{n}{n^{1-\alpha}} = n^{1+1-\alpha} = n^{\alpha} \]
- $E$ starts as $n^{\alpha}$ and we divide it between all the subintervals that we create. Therefore, $E \leq 1$ after these constant number of steps and the recursion will end.
- Notice that we didn’t treat $\epsilon$ and $c$. We will choose them properly later.
Now we will prove the conversion between coordinated matchings and matchings
Coordinated Matching to a matching

Let $A, I$ and $I_1, \ldots, I_k$ be as aforementioned. Let $S = |I|$ and $S' = S / k$. If $I_1, \ldots, I_k$ have a $(t, \sigma, \varepsilon S', \delta k)$ coordinated matching with $A$, then $I$ has a $(t, \beta S)$ matching with $A$, where:

$$\beta = (2\sigma / S' + \varepsilon + \delta).$$

*This transformation is worst case*
Coordinated Matching to a matching

Analyzing $\beta S$

$$\beta S = (2\sigma / S') + \varepsilon + (\delta)S$$

The subintervals with no matching contribute their entire length to the edit distance:

$$\delta k \times S' = \delta k \times S / k = \delta S$$
Coordinated Matching to a matching

Analyzing $\beta S$

$$\beta S = (2\sigma / S' + \varepsilon + \delta) S$$

Every interval with a match contribute to the total edit distance:

$$\varepsilon S' \times k = \varepsilon (S / k) k = \varepsilon S$$
Coordinated Matching to a matching

Analyzing $\beta S$

$$\beta S = (2\sigma / S + \varepsilon + \delta)S$$

Since intervals can oscillate to the sides they can “step” one over the other and “eat” each other’s match, which adds to the total edit distance:

$$k \times 2\sigma = \frac{S}{S'} \times 2\sigma = \frac{2\sigma}{S'} S$$

Each subinterval can be eaten from both sides

Max deviation for each subinterval

Edit size on each interval

Average shift

$$(t, \sigma, \varepsilon_S', \delta k)$$

# of unmatched subintervals

The max edit distance of the matched segments
Now we will go the other way:
Matching to a coordinated matching
Matching to a coordinated matching

Let $A$, $I_1, \ldots, I_k$ be as aforementioned. Let $S' = I_1$ and $S = S'/k$. Let $c > 1$ and $S > cE$. If $I$ has a $(t, E)$-matching with $A$, then $I$ has a $(t, E, cE/k, k/c)$-coordinated matching with $A$.

- The max edit distance of the matched segments.
- Average shift for each subinterval.
- Edit size on each interval.
- # of unmatched subintervals.
- Max deviation for each subinterval.

*This transformation is worst case.*
Matching to a coordinated matching

Analysis

- Let’s consider the matching from $A[s+t…q]$ to $I$ that has edit distance $E$.
- For each $I_i$, consider the smallest interval $I'_i$ of $A$ containing all the characters matched in $I_i$. If none exists, $I_i$ is not assigned.
- This process provides us with the desired coordinated matching.

$$(t, E, cE / k, k / c)$$

- Edit size on each interval
- Max deviation for each subinterval
- Average shift
- # of unmatched subintervals

The max edit distance of the matched segments
Matching to a coordinated matching

Example

A has a \((5,6)\) matching with \(A\)

\[
(t, E, cE / k, k / c)
\]

Max deviation for each subinterval

Edit size on each interval

Average shift

\# of unmatched subintervals

\[
E = 1
\]

\[
t - t_i = 1
\]

\[
E = 1
\]

\[
t - t_i = 2
\]

\[
E = 1
\]

\[
t - t_i = 2
\]

Let \(c = 2 > 1\)

We can see that \(S > cE\)

I has a \((5,6,3,2)\) coordinated matching with \(A\)
Matching to a coordinated matching

Analysis

We will show that we have at most $E$ edit operation in the new coordinated matching:

- Since edit operation don’t reorder the characters in $A$, $I’$‘s are disjoint
- Each edit operation affects only one $<I_i, I_i'>$ pair or an unassigned $I_i$.
- Hence, the sum edit distances across all pairs in with the sum of edits across unassigned $I_i$‘s is at most $E$.

Max deviation for each subinterval
Edit size on each interval
Average shift
# of unmatched subintervals

The max edit distance of the matched segments

$(t, E, cE/k, k/c)$
Matching to a coordinated matching

Analysis

We’ll prove that there can’t be more than \( k/c \) unmatched subintervals:

- This proof follows the logic behind Markov’s inequality
- Assume that \( m > k/c \) subintervals are matched with an edit distance > \( cE/k \) or are unmatched.
- The sum of the edit distance across these will be >\( E \) in contradiction to what we proved before:
  \[
  m \times \frac{cE}{k} > \frac{k}{c} \times \frac{cE}{k} \geq E
  \]
Matching to a coordinated matching

Analysis

- We'll now show that the shift between $I'_i$ and $I_i$ satisfies $|t_i - t| \leq E$
- Each operation on $A[s+t\ldots q]$ to obtain $I$ can create a shift of size 1 at most.
- Each such operation changes exactly one $<l'_i, l_i>$, thus each operation changes any $t_i$ by at most 1.
- However, at the end of the process, when $A$ is identical to $I$, $t_i = t$ for every $i$.
- Therefore, the original values of $t_i$ satisfies:

  \[ |t_i - t| \leq E \]
Back to analyzing the code
The Code: COORD-MATCHES()

COORD-MATCHES(A,\{I_1,...,I_k\},\sigma,E,D)

1. Let \( d \) be s.t. \( d \log n \) samples provide a good answer with high probability and let \( l = d \log n \)
2. Choose samples \( i_1,...,i_l \) in \([1...k]\) uniformly
3. For each sample \( j \) \( T_j = MATCHES(A, I_j, E) \)
4. Let \( \Delta = (D/k + \varepsilon/2)l \)
5. Return the set \( \{ t : T_j \cap [t-\sigma...t+\sigma] \text{ is empty for at most } \Delta T_j s \} \)
Approximating a Coordinated Matching Through Sampling

- Through what we previously showed we can see that a randomly selected $O(\log n)$ subintervals will approximate the behavior of the entire set. Further more:

With probability $1 - \frac{1}{n^c}$ the output $T$ of $COORD - MATCHES(A, \{I_{1...k}\}, \sigma, E, D, \varepsilon, c)$ has the following two properties:

1. $\{I_{1...k}\}$ has a $(t, \sigma, E, D)$ co-matching $\Rightarrow t \in T$
2. $t \in T$ $\Rightarrow \{I_{1...k}\}$ has a $(t, \sigma, E, D + \varepsilon k)$ co-matching
Approximating a Coordinated Matching Through Sampling

Analysis of COORD-MATCH()

1. \( \{I_{1..k}\} \) has a \((t, \sigma, E, D)\) co-matching \(\Rightarrow t \in T\)

\((t, \sigma, E, D)\) co-matching

\(\Rightarrow\) At most \(D\) unmatched with a \(t_i\) s.t. \(|t_i - t| \leq \sigma\)

\(\Rightarrow\) At most \(\frac{D}{k} + \frac{\varepsilon}{2}\) fraction are unmatched

\(\Rightarrow\) \(\Delta\) gives the number of unmatched

\(\Rightarrow\) \(t \in T\)

Let \(\Delta = (D/k + \varepsilon/2)\)

Return the set 
\(\{t : T \cap [t-\sigma...t+\sigma] \text{ is empty for at most } \Delta T\text{'s}\}\)
Approximating a Coordinated Matching Through Sampling

Analysis of COORD-MATCH()

2. \( t \in T \Rightarrow \{ I_{1..k} \} \) has a \((t, \sigma, E, D + \varepsilon k)\) co-matching

\((t, \sigma, E, D + \varepsilon k)\) not a co-matching

\( \Rightarrow \) More than \( D + \varepsilon k \) unmatched with a \( t_i \) s.t. \( |t_i - t| \leq \sigma \)

\( \Rightarrow \) at least \( \frac{D + \varepsilon k}{k} - \frac{\varepsilon}{2} \left( = \frac{D}{k} + \frac{\varepsilon}{2} \right) \) fraction are unmatched

\( \Rightarrow \Delta \) gives the number of unmatched, therefore at least \( \Delta \) are unmatched

\( \Rightarrow t \notin T \)

\( \{ t : T \cap [t-\sigma...t+\sigma] \text{ is empty for at most } \Delta T/s \} \)

Let \( \Delta = (D/k + \varepsilon/2)l \)

Return the set

With prob. at least \( 1 - 1/r^c \)
Approximating a Coordinated Matching Through Sampling
Analysis of COORD-MATCH()

- For both types of mistakes the probability of a mistake is at most $1/n^c$
- There are at most $n$ possible errors, which leads to an error probability of $1/n^{c-1}$
- That leaves us with a probability $1-1/n^{c-1}$
Correctness Proof (sketch)

\[ D(A,B) \leq n^\alpha \rightarrow \text{"CLOSE"} \]

- We can show this using a simple induction on the recursion level.
- The first level \((h=0)\) is in the last level of the recursion, when SHIFTS is called.
- Later we assume for \(h-1\) and solve for \(h\) through the relationship between matching and coordinated-matching.
Correctness Proof (sketch)

"CLOSE" $\Rightarrow D(A,B) < n/C$

- This is shown by bounding the mistake accumulated from the two point of degradation:
  - The SHIFTS at the lowest level of the recursion
  - The conversion between coordinated matches and matches along the recursion path
- By going up the recursion and accumulating the mistake we get a the value of the mistake as an inequality with the parameters $\epsilon$ and $c_1$ (and without $n$).
- Using this inequality it’s possible to adjust the parameters and bound the error as desired.
Running Time Analysis

The analysis is based on three cases:

- $\alpha < \frac{1}{2}$
- $\frac{1}{2} < \alpha < \frac{2}{3}$
- $\frac{2}{3} < \alpha$
Running Time Analysis

\( \alpha < 1/2 \)

In the first call \( k = \) to MATCHES: \( k = \min\{\varepsilon n^{1-\alpha}, 2c_1n^\alpha\} \)

\[ \alpha < \frac{1}{2} \Rightarrow n^{1-\alpha} > n^\alpha \Rightarrow k = 2c_1n^\alpha \]

As shown, in the next call \( E < 1 \Rightarrow \text{SHIFTS will be called} \)

\( \Rightarrow \) There are \( d \log n \) calls to SHIFTS, each with:

\[ O(\max\{\sqrt{u}, u/b, u/Q\}) \log n \Rightarrow O(n^{\alpha/2} \log n) \]

Before the return of \( \text{COORD - MATCHES} \) the is a single merge

with \( O(n^{\alpha/2} \log n) \)

Thus the total running time is \( O(n^{\alpha/2} \log^2 n) \)
Running Time Analysis

\( \frac{1}{2} < \alpha < \frac{2}{3} \)

First visit to MATCHES:

\[ E = n^\alpha, k = \min\{ \epsilon n^{1-\alpha}, 2c_i n^\alpha \} = \epsilon n^{1-\alpha} \]

Second visit to MATCHES:

\[ |I| = \frac{n^\alpha}{\epsilon}, E = \frac{c_i n^{2\alpha-1}}{\epsilon}, k = \min\{ \epsilon n^{1-\alpha}, \frac{c_i n^{2\alpha-1}}{\epsilon} \} = \frac{c_i n^{2\alpha-1}}{\epsilon} \]

\( \Rightarrow \) Next visit to MATCHES will call SHIFTS

\( \Rightarrow \) There are \( O(\log^2 n) \) calls to SHIFTS

As before, each call to SHIFTS costs \( O(n^{\alpha/2} \log n) \)

Thus the total running time is \( O(n^{\alpha/2} \log^3 n) \)
Running Time Analysis

\[ 2/3 < \alpha \]

The major source of time and query overhead is during the SHIFTS calls. Each invocation of SHIFTS has to find shifts in the range of \([-n^\alpha \ldots n^\alpha]\) of a block sized \(O(n^{1-\alpha})\).

\[ \alpha < 2/3 \Rightarrow n^{1-\alpha} < \sqrt[n^\alpha] \Rightarrow \text{The ruler has to be assymetric.} \]

Each invocation has a running time of \(O\left(\frac{u}{b} \log n\right) = O(n^{2\alpha-1} \log n)\). The other parts of the algorithm add a few more log s. Thus the total is \(\tilde{O}(n^{2\alpha-1})\).
Running Time Analysis

Summary

- We got a worst case of:
  \[ \tilde{O}(n^{2\alpha-1}) \]
- Not constant and not even logarithmic, but still sublinear.