Pattern Matching on Weighted Strings

Jakub Radoszewski

University of Warsaw, Poland

Prague Stringology Conference 2019

Plan of Presentation

1. Weighted Strings
2. Weighted Pattern Matching and Profile Matching
3. General Weighted Pattern Matching
4. Weighted Indexing
5. On-line and Streaming Weighted Pattern Matching
6. Weighted LCS and SCS

Plan of Presentation

1. Weighted Strings
2. Weighted Pattern Matching and Profile Matching
3. General Weighted Pattern Matching
4. Weighted Indexing
5. On-line and Streaming Weighted Pattern Matching
6. Weighted LCS and SCS

Strings, Partial Words, Indeterminate Strings

Strings (solid strings):

`a` `c` `a` `b` `b` `b`

Strings, Partial Words, Indeterminate Strings

Strings (solid strings):

$\mathtt{a} \quad \mathtt{c} \quad \mathtt{a} \quad \mathtt{b} \quad \mathtt{b} \quad \mathtt{b}$

Partial words (strings with don't care symbols):

$\mathtt{a} \quad \diamondsuit \quad \mathtt{a} \quad \mathtt{b} \quad \diamondsuit \quad \mathtt{b}$

Strings, Partial Words, Indeterminate Strings

Strings (solid strings):

a	c	a	b	b	b

Partial words (strings with don't care symbols):

a	$\diamondsuit$	a	b	$\diamondsuit$	b
a	c	a	b	b	b
a	a	a	b	a	b
a	b	a	b	c	b

$\vdots$

Strings, Partial Words, Indeterminate Strings

Strings (solid strings):

a	c	a	b	b	b

Partial words (strings with don't care symbols):

a	$\diamondsuit$	a	b	$\diamondsuit$	b
a	c	a	b	b	b
a	a	a	b	a	b
a	b	a	b	c	b

$\vdots$

Indeterminate strings:

a	b c	a	b	a b	b

Strings, Partial Words, Indeterminate Strings

Strings (solid strings):

a	c	a	b	b	b

Partial words (strings with don't care symbols):

a	$\diamondsuit$	a	b	$\diamondsuit$	b
a	c	a	b	b	b
a	a	a	b	a	b
a	b	a	b	c	b

$\vdots$

Indeterminate strings:

a	b c	a	b	a b	b
a	c	a	b	b	b
a	c	a	b	a	b
a	b	a	b	b	b
a	b	a	b	a	b

Weighted Strings (PPMs) and Profiles (PWMs)

Weighted Strings (Position Probability Matrices):

a	b 0.2 c 0.8	a	b	a 0.6 b 0.4	b

Weighted Strings (PPMs) and Profiles (PWMs)

Weighted Strings (Position Probability Matrices):

a	b 0.2 c 0.8	a	b	a 0.6 b 0.4	b	probability
a	c	a	b	b	b	0.32
a	c	a	b	a	b	0.48
a	b	a	b	b	b	0.08
a	b	a	b	a	b	0.12

Weighted Strings (PPMs) and Profiles (PWMs)

Weighted Strings (Position Probability Matrices):

a 1 b 0 c 0	a 0 b 0.2 c 0.8	a 1 b 0 c 0	a 0 b 1 c 0	a 0.6 b 0.4 c 0	a 0 b 1 c 0	probability
a	c	a	b	b	b	0.32
a	c	a	b	a	b	0.48
a	b	a	b	b	b	0.08
a	b	a	b	a	b	0.12

Weighted Strings (PPMs) and Profiles (PWMs)

Weighted Strings (Position Probability Matrices):

a 1	a 0	a 1	a 0	a 0.6	a 0	
b 0	b 0.2	b 0	b 1	b 0.4	b 1	
c 0	c 0.8	c 0	c 0	c 0	c 0	probability
a	c	a	b	b	b	0.32
a	c	a	b	a	b	0.48
a	b	a	b	b	b	0.08
a	b	a	b	a	b	0.12

Profiles (Position Weight Matrices):

a 7	a 3	a 0	a 0	a 6	a 1
b 0	b 2	b 1	b 5	b 4	b 9
c 1	c 8	c 0	c 0	c 3	c 0

Weighted Strings (PPMs) and Profiles (PWMs)

Weighted Strings (Position Probability Matrices):

a 1	a 0	a 1	a 0	a 0.6	a 0	
b 0	b 0.2	b 0	b 1	b 0.4	b 1	probability
c 0	c 0.8	c 0	c 0	c 0	c 0	
a	c	a	b	b	b	0.32
a	c	a	b	a	b	0.48
a	b	a	b	b	b	0.08
a	b	a	b	a	b	0.12

Profiles (Position Weight Matrices):

a 7	a 3	a 0	a 0	a 6	a 1	
b 0	b 2	b 1	b 5	b 4	b 9	score
c 1	c 8	c 0	c 0	c 3	c 0	
a	c	a	b	b	b	33
a	c	a	b	a	b	35
a	b	a	b	b	b	27
a	b	a	b	a	b	29

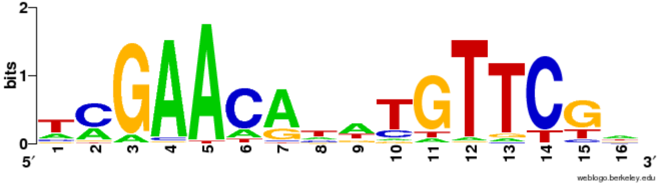
...

Applications of Uncertain Strings

Bioinformatics

- introduced in:

  Stormo, Schneider, Gold, and Ehrenfeucht (1982). "Use of the 'Perceptron' algorithm to distinguish translational initiation sites in E. coli". *Nucleic Acids Research* **10** (9): 2997–3011.
- one of the standard representations of motifs

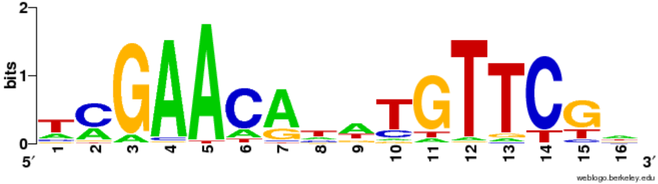


Source: Gnomehacker at English Wikipedia [GFDL, CC BY-SA 3.0] via Wikimedia Commons

Bioinformatics

- introduced in:

  Stormo, Schneider, Gold, and Ehrenfeucht (1982). “Use of the ‘Perceptron’ algorithm to distinguish translational initiation sites in E. coli”. *Nucleic Acids Research* **10** (9): 2997–3011.
- one of the standard representations of motifs



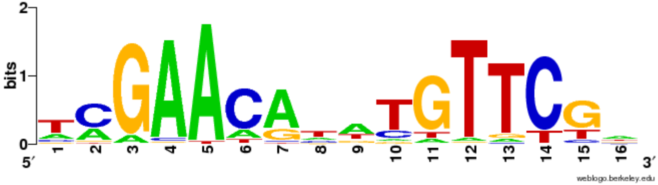
Source: Gnomehacker at English Wikipedia [GFDL, CC BY-SA 3.0] via Wikimedia Commons

- Single Nucleotide Polymorphisms, errors in genome sequencing. . .

Bioinformatics

- introduced in:

  Stormo, Schneider, Gold, and Ehrenfeucht (1982). "Use of the 'Perceptron' algorithm to distinguish translational initiation sites in E. coli". *Nucleic Acids Research* **10** (9): 2997–3011.
- one of the standard representations of motifs



Source: Gnomehacker at English Wikipedia [GFDL, CC BY-SA 3.0] via Wikimedia Commons

- Single Nucleotide Polymorphisms, errors in genome sequencing...
- $|\Sigma| = 4$ for DNA sequences
- $|\Sigma| = 20$ for protein sequences

Applications of Uncertain Strings

- Noisy sensor data, Probabilistic databases
  Measurement and sampling errors, resource limitations

Applications of Uncertain Strings

- Noisy sensor data, Probabilistic databases
  Measurement and sampling errors, resource limitations
- Privacy preserving
  Artificial uncertainty can be introduced to sanitize data but keep its utility

Applications of Uncertain Strings

- Noisy sensor data, Probabilistic databases
  Measurement and sampling errors, resource limitations
- Privacy preserving
  Artificial uncertainty can be introduced to sanitize data but keep its utility
- Missing parts of data
  Unknown parameters assumed to take any legal value equally likely

Threshold

- Score function and probability distribution are defined on all solid strings of matching length
- Typically, only high values are considered significant

Threshold

- Score function and probability distribution are defined on all solid strings of matching length
- Typically, only high values are considered significant

Definition

A string S matches a weighed string X if $\mathcal{P}(S, X) \geq \frac{1}{z}$ for a given threshold $\frac{1}{z}$. By $\mathcal{M}_z(X)$ we denote the set of all strings that match X for threshold z.

Threshold

- Score function and probability distribution are defined on all solid strings of matching length
- Typically, only high values are considered significant

Definition

A string S matches a weighed string X if $\mathcal{P}(S, X) \geq \frac{1}{z}$ for a given threshold $\frac{1}{z}$. By $\mathcal{M}_z(X)$ we denote the set of all strings that match X for threshold z.

Fact 1

$|\mathcal{M}_z(X)| \leq z$.

Threshold

- Score function and probability distribution are defined on all solid strings of matching length
- Typically, only high values are considered significant

Definition

A string S matches a weighed string X if $\mathcal{P}(S, X) \geq \frac{1}{z}$ for a given threshold $\frac{1}{z}$. By $\mathcal{M}_z(X)$ we denote the set of all strings that match X for threshold z.

Fact 1

$|\mathcal{M}_z(X)| \leq z$.

Proof. For every $S \in \mathcal{M}_z(X)$, we have $\mathcal{P}(S, X) \geq \frac{1}{z}$. Moreover, $\sum_{S \in \mathcal{M}_z(X)} \mathcal{P}(S, X) \leq 1$.

Threshold

- Score function and probability distribution are defined on all solid strings of matching length
- Typically, only high values are considered significant

Definition

A string S matches a weighed string X if $\mathcal{P}(S,X) \geq \frac{1}{z}$ for a given threshold $\frac{1}{z}$. By $\mathcal{M}_z(X)$ we denote the set of all strings that match X for threshold z.

Fact 1

$|\mathcal{M}_z(X)| \leq z$.

Proof. For every $S \in \mathcal{M}_z(X)$, we have $\mathcal{P}(S,X) \geq \frac{1}{z}$. Moreover, $\sum_{S \in \mathcal{M}_z(X)} \mathcal{P}(S,X) \leq 1$.

- z can be used as a parameter for designing algorithms

Plan of Presentation

1. Weighted Strings
2. Weighted Pattern Matching and Profile Matching
3. General Weighted Pattern Matching
4. Weighted Indexing
5. On-line and Streaming Weighted Pattern Matching
6. Weighted LCS and SCS

Weighted Pattern Matching

Input

- T – weighted string text of length n ($T[0, n-1]$), represented as an $n \times \sigma$ array
- P – string pattern of length m ($P[0, m-1]$)
- $\frac{1}{z}$ – threshold probability
- Σ – integer alphabet of size σ
- Model: probabilities can be multiplied in $\mathcal{O}(1)$ time

Output

All positions i in T where $\mathcal{P}(P, T[i, i+m-1]) \geq \frac{1}{z}$

Weighted Pattern Matching

Input

- T – weighted string text of length n ($T[0, n-1]$), represented as an $n \times \sigma$ array
- P – string pattern of length m ($P[0, m-1]$)
- $\frac{1}{z}$ – threshold probability
- Σ – integer alphabet of size σ
- Model: probabilities can be multiplied in $\mathcal{O}(1)$ time

Output

All positions i in T where $\mathcal{P}(P, T[i, i+m-1]) \geq \frac{1}{z}$

T	a $\frac{1}{4}$ b $\frac{3}{4}$	a 1 b 0	a $\frac{3}{4}$ b $\frac{1}{4}$	a $\frac{1}{2}$ b $\frac{1}{2}$	a 1 b 0
P	a	a	b		

$$z = 8$$

Weighted Pattern Matching

Input

- T – weighted string text of length n ($T[0, n-1]$), represented as an $n \times \sigma$ array
- P – string pattern of length m ($P[0, m-1]$)
- $\frac{1}{z}$ – threshold probability
- Σ – integer alphabet of size σ
- Model: probabilities can be multiplied in $\mathcal{O}(1)$ time

Output

All positions i in T where $\mathcal{P}(P, T[i, i+m-1]) \geq \frac{1}{z}$

T	a $\frac{1}{4}$	a 1	a $\frac{3}{4}$	a $\frac{1}{2}$	a 1
	b $\frac{3}{4}$	b 0	b $\frac{1}{4}$	b $\frac{1}{2}$	b 0
P	a	a	b	$\frac{1}{16} < \frac{1}{z}$ **NO**	

$z = 8$

Weighted Pattern Matching

Input

- T – weighted string text of length n ($T[0, n-1]$), represented as an $n \times \sigma$ array
- P – string pattern of length m ($P[0, m-1]$)
- $\frac{1}{z}$ – threshold probability
- Σ – integer alphabet of size σ
- Model: probabilities can be multiplied in $\mathcal{O}(1)$ time

Output

All positions i in T where $\mathcal{P}(P, T[i, i+m-1]) \geq \frac{1}{z}$

T	a $\frac{1}{4}$	a 1	a $\frac{3}{4}$	a $\frac{1}{2}$	a 1
	b $\frac{3}{4}$	b 0	b $\frac{1}{4}$	b $\frac{1}{2}$	b 0
P	a	a	b		

$z = 8$

Input

- T – weighted string text of length n ($T[0, n-1]$), represented as an $n \times \sigma$ array
- P – string pattern of length m ($P[0, m-1]$)
- $\frac{1}{z}$ – threshold probability
- Σ – integer alphabet of size σ
- Model: probabilities can be multiplied in $\mathcal{O}(1)$ time

Output

All positions i in T where $\mathcal{P}(P, T[i, i+m-1]) \geq \frac{1}{z}$

T	a $\frac{1}{4}$	a 1	a $\frac{3}{4}$	a $\frac{1}{2}$	a 1
	b $\frac{3}{4}$	b 0	b $\frac{1}{4}$	b $\frac{1}{2}$	b 0
P		a	a	b	

$z = 8$

Input

- T – weighted string text of length n ($T[0, n-1]$), represented as an $n \times \sigma$ array
- P – string pattern of length m ($P[0, m-1]$)
- $\frac{1}{z}$ – threshold probability
- Σ – integer alphabet of size σ
- Model: probabilities can be multiplied in $\mathcal{O}(1)$ time

Output

All positions i in T where $\mathcal{P}(P, T[i, i+m-1]) \geq \frac{1}{z}$

T	a $\frac{1}{4}$	a 1	a $\frac{3}{4}$	a $\frac{1}{2}$	a 1	
	b $\frac{3}{4}$	b 0	b $\frac{1}{4}$	b $\frac{1}{2}$	b 0	
P		a	a	b		$\frac{3}{8} \geq \frac{1}{z}$ **YES**

$z = 8$

Input

- T – weighted string text of length n ($T[0, n-1]$), represented as an $n \times \sigma$ array
- P – string pattern of length m ($P[0, m-1]$)
- $\frac{1}{z}$ – threshold probability
- Σ – integer alphabet of size σ
- Model: probabilities can be multiplied in $\mathcal{O}(1)$ time

Output

All positions i in T where $\mathcal{P}(P, T[i, i+m-1]) \geq \frac{1}{z}$

T	a $\frac{1}{4}$	a 1	a $\frac{3}{4}$	a $\frac{1}{2}$	a 1
	b $\frac{3}{4}$	b 0	b $\frac{1}{4}$	b $\frac{1}{2}$	b 0
P		a	a	b	

$z = 8$

Weighted Pattern Matching

Input

- T – weighted string text of length n ($T[0, n-1]$), represented as an $n \times \sigma$ array
- P – string pattern of length m ($P[0, m-1]$)
- $\frac{1}{z}$ – threshold probability
- Σ – integer alphabet of size σ
- Model: probabilities can be multiplied in $\mathcal{O}(1)$ time

Output

All positions i in T where $\mathcal{P}(P, T[i, i+m-1]) \geq \frac{1}{z}$

T	a $\frac{1}{4}$ b $\frac{3}{4}$	a 1 b 0	a $\frac{3}{4}$ b $\frac{1}{4}$	a $\frac{1}{2}$ b $\frac{1}{2}$	a 1 b 0
P			a	a	b

$z = 8$

Input

- T – weighted string text of length n ($T[0, n-1]$), represented as an $n \times \sigma$ array
- P – string pattern of length m ($P[0, m-1]$)
- $\frac{1}{z}$ – threshold probability
- Σ – integer alphabet of size σ
- Model: probabilities can be multiplied in $\mathcal{O}(1)$ time

Output

All positions i in T where $\mathcal{P}(P, T[i, i+m-1]) \geq \frac{1}{z}$

T	a $\frac{1}{4}$	a 1	a $\frac{3}{4}$	a $\frac{1}{2}$	a 1
	b $\frac{3}{4}$	b 0	b $\frac{1}{4}$	b $\frac{1}{2}$	b 0
P			a	a	b

$z = 8$

Input

- T – weighted string text of length n ($T[0, n-1]$), represented as an $n \times \sigma$ array
- P – string pattern of length m ($P[0, m-1]$)
- $\frac{1}{z}$ – threshold probability
- Σ – integer alphabet of size σ
- Model: probabilities can be multiplied in $\mathcal{O}(1)$ time

Output

All positions i in T where $\mathcal{P}(P, T[i, i+m-1]) \geq \frac{1}{z}$

T	a $\frac{1}{4}$	a 1	a $\frac{3}{4}$	a $\frac{1}{2}$	a 1	
	b $\frac{3}{4}$	b 0	b $\frac{1}{4}$	b $\frac{1}{2}$	b 0	
P			a	a	b	$0 < \frac{1}{z}$ **NO**

$z = 8$

Solutions to WPM

$\mathcal{O}(nm)$ time	naïve solution	
$\mathcal{O}(\sigma n \log m)$ time	σ times FFT	[CIMT'04]
$\mathcal{O}(n \log z)$ time	lookahead scoring and k-Mismatch	[Kociumaka-Pissis-R'16]

Solutions to WPM

$\mathcal{O}(nm)$ time	naïve solution	
$\mathcal{O}(\sigma n \log m)$ time	σ times FFT	[CIMT'04]
$\mathcal{O}(n \log z)$ time	lookahead scoring and k-Mismatch	[Kociumaka-Pissis-R'16]

WPM via Lookahead Scoring

X	a $\frac{1}{4}$	a 1	a $\frac{3}{4}$	a $\frac{1}{2}$	a 1
	b $\frac{3}{4}$	b 0	b $\frac{1}{4}$	b $\frac{1}{2}$	b 0

WPM via Lookahead Scoring

	X	a $\frac{1}{4}$	a 1	a $\frac{3}{4}$	a $\frac{1}{2}$	a 1
		b $\frac{3}{4}$	b 0	b $\frac{1}{4}$	b $\frac{1}{2}$	b 0
heavy string	**X**	b	a	a	a	a

WPM via Lookahead Scoring

	X	a $\frac{1}{4}$	a 1	a $\frac{3}{4}$	a $\frac{1}{2}$	a 1
		b $\frac{3}{4}$	b 0	b $\frac{1}{4}$	b $\frac{1}{2}$	b 0
heavy string	$\mathbf{X}$	b	a	a	a	a

- $d_H(S, T)$ – Hamming distance between strings S and T (the number of mismatches between S and T)

Fact 2

If $S \in \mathcal{M}_z(X)$ for string S and weighted string X and $\mathbf{X}$ is a heavy string of X, then $d_H(S, \mathbf{X}) \le \log_2 z$.

	X	a $\frac{1}{4}$	a 1	a $\frac{3}{4}$	a $\frac{1}{2}$	a 1
		b $\frac{3}{4}$	b 0	b $\frac{1}{4}$	b $\frac{1}{2}$	b 0
heavy string	$\mathbf{X}$	b	a	a	a	a

- $d_H(S, T)$ – Hamming distance between strings S and T (the number of mismatches between S and T)

Fact 2

If $S \in \mathcal{M}_z(X)$ for string S and weighted string X and $\mathbf{X}$ is a heavy string of X, then $d_H(S, \mathbf{X}) \leq \log_2 z$.

Proof. At each mismatch position between S and $\mathbf{X}$, the probability of the letter of S in X is ≤ 0.5.

WPM via Lookahead Scoring

	X	a $\frac{1}{4}$ b $\frac{3}{4}$	a 1 b 0	a $\frac{3}{4}$ b $\frac{1}{4}$	a $\frac{1}{2}$ b $\frac{1}{2}$	a 1 b 0
heavy string	**X**	b	a	a	a	a

- $d_H(S,T)$ – Hamming distance between strings S and T (the number of mismatches between S and T)

Fact 2

If $S \in \mathcal{M}_z(X)$ for string S and weighted string X and $\mathbf{X}$ is a heavy string of X, then $d_H(S, \mathbf{X}) \le \log_2 z$.

Proof. At each mismatch position between S and $\mathbf{X}$, the probability of the letter of S in X is ≤ 0.5.

- The heavy string method is also known as lookahead scoring

WPM via Lookahead Scoring

k-Mismatch Problem

For two strings P and T, find all positions where P matches T with at most k mismatches (and recover the mismatches).

WPM via Lookahead Scoring

k-Mismatch Problem

For two strings P and T, find all positions where P matches T with at most k mismatches (and recover the mismatches).

k-Mismatch can be solved in $\mathcal{O}(nk)$ time using kangaroo jumps:

1. Construct a data structure for answering *lcp*-queries for $T\#P$ ($\mathcal{O}(n+m)$ time via SA and RMQ)
2. For every position i in T, ask at most $k+1$ *lcp*-queries:

WPM via Lookahead Scoring

k-Mismatch Problem

For two strings P and T, find all positions where P matches T with at most k mismatches (and recover the mismatches).

k-Mismatch can be solved in $\mathcal{O}(nk)$ time using kangaroo jumps:

1. Construct a data structure for answering *lcp*-queries for $T\#P$ ($\mathcal{O}(n+m)$ time via SA and RMQ)
2. For every position i in T, ask at most $k+1$ *lcp*-queries:

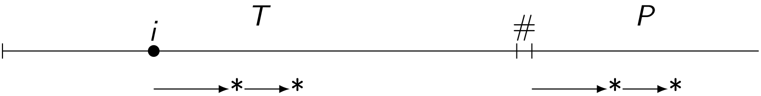
WPM via Lookahead Scoring

k-Mismatch Problem

For two strings P and T, find all positions where P matches T with at most k mismatches (and recover the mismatches).

k-Mismatch can be solved in $\mathcal{O}(nk)$ time using kangaroo jumps:

1. Construct a data structure for answering *lcp*-queries for $T\#P$ ($\mathcal{O}(n+m)$ time via SA and RMQ)
2. For every position i in T, ask at most $k+1$ *lcp*-queries:



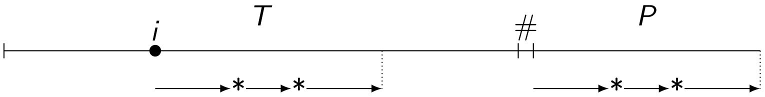
WPM via Lookahead Scoring

k-Mismatch Problem

For two strings P and T, find all positions where P matches T with at most k mismatches (and recover the mismatches).

k-Mismatch can be solved in $\mathcal{O}(nk)$ time using kangaroo jumps:

1. Construct a data structure for answering *lcp*-queries for $T\#P$ ($\mathcal{O}(n+m)$ time via SA and RMQ)
2. For every position i in T, ask at most $k+1$ *lcp*-queries:



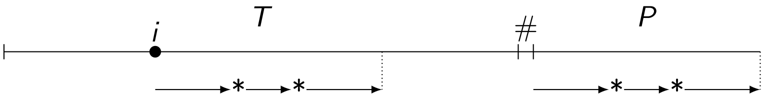
WPM via Lookahead Scoring

k-Mismatch Problem

For two strings P and T, find all positions where P matches T with at most k mismatches (and recover the mismatches).

k-Mismatch can be solved in $\mathcal{O}(nk)$ time using kangaroo jumps:

1. Construct a data structure for answering *lcp*-queries for $T\#P$ ($\mathcal{O}(n+m)$ time via SA and RMQ)
2. For every position i in T, ask at most $k+1$ *lcp*-queries:



Break after reaching the end of the string or after the $(k+1)$th mismatch

WPM via Lookahead Scoring

1. Compute $\alpha := \mathcal{P}(\mathbf{T}[0, m-1], T[0, m-1])$
2. $k := \log_2 z$
3. For every position $i := 0$ to $n - m$ do:
   1. If $\mathbf{T}[i, i+m-1]$ and P have at most k mismatches, let A be the set of their positions:
      1. $\alpha' := \alpha$
      2. For every $j \in A$,
         $\alpha' := \alpha' \cdot \mathcal{P}(\mathbf{T}[i+j], T[i+j]) / \mathcal{P}(P[j], T[i+j])$
      3. If $\alpha' \geq \frac{1}{z}$, return a match at position i
   2. $\alpha := \alpha \cdot \mathcal{P}(\mathbf{T}[i+m], T[i+m]) / \mathcal{P}(\mathbf{T}[i], T[i])$

WPM via Lookahead Scoring

1. Compute $\alpha := \mathcal{P}(\mathbf{T}[0, m-1], T[0, m-1])$
2. $k := \log_2 z$
3. For every position $i := 0$ to $n - m$ do:
   1. If $\mathbf{T}[i, i+m-1]$ and P have at most k mismatches, let A be the set of their positions:
      1. $\alpha' := \alpha$
      2. For every $j \in A$,
         $\alpha' := \alpha' \cdot \mathcal{P}(\mathbf{T}[i+j], T[i+j]) / \mathcal{P}(P[j], T[i+j])$
      3. If $\alpha' \geq \frac{1}{z}$, return a match at position i
   2. $\alpha := \alpha \cdot \mathcal{P}(\mathbf{T}[i+m], T[i+m]) / \mathcal{P}(\mathbf{T}[i], T[i])$

	T	a $\frac{1}{4}$	a 1	a $\frac{3}{4}$	a $\frac{1}{2}$	a 1
		b $\frac{3}{4}$	b 0	b $\frac{1}{4}$	b $\frac{1}{2}$	b 0
heavy string	$\mathbf{T}$	b	a	a	a	a

$z = 8$

WPM via Lookahead Scoring

1. Compute $\alpha := \mathcal{P}(\mathbf{T}[0, m-1], T[0, m-1])$
2. $k := \log_2 z$
3. For every position $i := 0$ to $n - m$ do:
   1. If $\mathbf{T}[i, i+m-1]$ and P have at most k mismatches, let A be the set of their positions:
      1. $\alpha' := \alpha$
      2. For every $j \in A$,
         $\alpha' := \alpha' \cdot \mathcal{P}(\mathbf{T}[i+j], T[i+j]) / \mathcal{P}(P[j], T[i+j])$
      3. If $\alpha' \geq \frac{1}{z}$, return a match at position i
   2. $\alpha := \alpha \cdot \mathcal{P}(\mathbf{T}[i+m], T[i+m]) / \mathcal{P}(\mathbf{T}[i], T[i])$

	T	a $\frac{1}{4}$	a 1	a $\frac{3}{4}$	a $\frac{1}{2}$	a 1
		b $\frac{3}{4}$	b 0	b $\frac{1}{4}$	b $\frac{1}{2}$	b 0
heavy string	$\mathbf{T}$	b	a	a	a	a
	$z = 8$	a	a	b		

WPM via Lookahead Scoring

1. Compute $\alpha := \mathcal{P}(\mathbf{T}[0, m-1], T[0, m-1])$
2. $k := \log_2 z$
3. For every position $i := 0$ to $n - m$ do:
   1. If $\mathbf{T}[i, i+m-1]$ and P have at most k mismatches, let A be the set of their positions:
      1. $\alpha' := \alpha$
      2. For every $j \in A$,
         $\alpha' := \alpha' \cdot \mathcal{P}(\mathbf{T}[i+j], T[i+j]) / \mathcal{P}(P[j], T[i+j])$
      3. If $\alpha' \geq \frac{1}{z}$, return a match at position i
   2. $\alpha := \alpha \cdot \mathcal{P}(\mathbf{T}[i+m], T[i+m]) / \mathcal{P}(\mathbf{T}[i], T[i])$

	T	a $\frac{1}{4}$	a 1	a $\frac{3}{4}$	a $\frac{1}{2}$	a 1
		b $\frac{3}{4}$	b 0	b $\frac{1}{4}$	b $\frac{1}{2}$	b 0
heavy string	$\mathbf{T}$	b	a	a	$\frac{9}{16}$	
	$z = 8$	a	a	b		

WPM via Lookahead Scoring

1. Compute $\alpha := \mathcal{P}(\mathbf{T}[0, m-1], T[0, m-1])$
2. $k := \log_2 z$
3. For every position $i := 0$ to $n - m$ do:
   1. If $\mathbf{T}[i, i+m-1]$ and P have at most k mismatches, let A be the set of their positions:
      1. $\alpha' := \alpha$
      2. For every $j \in A$,
         $\alpha' := \alpha' \cdot \mathcal{P}(\mathbf{T}[i+j], T[i+j]) / \mathcal{P}(P[j], T[i+j])$
      3. If $\alpha' \geq \frac{1}{z}$, return a match at position i
   2. $\alpha := \alpha \cdot \mathcal{P}(\mathbf{T}[i+m], T[i+m]) / \mathcal{P}(\mathbf{T}[i], T[i])$

	T	a $\frac{1}{4}$	a 1	a $\frac{3}{4}$	a $\frac{1}{2}$	a 1
		b $\frac{3}{4}$	b 0	b $\frac{1}{4}$	b $\frac{1}{2}$	b 0
heavy string	$\mathbf{T}$	b	a	a	$\frac{9}{16}$	
		$\neq$	$=$	$\neq$		
$z = 8$		a	a	b		

WPM via Lookahead Scoring

1. Compute $\alpha := \mathcal{P}(\mathbf{T}[0, m-1], T[0, m-1])$
2. $k := \log_2 z$
3. For every position $i := 0$ to $n - m$ do:
    1. If $\mathbf{T}[i, i+m-1]$ and P have at most k mismatches, let A be the set of their positions:
        1. $\alpha' := \alpha$
        2. For every $j \in A$, $\alpha' := \alpha' \cdot \mathcal{P}(\mathbf{T}[i+j], T[i+j]) / \mathcal{P}(P[j], T[i+j])$
        3. If $\alpha' \geq \frac{1}{z}$, return a match at position i
    2. $\alpha := \alpha \cdot \mathcal{P}(\mathbf{T}[i+m], T[i+m]) / \mathcal{P}(\mathbf{T}[i], T[i])$

	T	a $\frac{1}{4}$	a 1	a $\frac{3}{4}$	a $\frac{1}{2}$	a 1
		b $\frac{3}{4}$	b 0	b $\frac{1}{4}$	b $\frac{1}{2}$	b 0
heavy string	$\mathbf{T}$	b	a	a	$\frac{9}{16}$	
		$\nparallel$	$\parallel$	$\nparallel$		
$z = 8$		a	a	b	$\frac{1}{16} < \frac{1}{z}$ **NO**	

WPM via Lookahead Scoring

1. Compute $\alpha := \mathcal{P}(\mathbf{T}[0, m-1], T[0, m-1])$
2. $k := \log_2 z$
3. For every position $i := 0$ to $n - m$ do:
   1. If $\mathbf{T}[i, i+m-1]$ and P have at most k mismatches, let A be the set of their positions:
      1. $\alpha' := \alpha$
      2. For every $j \in A$,
         $\alpha' := \alpha' \cdot \mathcal{P}(\mathbf{T}[i+j], T[i+j]) / \mathcal{P}(P[j], T[i+j])$
      3. If $\alpha' \geq \frac{1}{z}$, return a match at position i
   2. $\alpha := \alpha \cdot \mathcal{P}(\mathbf{T}[i+m], T[i+m]) / \mathcal{P}(\mathbf{T}[i], T[i])$

	T	a $\frac{1}{4}$ b $\frac{3}{4}$	a 1 b 0	a $\frac{3}{4}$ b $\frac{1}{4}$	a $\frac{1}{2}$ b $\frac{1}{2}$	a 1 b 0
heavy string	$\mathbf{T}$	b	a	a	$\frac{9}{16}$	
	$z = 8$		a	a	b	

1. Compute $\alpha := \mathcal{P}(\mathbf{T}[0, m-1], T[0, m-1])$
2. $k := \log_2 z$
3. For every position $i := 0$ to $n - m$ do:
    1. If $\mathbf{T}[i, i+m-1]$ and P have at most k mismatches, let A be the set of their positions:
        1. $\alpha' := \alpha$
        2. For every $j \in A$, $\alpha' := \alpha' \cdot \mathcal{P}(\mathbf{T}[i+j], T[i+j]) / \mathcal{P}(P[j], T[i+j])$
        3. If $\alpha' \geq \frac{1}{z}$, return a match at position i
    2. $\alpha := \alpha \cdot \mathcal{P}(\mathbf{T}[i+m], T[i+m]) / \mathcal{P}(\mathbf{T}[i], T[i])$

	T	a $\frac{1}{4}$ b $\frac{3}{4}$	a 1 b 0	a $\frac{3}{4}$ b $\frac{1}{4}$	a $\frac{1}{2}$ b $\frac{1}{2}$	a 1 b 0
heavy string	$\mathbf{T}$		a	a	a	$\frac{3}{8}$
$z = 8$			a	a	b	

WPM via Lookahead Scoring

1. Compute $\alpha := \mathcal{P}(\mathbf{T}[0, m-1], T[0, m-1])$
2. $k := \log_2 z$
3. For every position $i := 0$ to $n - m$ do:
   1. If $\mathbf{T}[i, i+m-1]$ and P have at most k mismatches, let A be the set of their positions:
      1. $\alpha' := \alpha$
      2. For every $j \in A$,
         $\alpha' := \alpha' \cdot \mathcal{P}(\mathbf{T}[i+j], T[i+j]) / \mathcal{P}(P[j], T[i+j])$
      3. If $\alpha' \geq \frac{1}{z}$, return a match at position i
   2. $\alpha := \alpha \cdot \mathcal{P}(\mathbf{T}[i+m], T[i+m]) / \mathcal{P}(\mathbf{T}[i], T[i])$

	T	a $\frac{1}{4}$ b $\frac{3}{4}$	a 1 b 0	a $\frac{3}{4}$ b $\frac{1}{4}$	a $\frac{1}{2}$ b $\frac{1}{2}$	a 1 b 0
heavy string	$\mathbf{T}$		a	a	a	$\frac{3}{8}$
			$\parallel$	$\parallel$	$\nparallel$	
	$z = 8$		a	a	b	

WPM via Lookahead Scoring

1. Compute $\alpha := \mathcal{P}(\mathbf{T}[0, m-1], T[0, m-1])$
2. $k := \log_2 z$
3. For every position $i := 0$ to $n - m$ do:
    1. If $\mathbf{T}[i, i+m-1]$ and P have at most k mismatches, let A be the set of their positions:
        1. $\alpha' := \alpha$
        2. For every $j \in A$, $\alpha' := \alpha' \cdot \mathcal{P}(\mathbf{T}[i+j], T[i+j]) / \mathcal{P}(P[j], T[i+j])$
        3. If $\alpha' \geq \frac{1}{z}$, return a match at position i
    2. $\alpha := \alpha \cdot \mathcal{P}(\mathbf{T}[i+m], T[i+m]) / \mathcal{P}(\mathbf{T}[i], T[i])$

T	a $\frac{1}{4}$	a 1	a $\frac{3}{4}$	a $\frac{1}{2}$	a 1
	b $\frac{3}{4}$	b 0	b $\frac{1}{4}$	b $\frac{1}{2}$	b 0
heavy string $\mathbf{T}$		a	a	a	$\frac{3}{8}$
		$\vert\vert$	$\vert\vert$	$\not\vert\vert$	
$z = 8$		a	a	b	$\frac{3}{8} \geq \frac{1}{z}$ **YES**

Application: Profile Matching

Input

- T – string text of length n
- P – pattern being an $m \times \sigma$ profile
- Z – threshold
- σ – alphabet size

Output

All positions i in T where $T[i, i+m-1]$ matches P with score at least Z

Application: Profile Matching

Input

- T – string text of length n
- P – pattern being an $m \times \sigma$ profile
- Z – threshold
- σ – alphabet size

Output

All positions i in T where $T[i, i+m-1]$ matches P with score at least Z

$\mathcal{O}(nm)$ time	naïve solution with heuristics	see [Pizzi-Ukkonen'08]
$\mathcal{O}(\sigma n \log m)$ time	σ times FFT	[Rajasekaran-Jin-Spouge'02]
$\mathcal{O}(n \log \lvert\mathcal{M}_Z(P)\rvert)$ time	lookahead scoring and k-Mismatch	[Kociumaka-Pissis-R'16]

Input

- T – string text of length n
- P – pattern being an $m \times \sigma$ profile
- Z – threshold
- σ – alphabet size

Output

All positions i in T where $T[i, i+m-1]$ matches P with score at least Z

$\mathcal{O}(nm)$ time	naïve solution with heuristics	see [Pizzi-Ukkonen'08]
$\mathcal{O}(\sigma n \log m)$ time	σ times FFT	[Rajasekaran-Jin-Spouge'02]
$\mathcal{O}(n \log \lvert\mathcal{M}_Z(P)\rvert)$ time	lookahead scoring and k-Mismatch	[Kociumaka-Pissis-R'16]

Application: Profile Matching

- $\mathcal{M}_Z(P)$ – the set of strings that match the profile P with score above Z

Application: Profile Matching

- $\mathcal{M}_Z(P)$ – the set of strings that match the profile P with score above Z
- Obviously, $|\mathcal{M}_Z(P)| \leq \sigma^m$, i.e. $\mathcal{O}(n \log |\mathcal{M}_Z(P)|) = \mathcal{O}(nm \log \sigma)$

Application: Profile Matching

- $\mathcal{M}_Z(P)$ – the set of strings that match the profile P with score above Z
- Obviously, $|\mathcal{M}_Z(P)| \leq \sigma^m$, i.e. $\mathcal{O}(n \log |\mathcal{M}_Z(P)|) = \mathcal{O}(nm \log \sigma)$
- However, in practice it is expected to be much smaller

Application: Profile Matching

- $\mathcal{M}_Z(P)$ – the set of strings that match the profile P with score above Z
- Obviously, $|\mathcal{M}_Z(P)| \leq \sigma^m$, i.e. $\mathcal{O}(n \log |\mathcal{M}_Z(P)|) = \mathcal{O}(nm \log \sigma)$
- However, in practice it is expected to be much smaller

Fact

If a string S matches a profile P with score at least Z and $d_H(S, \mathbf{P}) = k$, then $|\mathcal{M}_Z(P)| \geq 2^k$.

Application: Profile Matching

- $\mathcal{M}_Z(P)$ – the set of strings that match the profile P with score above Z
- Obviously, $|\mathcal{M}_Z(P)| \leq \sigma^m$, i.e. $\mathcal{O}(n \log |\mathcal{M}_Z(P)|) = \mathcal{O}(nm \log \sigma)$
- However, in practice it is expected to be much smaller

Fact

If a string S matches a profile P with score at least Z and $d_H(S, \mathbf{P}) = k$, then $|\mathcal{M}_Z(P)| \geq 2^k$.

Hence, $k \leq \log |\mathcal{M}_Z(P)|$.

Application: Profile Matching

- $\mathcal{M}_Z(P)$ – the set of strings that match the profile P with score above Z
- Obviously, $|\mathcal{M}_Z(P)| \leq \sigma^m$, i.e. $\mathcal{O}(n \log |\mathcal{M}_Z(P)|) = \mathcal{O}(nm \log \sigma)$
- However, in practice it is expected to be much smaller

Fact

If a string S matches a profile P with score at least Z and $d_H(S, \mathbf{P}) = k$, then $|\mathcal{M}_Z(P)| \geq 2^k$.

Hence, $k \leq \log |\mathcal{M}_Z(P)|$.

Lookahead scoring:

- Matching heavy string $\mathbf{P}$ in the text T allowing mismatches – kangaroo jumps
- Start at next position if the score drops below Z

Plan of Presentation

1. Weighted Strings
2. Weighted Pattern Matching and Profile Matching
3. General Weighted Pattern Matching
4. Weighted Indexing
5. On-line and Streaming Weighted Pattern Matching
6. Weighted LCS and SCS

Three variants of WPM

SPWT (WPM)

Input: T – **weighted** string text, P – string pattern
Output:
All positions i in T where $\mathcal{P}(P, T[i, i+m-1]) \geq \frac{1}{z}$

WPST

Input: T – string text, P – **weighted** string pattern
Output:
All positions i in T where $\mathcal{P}(T[i, i+m-1], P) \geq \frac{1}{z}$

WPWT (General WPM)

Input: T – **weighted** string text, P – **weighted** string pattern
Output:
All positions i in T for which there exists a string S such that $\mathcal{P}(S, P) \geq \frac{1}{z}$ and $\mathcal{P}(S, T[i, i+m-1]) \geq \frac{1}{z}$

Three variants of WPM

SPWT (WPM)

Input: T – **weighted** string text, P – string pattern
Output:
All positions i in T where $\mathcal{P}(P, T[i, i+m-1]) \geq \frac{1}{z}$

WPST

Input: T – string text, P – **weighted** string pattern
Output:
All positions i in T where $\mathcal{P}(T[i, i+m-1], P) \geq \frac{1}{z}$

Off-line solutions for WPST and SPWT are the same

WPWT (General WPM)

Input: T – **weighted** string text, P – **weighted** string pattern
Output:
All positions i in T for which there exists a string S such that
$\mathcal{P}(S, P) \geq \frac{1}{z}$ and $\mathcal{P}(S, T[i, i+m-1]) \geq \frac{1}{z}$

General WPM

T	a $\frac{1}{8}$	a $\frac{1}{2}$	a $\frac{1}{2}$	a 1	a 0	a $\frac{1}{2}$	a $\frac{1}{2}$	$\frac{1}{8}$
	b $\frac{7}{8}$	b $\frac{1}{2}$	b $\frac{1}{2}$	b 0	b 1	b $\frac{1}{2}$	b $\frac{1}{2}$	
P		a $\frac{1}{4}$	a 1	a $\frac{3}{4}$	a $\frac{1}{2}$	a 1		$\frac{9}{32}$
		b $\frac{3}{4}$	b 0	b $\frac{1}{4}$	b $\frac{1}{2}$	b 0		

$$z = 8$$

General WPM

T	a $\frac{1}{8}$	a $\frac{1}{2}$	a $\frac{1}{2}$	a 1	a 0	a $\frac{1}{2}$	a $\frac{1}{2}$	$\frac{1}{8}$
	b $\frac{7}{8}$	b $\frac{1}{2}$	b $\frac{1}{2}$	b 0	b 1	b $\frac{1}{2}$	b $\frac{1}{2}$	
P		a $\frac{1}{4}$	a 1	a $\frac{3}{4}$	a $\frac{1}{2}$	a 1		$\frac{9}{32}$
		b $\frac{3}{4}$	b 0	b $\frac{1}{4}$	b $\frac{1}{2}$	b 0		

$$z = 8$$

- λ – the maximum number of letters with probability $\geq \frac{1}{z}$ at one position ($\lambda \leq \min(z, \sigma)$)

$\mathcal{O}(nz^2 \log z)$ time	[Barton-Liu-Pissis'15]
$\mathcal{O}(n\sqrt{z\lambda}(\log \log z + \log \lambda))$ time	[Kociumaka-Pissis-R'16]

General WPM

T	a $\frac{1}{8}$	a $\frac{1}{2}$	a $\frac{1}{2}$	a 1	a 0	a $\frac{1}{2}$	a $\frac{1}{2}$	$\frac{1}{8}$
	b $\frac{7}{8}$	b $\frac{1}{2}$	b $\frac{1}{2}$	b 0	b 1	b $\frac{1}{2}$	b $\frac{1}{2}$	
P		a $\frac{1}{4}$	a 1	a $\frac{3}{4}$	a $\frac{1}{2}$	a 1		$\frac{9}{32}$
		b $\frac{3}{4}$	b 0	b $\frac{1}{4}$	b $\frac{1}{2}$	b 0		

$$z = 8$$

- λ – the maximum number of letters with probability $\geq \frac{1}{z}$ at one position ($\lambda \leq \min(z, \sigma)$)

$\mathcal{O}(nz^2 \log z)$ time	[Barton-Liu-Pissis'15]
$\mathcal{O}(n\sqrt{z\lambda}(\log\log z + \log\lambda))$ time	[Kociumaka-Pissis-R'16]
$\mathcal{O}(n\sqrt{z}\log^2 z)$ time for $\sigma = \mathcal{O}(1)$	this presentation

Weighted Consensus

- Weighted Consensus Problem (WCP): $n = m$
- General WPM reduces to $n - m + 1$ instances of WCP of size m

Weighted Consensus

- Weighted Consensus Problem (WCP): $n = m$
- General WPM reduces to $n - m + 1$ instances of WCP of size m

Fact 3 (from Fact 2)

If $\mathcal{M}_z(X) \cap \mathcal{M}_z(Y) \neq \emptyset$ for weighted strings X, Y and $\mathbf{X}$, $\mathbf{Y}$ are heavy strings of X and Y, resp., then $d_H(\mathbf{X}, \mathbf{Y}) \leq 2\log_2 z$.

Weighted Consensus

- Weighted Consensus Problem (WCP): $n = m$
- General WPM reduces to $n - m + 1$ instances of WCP of size m

Fact 3 (from Fact 2)

If $\mathcal{M}_z(X) \cap \mathcal{M}_z(Y) \neq \emptyset$ for weighted strings X, Y and $\mathbf{X}$, $\mathbf{Y}$ are heavy strings of X and Y, resp., then $d_H(\mathbf{X}, \mathbf{Y}) \leq 2 \log_2 z$.

Proof. If $S \in \mathcal{M}_z(X) \cap \mathcal{M}_z(Y)$, then $d_H(S, \mathbf{X})$, $d_H(S, \mathbf{Y}) \leq \log_2 z$.

Weighted Consensus

- Weighted Consensus Problem (WCP): $n = m$
- General WPM reduces to $n - m + 1$ instances of WCP of size m

Fact 3 (from Fact 2)

If $\mathcal{M}_z(X) \cap \mathcal{M}_z(Y) \neq \emptyset$ for weighted strings X, Y and $\mathbf{X}$, $\mathbf{Y}$ are heavy strings of X and Y, resp., then $d_H(\mathbf{X}, \mathbf{Y}) \leq 2 \log_2 z$.

Proof. If $S \in \mathcal{M}_z(X) \cap \mathcal{M}_z(Y)$, then $d_H(S, \mathbf{X})$, $d_H(S, \mathbf{Y}) \leq \log_2 z$.

Moreover, there is a string $S \in \mathcal{M}_z(X) \cap \mathcal{M}_z(Y)$ such that $S[i] = \mathbf{X}[i] = \mathbf{Y}[i]$ unless $\mathbf{X}[i] \neq \mathbf{Y}[i]$.

Weighted Consensus

- Weighted Consensus Problem (WCP): $n = m$
- General WPM reduces to $n - m + 1$ instances of WCP of size m

Fact 3 (from Fact 2)

If $\mathcal{M}_z(X) \cap \mathcal{M}_z(Y) \neq \emptyset$ for weighted strings X, Y and $\mathbf{X}$, $\mathbf{Y}$ are heavy strings of X and Y, resp., then $d_H(\mathbf{X}, \mathbf{Y}) \leq 2\log_2 z$.

Proof. If $S \in \mathcal{M}_z(X) \cap \mathcal{M}_z(Y)$, then $d_H(S, \mathbf{X})$, $d_H(S, \mathbf{Y}) \leq \log_2 z$.

Moreover, there is a string $S \in \mathcal{M}_z(X) \cap \mathcal{M}_z(Y)$ such that $S[i] = \mathbf{X}[i] = \mathbf{Y}[i]$ unless $\mathbf{X}[i] \neq \mathbf{Y}[i]$.

- Using k-Mismatch for $k = 2\log_2 z$, in $\mathcal{O}(n \log z)$ time General WPM reduces to $n - m + 1$ instances of WCP of size $\mathcal{O}(\log z)$

Weighted Consensus

Fact 4

If $S \in \mathcal{M}_z(X)$ for a string S and weighted sequence X of length n, then there exists a position i such that $\mathcal{P}(S[0, i-1], X[0, i-1])$, $\mathcal{P}(S[i+1, n-1], X[i+1, n-1]) \geq \frac{1}{\sqrt{z}}$.

Weighted Consensus

Fact 4

If $S \in \mathcal{M}_z(X)$ for a string S and weighted sequence X of length n, then there exists a position i such that
$\mathcal{P}(S[0, i-1], X[0, i-1])$, $\mathcal{P}(S[i+1, n-1], X[i+1, n-1]) \geq \frac{1}{\sqrt{z}}$.

Proof.

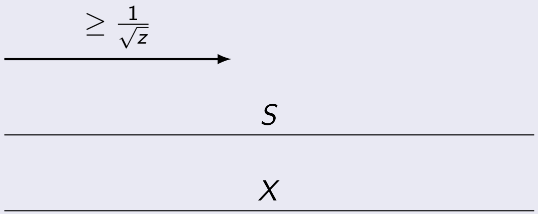
S

X

Fact 4

If $S \in \mathcal{M}_z(X)$ for a string S and weighted sequence X of length n, then there exists a position i such that
$\mathcal{P}(S[0, i-1], X[0, i-1]),\ \mathcal{P}(S[i+1, n-1], X[i+1, n-1]) \geq \frac{1}{\sqrt{z}}$.

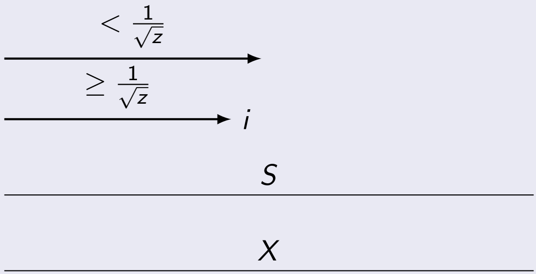
Proof.



Fact 4

If $S \in \mathcal{M}_z(X)$ for a string S and weighted sequence X of length n, then there exists a position i such that $\mathcal{P}(S[0, i-1], X[0, i-1]), \mathcal{P}(S[i+1, n-1], X[i+1, n-1]) \geq \frac{1}{\sqrt{z}}$.

Proof.

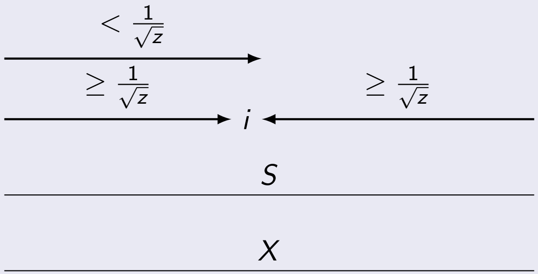


Weighted Consensus

Fact 4

If $S \in \mathcal{M}_z(X)$ for a string S and weighted sequence X of length n, then there exists a position i such that $\mathcal{P}(S[0,i-1],X[0,i-1])$, $\mathcal{P}(S[i+1,n-1],X[i+1,n-1]) \geq \frac{1}{\sqrt{z}}$.

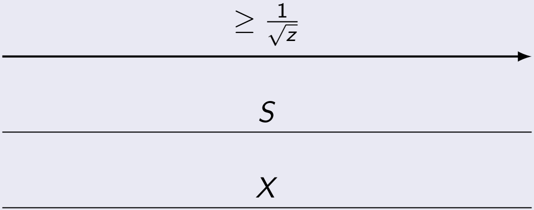
Proof.



Fact 4

If $S \in \mathcal{M}_z(X)$ for a string S and weighted sequence X of length n, then there exists a position i such that $\mathcal{P}(S[0, i-1], X[0, i-1])$, $\mathcal{P}(S[i+1, n-1], X[i+1, n-1]) \geq \frac{1}{\sqrt{z}}$.

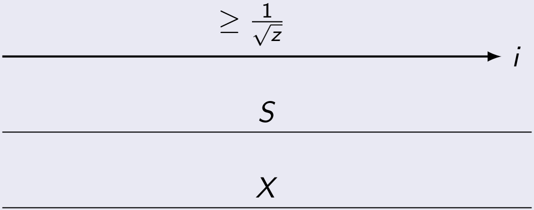
Proof.



Fact 4

If $S \in \mathcal{M}_z(X)$ for a string S and weighted sequence X of length n, then there exists a position i such that $\mathcal{P}(S[0, i-1], X[0, i-1])$, $\mathcal{P}(S[i+1, n-1], X[i+1, n-1]) \geq \frac{1}{\sqrt{z}}$.

Proof.

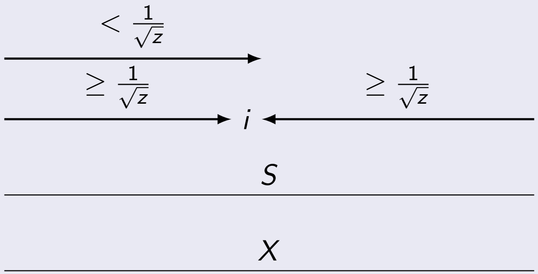


Weighted Consensus

Fact 4

If $S \in \mathcal{M}_z(X)$ for a string S and weighted sequence X of length n, then there exists a position i such that $\mathcal{P}(S[0, i-1], X[0, i-1])$, $\mathcal{P}(S[i+1, n-1], X[i+1, n-1]) \geq \frac{1}{\sqrt{z}}$.

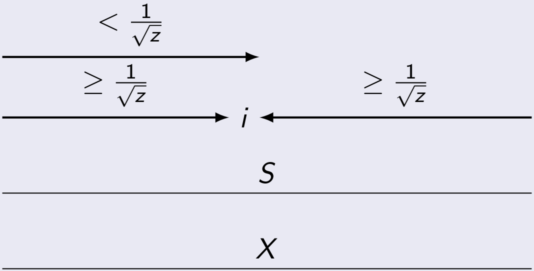
Proof.



Fact 4

If $S \in \mathcal{M}_z(X)$ for a string S and weighted sequence X of length n, then there exists a position i such that
$\mathcal{P}(S[0, i-1], X[0, i-1]), \mathcal{P}(S[i+1, n-1], X[i+1, n-1]) \geq \frac{1}{\sqrt{z}}$.

Proof.



- By Fact 1, $|\mathcal{M}_{\sqrt{z}}(X)| \leq \sqrt{z}$ for a weighted sequence X

Weighted Consensus

Meet-in-the-middle:

1. For $i = 0, \ldots, n-1$ in order, compute $\mathcal{M}_{\sqrt{z}}(T[0, i])$
2. For $i = n, \ldots, 0$ in order, compute $\mathcal{M}_{\sqrt{z}}(T[i, n-1])$
3. // $\mathcal{O}(\sqrt{z} \log^2 z)$ time since $n = \mathcal{O}(\log z)$

Weighted Consensus

Meet-in-the-middle:

1. For $i = 0, \ldots, n-1$ in order, compute $\mathcal{M}_{\sqrt{z}}(T[0,i])$
2. For $i = n, \ldots, 0$ in order, compute $\mathcal{M}_{\sqrt{z}}(T[i,n-1])$
3. // $\mathcal{O}(\sqrt{z}\log^2 z)$ time since $n = \mathcal{O}(\log z)$
4. For every $i = 0, \ldots, n-1$:
   1. try to join every $U \in \mathcal{M}_{\sqrt{z}}(T[0,i-1])$ and $V \in \mathcal{M}_{\sqrt{z}}(T[i+1,n-1])$ with a letter $c \in \Sigma$ at position i:

Weighted Consensus

Meet-in-the-middle:

1. For $i = 0, \ldots, n-1$ in order, compute $\mathcal{M}_{\sqrt{z}}(T[0,i])$
2. For $i = n, \ldots, 0$ in order, compute $\mathcal{M}_{\sqrt{z}}(T[i,n-1])$
3. // $\mathcal{O}(\sqrt{z}\log^2 z)$ time since $n = \mathcal{O}(\log z)$
4. For every $i = 0, \ldots, n-1$:
   1. try to join every $U \in \mathcal{M}_{\sqrt{z}}(T[0,i-1])$ and $V \in \mathcal{M}_{\sqrt{z}}(T[i+1,n-1])$ with a letter $c \in \Sigma$ at position i:
      - $\mathcal{P}(Uc, P[0,i]), \mathcal{P}(Uc, T[0,i]) \geq \frac{1}{z}$
      - $\mathcal{P}(V, P[i,n-1]) \geq \frac{1}{z}$

Weighted Consensus

Meet-in-the-middle:

1. For $i = 0, \ldots, n-1$ in order, compute $\mathcal{M}_{\sqrt{z}}(T[0,i])$
2. For $i = n, \ldots, 0$ in order, compute $\mathcal{M}_{\sqrt{z}}(T[i,n-1])$
3. // $\mathcal{O}(\sqrt{z}\log^2 z)$ time since $n = \mathcal{O}(\log z)$
4. For every $i = 0, \ldots, n-1$:
   1. try to join every $U \in \mathcal{M}_{\sqrt{z}}(T[0,i-1])$ and $V \in \mathcal{M}_{\sqrt{z}}(T[i+1,n-1])$ with a letter $c \in \Sigma$ at position i:
      - $\mathcal{P}(Uc, P[0,i]), \mathcal{P}(Uc, T[0,i]) \geq \frac{1}{z}$
      - $\mathcal{P}(V, P[i,n-1]) \geq \frac{1}{z}$
   2. Auxiliary problem: Given two sets $\mathcal{P}, \mathcal{Q}$ of 2D points, find $(x_1, y_1) \in \mathcal{P}$ and $(x_2, y_2) \in \mathcal{Q}$ such that $x_1x_2,\ y_1y_2 \geq \frac{1}{z}$

Weighted Consensus

Meet-in-the-middle:

1. For $i = 0, \ldots, n-1$ in order, compute $\mathcal{M}_{\sqrt{z}}(T[0,i])$
2. For $i = n, \ldots, 0$ in order, compute $\mathcal{M}_{\sqrt{z}}(T[i,n-1])$
3. // $\mathcal{O}(\sqrt{z}\log^2 z)$ time since $n = \mathcal{O}(\log z)$
4. For every $i = 0, \ldots, n-1$:
   1. try to join every $U \in \mathcal{M}_{\sqrt{z}}(T[0,i-1])$ and $V \in \mathcal{M}_{\sqrt{z}}(T[i+1,n-1])$ with a letter $c \in \Sigma$ at position i:
      - $\mathcal{P}(Uc, P[0,i]), \mathcal{P}(Uc, T[0,i]) \geq \frac{1}{z}$
      - $\mathcal{P}(V, P[i,n-1]) \geq \frac{1}{z}$
   2. Auxiliary problem: Given two sets $\mathcal{P}, \mathcal{Q}$ of 2D points, find $(x_1, y_1) \in \mathcal{P}$ and $(x_2, y_2) \in \mathcal{Q}$ such that $x_1 x_2,\ y_1 y_2 \geq \frac{1}{z}$
   3. Line sweep in $\mathcal{O}(|\mathcal{P}| + |\mathcal{Q}|)$ time.

Weighted Consensus

Meet-in-the-middle:

1. For $i = 0, \ldots, n-1$ in order, compute $\mathcal{M}_{\sqrt{z}}(T[0, i])$
2. For $i = n, \ldots, 0$ in order, compute $\mathcal{M}_{\sqrt{z}}(T[i, n-1])$
3. // $\mathcal{O}(\sqrt{z} \log^2 z)$ time since $n = \mathcal{O}(\log z)$
4. For every $i = 0, \ldots, n-1$:
    1. try to join every $U \in \mathcal{M}_{\sqrt{z}}(T[0, i-1])$ and $V \in \mathcal{M}_{\sqrt{z}}(T[i+1, n-1])$ with a letter $c \in \Sigma$ at position i:
        - $\mathcal{P}(Uc, P[0, i]), \mathcal{P}(Uc, T[0, i]) \geq \frac{1}{z}$
        - $\mathcal{P}(V, P[i, n-1]) \geq \frac{1}{z}$
    2. Auxiliary problem: Given two sets $\mathcal{P}, \mathcal{Q}$ of 2D points, find $(x_1, y_1) \in \mathcal{P}$ and $(x_2, y_2) \in \mathcal{Q}$ such that $x_1 x_2, y_1 y_2 \geq \frac{1}{z}$
    3. Line sweep in $\mathcal{O}(|\mathcal{P}| + |\mathcal{Q}|)$ time.

In total: $\mathcal{O}(\sqrt{z} \log^2 z)$ time.

General WPM and WCP

$\mathcal{O}(nz^2 \log z)$ time	[Barton-Liu-Pissis'15]
$\mathcal{O}(n\sqrt{z\lambda}(\log\log z + \log\lambda))$ time	[Kociumaka-Pissis-R'16]
$\mathcal{O}(n\sqrt{z}\log^2 z)$ time for $\sigma = \mathcal{O}(1)$	this presentation

[1] $\mathcal{O}^*$ and $\tilde{\mathcal{O}}$ suppress polynomial and polylog factors with respect to the instance size, resp.

General WPM and WCP

$\mathcal{O}(nz^2 \log z)$ time	[Barton-Liu-Pissis'15]
$\mathcal{O}(n\sqrt{z\lambda}(\log\log z + \log\lambda))$ time	[Kociumaka-Pissis-R'16]
$\mathcal{O}(n\sqrt{z}\log^2 z)$ time for $\sigma = \mathcal{O}(1)$	this presentation

Lower bounds[1]; see [Kociumaka-Pissis-R'16]:

no $\mathcal{O}^*(z^{\varepsilon})$ time for every $\varepsilon > 0$	unless the Exponential Time Hypothesis fails
no $\mathcal{O}^*(z^{0.5-\varepsilon})$ time for some $\varepsilon > 0$	unless a better algorithm for Subset Sum
no $\tilde{\mathcal{O}}(z^{0.5}\lambda^{0.5-\varepsilon})$ time for some $\varepsilon > 0$ and $n = \mathcal{O}(1)$	unless 3-Sum conjecture fails

[1] $\mathcal{O}^*$ and $\tilde{\mathcal{O}}$ suppress polynomial and polylog factors with respect to the instance size, resp.

Subset Sum and 3-Sum

Problem definitions:

- Subset Sum
  Input: a set A of n integers
  Output: a subset $B \subseteq A$ summing up to a given integer q, if any
- 3-Sum
  Input: three sets A, B, C of λ integers each
  Output: are there elements $a \in A$, $b \in B$, $c \in C$ such that $a + b + c = 0$?

Subset Sum and 3-Sum

Problem definitions:

- Subset Sum
  Input: a set A of n integers
  Output: a subset $B \subseteq A$ summing up to a given integer q, if any
- 3-Sum
  Input: three sets A, B, C of λ integers each
  Output: are there elements $a \in A$, $b \in B$, $c \in C$ such that $a + b + c = 0$?

Both problems are special cases of Multichoice Knapsack (MK).
We show a bidirectional reduction from MK to WCP.

Subset Sum and 3-Sum

Problem definitions:

- Subset Sum
  Input: a set A of n integers
  Output: a subset $B \subseteq A$ summing up to a given integer q, if any
- 3-Sum
  Input: three sets A, B, C of λ integers each
  Output: are there elements $a \in A$, $b \in B$, $c \in C$ such that $a + b + c = 0$?

Both problems are special cases of Multichoice Knapsack (MK).
We show a bidirectional reduction from MK to WCP.

Conditional hardness:

- No $\mathcal{O}(2^{0.5n-\varepsilon})$-time solution for Subset Sum is known ($\varepsilon > 0$)
- No $\mathcal{O}(\lambda^{2-\varepsilon})$-time solution for 3-Sum is known for $\varepsilon > 0$ (3-Sum conjecture)

Efficient Average-Case Algorithms for WPM

problem	preprocessing	avg search time	
WPST	$\mathcal{O}(m\sigma)$	$o(n)$ for small enough z/m	[Barton-Liu-Pissis'18]
SPWT	$\mathcal{O}(m)$	$\mathcal{O}(\frac{nz \log m}{m})$	[Barton-Liu-Pissis'16]
WPWT	$\mathcal{O}(mz)$	$\mathcal{O}(\frac{nz \log m}{m})$	[Barton-Liu-Pissis'16]

Efficient Average-Case Algorithms for WPM

problem	preprocessing	avg search time	
WPST	$\mathcal{O}(m\sigma)$	$o(n)$ for small enough z/m	[Barton-Liu-Pissis'18]
SPWT	$\mathcal{O}(m)$	$\mathcal{O}(\frac{nz \log m}{m})$	[Barton-Liu-Pissis'16]
WPWT	$\mathcal{O}(mz)$	$\mathcal{O}(\frac{nz \log m}{m})$	[Barton-Liu-Pissis'16]

Implementations provided

Plan of Presentation

1. Weighted Strings
2. Weighted Pattern Matching and Profile Matching
3. General Weighted Pattern Matching
4. Weighted Indexing
5. On-line and Streaming Weighted Pattern Matching
6. Weighted LCS and SCS

Weighted Indexing

Input

- T – weighted string text of length n
- $\frac{1}{z}$ – threshold probability
- $\sigma = \mathcal{O}(1)$ – alphabet size (this presentation)

Query

- Input: P – string pattern of length m
- Output: $\mathrm{Occ}_z(P, T)$ – the set of occurrences of P in T

Weighted Indexing

Input

- T – weighted string text of length n
- $\frac{1}{z}$ – threshold probability
- $\sigma = \mathcal{O}(1)$ – alphabet size (this presentation)

Query

- Input: P – string pattern of length m
- Output: $\mathrm{Occ}_z(P, T)$ – the set of occurrences of P in T

space	**construction**	
$\mathcal{O}(nf(z))$	$\mathcal{O}(nf(z))$	[IMPPTT'06]
$\mathcal{O}(nz^2 \log z)$	$\mathcal{O}(nz^2 \log z(\log\log z + \log\log n))$	[ACIKZ'06]
$\mathcal{O}(nz^2 \log z)$	$\mathcal{O}(nz^2 \log z)$	[Iliopoulos-Rahman'08], [Juan-Liu-Wang'09]
$\mathcal{O}(nz)$	$\mathcal{O}(nz)$	[BKLPR'16]

Query time: $\mathcal{O}(m + |\mathrm{Occ}_z(P, T)|)$

Weighted Indexing

Input

- T – weighted string text of length n
- $\frac{1}{z}$ – threshold probability
- $\sigma = \mathcal{O}(1)$ – alphabet size (this presentation)

Query

- Input: P – string pattern of length m
- Output: $\mathrm{Occ}_z(P, T)$ – the set of occurrences of P in T

space	**construction**	
$\mathcal{O}(nf(z))$	$\mathcal{O}(nf(z))$	[IMPPTT'06]
$\mathcal{O}(nz^2 \log z)$	$\mathcal{O}(nz^2 \log z(\log \log z + \log \log n))$	[ACIKZ'06]
$\mathcal{O}(nz^2 \log z)$	$\mathcal{O}(nz^2 \log z)$	[Iliopoulos-Rahman'08], [Juan-Liu-Wang'09]
$\mathcal{O}(nz)$	$\mathcal{O}(nz)$	[BKLPR'16]

Query time: $\mathcal{O}(m + |\mathrm{Occ}_z(P, T)|)$

Weighted Indexing

Definition

A property Π is a hereditary collection of integer intervals contained in $\{0, \ldots, n-1\}$.

It is represented as an array π such that the longest interval starting at position i in Π is $[i, \pi[i]]$.

Definition

A property Π is a hereditary collection of integer intervals contained in $\{0, \ldots, n-1\}$.

It is represented as an array π such that the longest interval starting at position i in Π is $[i, \pi[i]]$.

For strings P, S and property π, by $\mathrm{Occ}_\pi(P, S)$ we denote the set of occurrences i of P in S such that $i + |P| - 1 \le \pi[i]$.

Weighted Indexing

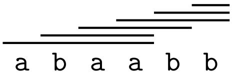
Definition

A property Π is a hereditary collection of integer intervals contained in $\{0, \dots, n-1\}$.

It is represented as an array π such that the longest interval starting at position i in Π is $[i, \pi[i]]$.

For strings P, S and property π, by $\mathrm{Occ}_\pi(P, S)$ we denote the set of occurrences i of P in S such that $i + |P| - 1 \le \pi[i]$.

i	0	1	2	3	4	5
$S[i]$	a	b	a	a	b	b
$\pi[i]$	3	3	4	5	5	5



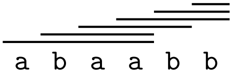
Definition

A property Π is a hereditary collection of integer intervals contained in $\{0, \ldots, n-1\}$.

It is represented as an array π such that the longest interval starting at position i in Π is $[i, \pi[i]]$.

For strings P, S and property π, by $\mathrm{Occ}_\pi(P, S)$ we denote the set of occurrences i of P in S such that $i + |P| - 1 \leq \pi[i]$.

i	0	1	2	3	4	5
$S[i]$	a	b	a	a	b	b
$\pi[i]$	3	3	4	5	5	5



Pattern $P = \mathtt{ab}$

Weighted Indexing

Definition

A property Π is a hereditary collection of integer intervals contained in $\{0, \ldots, n-1\}$.

It is represented as an array π such that the longest interval starting at position i in Π is $[i, \pi[i]]$.

For strings P, S and property π, by $\mathrm{Occ}_\pi(P, S)$ we denote the set of occurrences i of P in S such that $i + |P| - 1 \leq \pi[i]$.

i	0	1	2	3	4	5
$S[i]$	a	b	a	a	b	b
$\pi[i]$	3	3	4	5	5	5

a b a a b b

Pattern $P = \mathtt{ab}$

Weighted Indexing

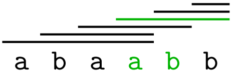
Definition

A property Π is a hereditary collection of integer intervals contained in $\{0, \ldots, n-1\}$.

It is represented as an array π such that the longest interval starting at position i in Π is $[i, \pi[i]]$.

For strings P, S and property π, by $\mathrm{Occ}_\pi(P, S)$ we denote the set of occurrences i of P in S such that $i + |P| - 1 \le \pi[i]$.

i	0	1	2	3	4	5
$S[i]$	a	b	a	a	b	b
$\pi[i]$	3	3	4	5	5	5



Pattern $P = \mathtt{ab}$

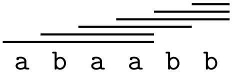
Definition

A property Π is a hereditary collection of integer intervals contained in $\{0, \ldots, n-1\}$.

It is represented as an array π such that the longest interval starting at position i in Π is $[i, \pi[i]]$.

For strings P, S and property π, by $\mathrm{Occ}_\pi(P, S)$ we denote the set of occurrences i of P in S such that $i + |P| - 1 \leq \pi[i]$.

i	0	1	2	3	4	5
$S[i]$	a	b	a	a	b	b
$\pi[i]$	3	3	4	5	5	5



Pattern $P = \mathtt{abaab}$

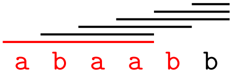
Definition

A property Π is a hereditary collection of integer intervals contained in $\{0, \ldots, n-1\}$.

It is represented as an array π such that the longest interval starting at position i in Π is $[i, \pi[i]]$.

For strings P, S and property π, by $\mathrm{Occ}_\pi(P, S)$ we denote the set of occurrences i of P in S such that $i + |P| - 1 \leq \pi[i]$.

i	0	1	2	3	4	5
$S[i]$	a	b	a	a	b	b
$\pi[i]$	3	3	4	5	5	5



Pattern $P = \mathtt{abaab}$

Weighted Indexing

Idea: construct a special family of $\lfloor z \rfloor$ strings with properties that are "equivalent" to the weighted sequence

Weighted Indexing

Idea: construct a special family of $\lfloor z \rfloor$ strings with properties that are "equivalent" to the weighted sequence

Fact 5

Input: a weighted sequence X of length n and a threshold z

Output: one can construct a family $\mathcal{S} = (S_j, \pi_j)$ of $\lfloor z \rfloor$ strings of length n with properties such that, for a string P and position i,

$$i \in \mathrm{Occ}_z(P, X) \quad \Leftrightarrow \quad i \in \mathrm{Occ}_\pi(P, S_j) \text{ for some } j.$$

Fact 5

Input: a weighted sequence X of length n and a threshold z

Output: one can construct a family $\mathcal{S} = (S_j, \pi_j)$ of $\lfloor z \rfloor$ strings of length n with properties such that, for a string P and position i,

$$i \in \mathrm{Occ}_z(P, X) \quad \Leftrightarrow \quad i \in \mathrm{Occ}_\pi(P, S_j) \text{ for some } j.$$

a 1	a $\frac{1}{2}$	a $\frac{3}{4}$	a $\frac{4}{5}$	a $\frac{1}{2}$	a $\frac{1}{4}$
b 0	b $\frac{1}{2}$	b $\frac{1}{4}$	b $\frac{1}{5}$	b $\frac{1}{2}$	b $\frac{3}{4}$

a b b b b b

a b a a b b

a a a a a a

a a a a a b

$z = 4$

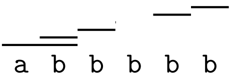
Fact 5

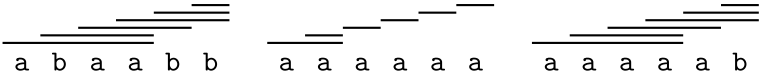
Input: a weighted sequence X of length n and a threshold z

Output: one can construct a family $\mathcal{S} = (S_j, \pi_j)$ of $\lfloor z \rfloor$ strings of length n with properties such that, for a string P and position i,

$$i \in \mathrm{Occ}_z(P, X) \quad \Leftrightarrow \quad i \in \mathrm{Occ}_\pi(P, S_j) \text{ for some } j.$$

a 1	a $\frac{1}{2}$	a $\frac{3}{4}$	a $\frac{4}{5}$	a $\frac{1}{2}$	a $\frac{1}{4}$
b 0	b $\frac{1}{2}$	b $\frac{1}{4}$	b $\frac{1}{5}$	b $\frac{1}{2}$	b $\frac{3}{4}$





$z = 4,\ P = \mathtt{aab}$

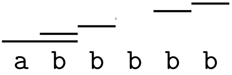
Fact 5

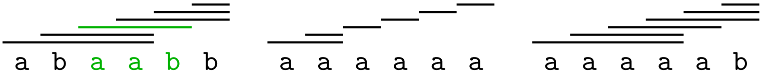
Input: a weighted sequence X of length n and a threshold z

Output: one can construct a family $\mathcal{S} = (S_j, \pi_j)$ of $\lfloor z \rfloor$ strings of length n with properties such that, for a string P and position i,

$$i \in \mathrm{Occ}_z(P, X) \quad \Leftrightarrow \quad i \in \mathrm{Occ}_\pi(P, S_j) \text{ for some } j.$$

a 1	a $\frac{1}{2}$	a $\frac{3}{4}$	a $\frac{4}{5}$	a $\frac{1}{2}$	a $\frac{1}{4}$
b 0	b $\frac{1}{2}$	b $\frac{1}{4}$	b $\frac{1}{5}$	b $\frac{1}{2}$	b $\frac{3}{4}$





$z = 4$, $P = \mathtt{aab}$, $i = \mathbf{2}$, $prob = 0.3$

Weighted Indexing

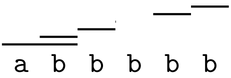
Fact 5

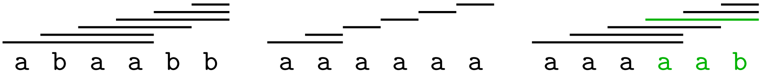
Input: a weighted sequence X of length n and a threshold z

Output: one can construct a family $\mathcal{S} = (S_j, \pi_j)$ of $\lfloor z \rfloor$ strings of length n with properties such that, for a string P and position i,

$$i \in \mathrm{Occ}_z(P, X) \quad \Leftrightarrow \quad i \in \mathrm{Occ}_\pi(P, S_j) \text{ for some } j.$$

a 1	a $\frac{1}{2}$	a $\frac{3}{4}$	a $\frac{4}{5}$	a $\frac{1}{2}$	a $\frac{1}{4}$
b 0	b $\frac{1}{2}$	b $\frac{1}{4}$	b $\frac{1}{5}$	b $\frac{1}{2}$	b $\frac{3}{4}$





$z = 4$, $P = \mathrm{aab}$, $i = \mathbf{3}$, $prob = 0.3$

Weighted Indexing

Fact 5

Input: a weighted sequence X of length n and a threshold z

Output: one can construct a family $\mathcal{S} = (S_j, \pi_j)$ of $\lfloor z \rfloor$ strings of length n with properties such that, for a string P and position i,

$$i \in \mathrm{Occ}_z(P, X) \quad \Leftrightarrow \quad i \in \mathrm{Occ}_\pi(P, S_j) \text{ for some } j.$$

a 1	a $\frac{1}{2}$	a $\frac{3}{4}$	a $\frac{4}{5}$	a $\frac{1}{2}$	a $\frac{1}{4}$
b 0	b $\frac{1}{2}$	b $\frac{1}{4}$	b $\frac{1}{5}$	b $\frac{1}{2}$	b $\frac{3}{4}$

a b b b b b

a b a a b b

a a a a a a

a a a a a b

$z = 4,\ P = \mathrm{ab},\ i = \mathbf{0},\ prob = 0.5$

Weighted Indexing

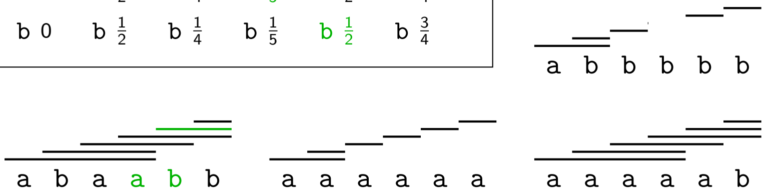
Fact 5

Input: a weighted sequence X of length n and a threshold z

Output: one can construct a family $\mathcal{S} = (S_j, \pi_j)$ of $\lfloor z \rfloor$ strings of length n with properties such that, for a string P and position i,

$$i \in \mathrm{Occ}_z(P, X) \quad \Leftrightarrow \quad i \in \mathrm{Occ}_\pi(P, S_j) \text{ for some } j.$$

a 1	a $\frac{1}{2}$	a $\frac{3}{4}$	a $\frac{4}{5}$	a $\frac{1}{2}$	a $\frac{1}{4}$
b 0	b $\frac{1}{2}$	b $\frac{1}{4}$	b $\frac{1}{5}$	b $\frac{1}{2}$	b $\frac{3}{4}$



$z = 4$, $P = \mathrm{ab}$, $i = \mathbf{3}$, $prob = 0.4$

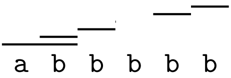
Fact 5

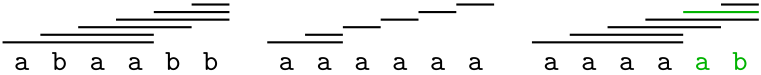
Input: a weighted sequence X of length n and a threshold z

Output: one can construct a family $\mathcal{S} = (S_j, \pi_j)$ of $\lfloor z \rfloor$ strings of length n with properties such that, for a string P and position i,

$$i \in \mathrm{Occ}_z(P, X) \quad \Leftrightarrow \quad i \in \mathrm{Occ}_\pi(P, S_j) \text{ for some } j.$$

a 1	a $\frac{1}{2}$	a $\frac{3}{4}$	a $\frac{4}{5}$	a $\frac{1}{2}$	a $\frac{1}{4}$
b 0	b $\frac{1}{2}$	b $\frac{1}{4}$	b $\frac{1}{5}$	b $\frac{1}{2}$	b $\frac{3}{4}$





$z = 4,\ P = \mathtt{ab},\ i = \mathbf{4},\ prob = 0.375$

Property Indexing

Input:

- S – string of length n
- π – property
- $\sigma = \mathcal{O}(1)$ – alphabet size (this presentation)

Query:

- Input: P – string pattern of length m
- Output: $\mathrm{Occ}_{\pi}(P, S)$

Input:

- S – string of length n
- π – property
- $\sigma = \mathcal{O}(1)$ – alphabet size (this presentation)

Query:

- Input: P – string pattern of length m
- Output: $\mathrm{Occ}_\pi(P, S)$

space	**construction**	
$\mathcal{O}(n)$	$\mathcal{O}(n \log\log n)$	[ACIKZ'06]
$\mathcal{O}(n)$	$\mathcal{O}(n)$	[Iliopoulos-Rahman'08], [Juan-Liu-Wang'09]

Query time: $\mathcal{O}(m + |\mathrm{Occ}_\pi(P, S)|)$

Property Indexing

Input:

- S – string of length n
- π – property
- $\sigma = \mathcal{O}(1)$ – alphabet size (this presentation)

Query:

- Input: P – string pattern of length m
- Output: $\mathrm{Occ}_\pi(P, S)$

space	construction	
$\mathcal{O}(n)$	$\mathcal{O}(n \log \log n)$	[ACIKZ'06]
$\mathcal{O}(n)$	$\mathcal{O}(n)$	[Iliopoulos-Rahman'08], [Juan-Liu-Wang'09]

Query time: $\mathcal{O}(m + |\mathrm{Occ}_\pi(P, S)|)$

- In [BKLPR'16]: a simpler construction based on Ukkonen's algorithm

Weighted Indexing

1. Construct a special family $\mathcal{S}$ of strings with properties ($\mathcal{O}(nz)$ time)
2. Concatenate the strings from $\mathcal{S}$ into a string S with property π
3. Construct a data structure for Property Indexing ($\mathcal{O}(nz)$ time and space, $\mathcal{O}(m + |\mathrm{Occ}_\pi(P, S)|)$ queries)

Weighted Indexing

1. Construct a special family $\mathcal{S}$ of strings with properties ($\mathcal{O}(nz)$ time)
2. Concatenate the strings from $\mathcal{S}$ into a string S with property π
3. Construct a data structure for Property Indexing ($\mathcal{O}(nz)$ time and space, $\mathcal{O}(m + |\mathrm{Occ}_\pi(P, S)|)$ queries)

Problem: If a pattern P occurs at position i in several strings from $\mathcal{S}$, then position i will be reported multiple times (i.e., $|\mathrm{Occ}_\pi(P, S)| \geq |\mathrm{Occ}_z(P, T)|$)

Weighted Indexing

1. Construct a special family $\mathcal{S}$ of strings with properties ($\mathcal{O}(nz)$ time)
2. Concatenate the strings from $\mathcal{S}$ into a string S with property π
3. Construct a data structure for Property Indexing ($\mathcal{O}(nz)$ time and space, $\mathcal{O}(m + |\mathrm{Occ}_\pi(P, S)|)$ queries)

Problem: If a pattern P occurs at position i in several strings from $\mathcal{S}$, then position i will be reported multiple times (i.e., $|\mathrm{Occ}_\pi(P, S)| \geq |\mathrm{Occ}_z(P, T)|$)

Solution: Use colored range listing of [Muthukrishnan'02].

Weighted Indexing

1. Construct a special family $\mathcal{S}$ of strings with properties ($\mathcal{O}(nz)$ time)
2. Concatenate the strings from $\mathcal{S}$ into a string S with property π
3. Construct a data structure for Property Indexing ($\mathcal{O}(nz)$ time and space, $\mathcal{O}(m + |\mathrm{Occ}_\pi(P, S)|)$ queries)

Problem: If a pattern P occurs at position i in several strings from $\mathcal{S}$, then position i will be reported multiple times (i.e., $|\mathrm{Occ}_\pi(P, S)| \geq |\mathrm{Occ}_z(P, T)|$)

Solution: Use colored range listing of [Muthukrishnan'02].

- The index answers decision and counting queries in $\mathcal{O}(1)$ time (for counting, we use the Color Set Size problem of [Hui'92])

Weighted Indexing

1. Construct a special family $\mathcal{S}$ of strings with properties ($\mathcal{O}(nz)$ time)
2. Concatenate the strings from $\mathcal{S}$ into a string S with property π
3. Construct a data structure for Property Indexing ($\mathcal{O}(nz)$ time and space, $\mathcal{O}(m + |\mathrm{Occ}_{\pi}(P, S)|)$ queries)

Problem: If a pattern P occurs at position i in several strings from $\mathcal{S}$, then position i will be reported multiple times (i.e., $|\mathrm{Occ}_{\pi}(P, S)| \geq |\mathrm{Occ}_{z}(P, T)|$)

Solution: Use colored range listing of [Muthukrishnan'02].

- The index answers decision and counting queries in $\mathcal{O}(1)$ time (for counting, we use the Color Set Size problem of [Hui'92])
- And we provide its implementation

Approximate Weighted Indexing

Input

- T – weighted string text of length n
- $\varepsilon > 0$ – allowed error
- z – maximum threshold probability (optional)

Query

- Input: P – string pattern of length m, $z' \le z$ – threshold probability
- Output: Occ – the set of occurrences of P in T with probab. $\ge \frac{1}{z'}$, allowing occurrences with probab. $\ge \frac{1}{z'} - \varepsilon$

Approximate and Generalized Weighted Indexing

Approximate Weighted Indexing

Input

- T – weighted string text of length n
- $\varepsilon > 0$ – allowed error
- z – maximum threshold probability (optional)

Query

- Input: P – string pattern of length m, $z' \leq z$ – threshold probability
- Output: Occ – the set of occurrences of P in T with probab. $\geq \frac{1}{z'}$, allowing occurrences with probab. $\geq \frac{1}{z'} - \varepsilon$

space	construction	query	
$\mathcal{O}(\frac{n}{\varepsilon}z^2)$	$\Omega(\frac{1}{\varepsilon}n^2z^2)$	$\mathcal{O}(m + \lvert\mathrm{Occ}\rvert)$	[BPTS'16]
$\mathcal{O}(\frac{n}{\varepsilon})$	$\mathcal{O}(\frac{n}{\varepsilon}\log\frac{n}{\varepsilon})$	$\mathcal{O}(m + \lvert\mathrm{Occ}\rvert)$	[BKLPR'16]

Approximate and Generalized Weighted Indexing

Generalized Weighted Indexing

Input

- T – weighted string text of length n
- z – maximum threshold probability

Query

- Input: P – string pattern of length m, $z' \leq z$ – threshold probability
- Output: Occ – the set $\mathrm{Occ}_{z'}(P, T)$

Approximate and Generalized Weighted Indexing

Generalized Weighted Indexing

Input

- T – weighted string text of length n
- z – maximum threshold probability

Query

- Input: P – string pattern of length m, $z' \leq z$ – threshold probability
- Output: Occ – the set $\mathrm{Occ}_{z'}(P, T)$

space	**query**	
$\mathcal{O}(\frac{n}{\varepsilon} z^2 \log z)$	$\mathcal{O}(m + m\,\lvert\mathrm{Occ}\rvert)$	[BPTS'16]
$\mathcal{O}(\frac{n}{\varepsilon})$	$\mathcal{O}(m + m\,\lvert\mathrm{Occ}\rvert)$	[BKLPR'16]

Approximate and Generalized Weighted Indexing

Fact 5′

Input: a weighted sequence X of length n and a threshold z

Output: one can construct a family $\mathcal{S} = (S_j, \pi_j)$ of $\lfloor z \rfloor$ strings of length n with properties such that, for a string P and position i,

$$\lfloor z\,\mathcal{P}(P, T[i, i+m-1]) \rfloor \;=\; |\{j \,:\, i \in \mathrm{Occ}_\pi(P, S_j)\}|.$$

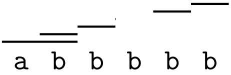
Fact 5′

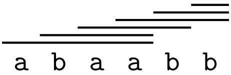
Input: a weighted sequence X of length n and a threshold z

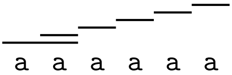
Output: one can construct a family $\mathcal{S} = (S_j, \pi_j)$ of $\lfloor z \rfloor$ strings of length n with properties such that, for a string P and position i,

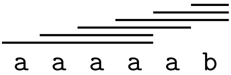
$$\lfloor z\mathcal{P}(P, T[i, i+m-1]) \rfloor \;=\; |\{j \;:\; i \in \mathrm{Occ}_\pi(P, S_j)\}|.$$

a 1	a $\frac{1}{2}$	a $\frac{3}{4}$	a $\frac{4}{5}$	a $\frac{1}{2}$	a $\frac{1}{4}$
b 0	b $\frac{1}{2}$	b $\frac{1}{4}$	b $\frac{1}{5}$	b $\frac{1}{2}$	b $\frac{3}{4}$









$z = 4$

Approximate and Generalized Weighted Indexing

Fact 5′

Input: a weighted sequence X of length n and a threshold z

Output: one can construct a family $\mathcal{S} = (S_j, \pi_j)$ of $\lfloor z \rfloor$ strings of length n with properties such that, for a string P and position i,

$$\lfloor z\,\mathcal{P}(P, T[i, i+m-1]) \rfloor \;=\; |\{j \,:\, i \in \mathrm{Occ}_{\pi}(P, S_j)\}|.$$

a 1	a $\frac{1}{2}$	a $\frac{3}{4}$	a $\frac{4}{5}$	a $\frac{1}{2}$	a $\frac{1}{4}$
b 0	b $\frac{1}{2}$	b $\frac{1}{4}$	b $\frac{1}{5}$	b $\frac{1}{2}$	b $\frac{3}{4}$

a b b b b b

a b a a b b

a a a a a a

a a a a a b

$z = 4,\ P = \mathrm{a},\ i = 0,\ prob = 1$

Fact 5′

Input: a weighted sequence X of length n and a threshold z

Output: one can construct a family $\mathcal{S} = (S_j, \pi_j)$ of $\lfloor z \rfloor$ strings of length n with properties such that, for a string P and position i,

$$\lfloor z\,\mathcal{P}(P, T[i, i+m-1])\,\rfloor \;=\; |\,\{j \,:\, i \in \mathrm{Occ}_{\pi}(P, S_j)\}\,|.$$

a 1	a $\frac{1}{2}$	a $\frac{3}{4}$	a $\frac{4}{5}$	a $\frac{1}{2}$	a $\frac{1}{4}$
b 0	b $\frac{1}{2}$	b $\frac{1}{4}$	b $\frac{1}{5}$	b $\frac{1}{2}$	b $\frac{3}{4}$

a b b b b b

a b a a b b   a a a a a a   a a a a a b

$z = 4,\ P = \mathrm{ab},\ i = 0,\ prob = 0.5$

Approximate and Generalized Weighted Indexing

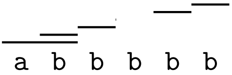
Fact 5′

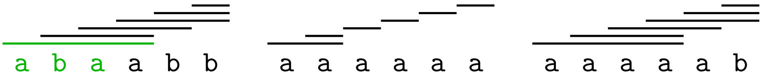
Input: a weighted sequence X of length n and a threshold z

Output: one can construct a family $\mathcal{S} = (S_j, \pi_j)$ of $\lfloor z \rfloor$ strings of length n with properties such that, for a string P and position i,

$$\lfloor z\,\mathcal{P}(P, T[i, i+m-1]) \rfloor \;=\; |\{j \,:\, i \in \mathrm{Occ}_\pi(P, S_j)\}|.$$

a 1	a $\frac{1}{2}$	a $\frac{3}{4}$	a $\frac{4}{5}$	a $\frac{1}{2}$	a $\frac{1}{4}$
b 0	b $\frac{1}{2}$	b $\frac{1}{4}$	b $\frac{1}{5}$	b $\frac{1}{2}$	b $\frac{3}{4}$





$z = 4,\ P = \text{aba},\ i = 0,\ prob = 0.375$

Plan of Presentation

1. Weighted Strings
2. Weighted Pattern Matching and Profile Matching
3. General Weighted Pattern Matching
4. Weighted Indexing
5. On-line and Streaming Weighted Pattern Matching
6. Weighted LCS and SCS

On-line WPM

Input:

- a pattern P of length m

Output:

- after $T[i]$ has been read, say if P matches $T[i-m+1,i]$

On-line WPM

Input:

- a pattern P of length m

Output:

- after $T[i]$ has been read, say if P matches $T[i-m+1, i]$

Objective:

- minimize time of processing a text position
- minimize working space

On-line WPM

Input:

- a pattern P of length m

Output:

- after $T[i]$ has been read, say if P matches $T[i-m+1, i]$

Objective:

- minimize time of processing a text position
- minimize working space

- We consider WPST, SPWT, WPWT

On-line WPM

Input:

- a pattern P of length m

Output:

- after $T[i]$ has been read, say if P matches $T[i-m+1,i]$

Objective:

- minimize time of processing a text position
- minimize working space

- We consider WPST, SPWT, WPWT
- All the previous algorithms were *not* on-line (FFT; suffix array of $T\$P$; meet-in-the-middle)

On-line WPM

Black-box scheme [Clifford-Efremenko-Porat-Porat'08]

Pattern matching off-line $\rightarrow$ on-line

On-line WPM

Black-box scheme [Clifford-Efremenko-Porat-Porat'08]

Pattern matching off-line $\rightarrow$ on-line

Off-line: $T(n, m)$ total time and $S(n, m)$ space
On-line: $\frac{1}{n}T(n, m)\log m$ time per position and $S(m, m)$ space

On-line WPM

Black-box scheme [Clifford-Efremenko-Porat-Porat'08]

Pattern matching off-line $\rightarrow$ on-line

Off-line: $T(n, m)$ total time and $S(n, m)$ space
On-line: $\frac{1}{n}T(n, m) \log m$ time per position and $S(m, m)$ space

Space to store a weighted string of length m: $\mathcal{O}(m \min(\sigma, z))$

problem	time/position	space	
WPST	$\mathcal{O}(\sigma \log^2 m)$	pattern	FFT+Scheme
WPST	$\mathcal{O}(\log z \log m)$	pattern	Lookahead+Scheme

On-line WPM

Black-box scheme [Clifford-Efremenko-Porat-Porat'08]

Pattern matching off-line → on-line

Off-line: $T(n,m)$ total time and $S(n,m)$ space
On-line: $\frac{1}{n}T(n,m)\log m$ time per position and $S(m,m)$ space

Space to store a weighted string of length m: $\mathcal{O}(m\min(\sigma,z))$

problem	**time/position**	**space**	
WPST	$\mathcal{O}(\sigma\log^2 m)$	pattern	FFT+Scheme
WPST	$\mathcal{O}(\log z\log m)$	pattern	Lookahead+Scheme
WPST	$\mathcal{O}(\sigma\log^2 m)$	$\mathcal{O}(m+z)$	[CIPR'19]
WPST	$\mathcal{O}(\log z\log m)$	$\mathcal{O}(m+z)$	[CIPR'19]

On-line WPM

Black-box scheme [Clifford-Efremenko-Porat-Porat'08]

Pattern matching off-line → on-line

Off-line: $T(n,m)$ total time and $S(n,m)$ space
On-line: $\frac{1}{n}T(n,m)\log m$ time per position and $S(m,m)$ space

Space to store a weighted string of length m: $\mathcal{O}(m \min(\sigma, z))$

problem	time/position	space	
WPST	$\mathcal{O}(\sigma \log^2 m)$	pattern	FFT+Scheme
WPST	$\mathcal{O}(\log z \log m)$	pattern	Lookahead+Scheme
WPST	$\mathcal{O}(\sigma \log^2 m)$	$\mathcal{O}(m+z)$	[CIPR'19]
WPST	$\mathcal{O}(\log z \log m)$	$\mathcal{O}(m+z)$	[CIPR'19]
SPWT	$\mathcal{O}(\sigma \log^2 m)$	text frag.	FFT+Scheme
SPWT	$\mathcal{O}((\sigma + \log z) \log m)$	text frag.	Lookahead+Scheme

On-line WPM

Black-box scheme [Clifford-Efremenko-Porat-Porat'08]

Pattern matching off-line $\rightarrow$ on-line

Off-line: $T(n, m)$ total time and $S(n, m)$ space
On-line: $\frac{1}{n}T(n, m)\log m$ time per position and $S(m, m)$ space

Space to store a weighted string of length m: $\mathcal{O}(m \min(\sigma, z))$

problem	time/position	space	
WPST	$\mathcal{O}(\sigma \log^2 m)$	pattern	FFT+Scheme
WPST	$\mathcal{O}(\log z \log m)$	pattern	Lookahead+Scheme
WPST	$\mathcal{O}(\sigma \log^2 m)$	$\mathcal{O}(m + z)$	[CIPR'19]
WPST	$\mathcal{O}(\log z \log m)$	$\mathcal{O}(m + z)$	[CIPR'19]
SPWT	$\mathcal{O}(\sigma \log^2 m)$	text frag.	FFT+Scheme
SPWT	$\mathcal{O}((\sigma + \log z) \log m)$	text frag.	Lookahead+Scheme
SPWT	$\mathcal{O}(z + \sigma)$	$\mathcal{O}(m + z)$	[CIPR'19]

On-line WPM

Black-box scheme [Clifford-Efremenko-Porat-Porat'08]

Pattern matching off-line $\rightarrow$ on-line

Off-line: $T(n,m)$ total time and $S(n,m)$ space
On-line: $\frac{1}{n}T(n,m)\log m$ time per position and $S(m,m)$ space

Space to store a weighted string of length m: $\mathcal{O}(m\min(\sigma,z))$

problem	time/position	space	
WPST	$\mathcal{O}(\sigma\log^2 m)$	pattern	FFT+Scheme
WPST	$\mathcal{O}(\log z\log m)$	pattern	Lookahead+Scheme
WPST	$\mathcal{O}(\sigma\log^2 m)$	$\mathcal{O}(m+z)$	[CIPR'19]
WPST	$\mathcal{O}(\log z\log m)$	$\mathcal{O}(m+z)$	[CIPR'19]
SPWT	$\mathcal{O}(\sigma\log^2 m)$	text frag.	FFT+Scheme
SPWT	$\mathcal{O}((\sigma+\log z)\log m)$	text frag.	Lookahead+Scheme
SPWT	$\mathcal{O}(z+\sigma)$	$\mathcal{O}(m+z)$	[CIPR'19]
WPWT	$\mathcal{O}(z+\sigma)$	$\mathcal{O}(mz^2)$	[CIPR'19]

Streaming WPM

Input:

- a pattern P of length m, read position by position

Output:

- after $T[i]$ has been read, say if P matches $T[i-m+1,i]$

Streaming WPM

Input:

- a pattern P of length m, read position by position

Output:

- after $T[i]$ has been read, say if P matches $T[i - m + 1, i]$

Objective:

- sublinear working space (store neither the pattern nor the text)
- often at the cost of randomization
- minimize time of processing a text position

Streaming WPM

Input:

- a pattern P of length m, read position by position

Output:

- after $T[i]$ has been read, say if P matches $T[i-m+1, i]$

Objective:

- sublinear working space (store neither the pattern nor the text)
- often at the cost of randomization
- minimize time of processing a text position

Streaming pattern matching on strings

First read the pattern, then read the text reporting the occurrences

Streaming WPM

Input:

- a pattern P of length m, read position by position

Output:

- after $T[i]$ has been read, say if P matches $T[i-m+1, i]$

Objective:

- sublinear working space (store neither the pattern nor the text)
- often at the cost of randomization
- minimize time of processing a text position

Streaming pattern matching on strings

First read the pattern, then read the text reporting the occurrences

space	time/position	
$\mathcal{O}(\log m)$	$\mathcal{O}(\log m)$ whp.	[Porat-Porat'09]
$\mathcal{O}(\log m)$	$\mathcal{O}(1)$ whp.	[Breslauer-Galil'11]

Streaming WPM

- In the Lookahead scoring for SPWT, we considered all occurrences of the pattern P in the heavy string **T** with at most $\log_2 z$ mismatches

Streaming WPM

- In the Lookahead scoring for SPWT, we considered all occurrences of the pattern P in the heavy string $\mathbf{T}$ with at most $\log_2 z$ mismatches

Streaming k-Mismatch on strings

"Error Correcting" (EC): Reports the positions of mismatches and differences of letters at these positions

Streaming WPM

- In the Lookahead scoring for SPWT, we considered all occurrences of the pattern P in the heavy string **T** with at most $\log_2 z$ mismatches

Streaming k-Mismatch on strings

"Error Correcting" (EC): Reports the positions of mismatches and differences of letters at these positions

space	time/position	EC	
$\tilde{O}(k^3)$	$\tilde{O}(k^2)$	No	[Porat-Porat'09]
$\tilde{O}(k^2)$	$\tilde{O}(\sqrt{k})$	No	[CFPSS'16]
$\tilde{O}(k^2)$	$\tilde{O}(k)$	Yes	[R-Starikovskaya'17]

Streaming WPM

- In the Lookahead scoring for SPWT, we considered all occurrences of the pattern P in the heavy string **T** with at most $\log_2 z$ mismatches

Streaming k-Mismatch on strings

"Error Correcting" (EC): Reports the positions of mismatches and differences of letters at these positions

space	**time/position**	**EC**	
$\tilde{\mathcal{O}}(k^3)$	$\tilde{\mathcal{O}}(k^2)$	No	[Porat-Porat'09]
$\tilde{\mathcal{O}}(k^2)$	$\tilde{\mathcal{O}}(\sqrt{k})$	No	[CFPSS'16]
$\tilde{\mathcal{O}}(k^2)$	$\tilde{\mathcal{O}}(k)$	Yes	[R-Starikovskaya'17]
$\tilde{\mathcal{O}}(k)$	$\tilde{\mathcal{O}}(k)$	Yes	[Clifford-Kociumaka-Porat'19]

Streaming WPM

- In the Lookahead scoring for SPWT, we considered all occurrences of the pattern P in the heavy string $\mathbf{T}$ with at most $\log_2 z$ mismatches

Streaming k-Mismatch on strings

"Error Correcting" (EC): Reports the positions of mismatches and differences of letters at these positions

space	**time/position**	**EC**	
$\tilde{\mathcal{O}}(k^3)$	$\tilde{\mathcal{O}}(k^2)$	No	[Porat-Porat'09]
$\tilde{\mathcal{O}}(k^2)$	$\tilde{\mathcal{O}}(\sqrt{k})$	No	[CFPSS'16]
$\tilde{\mathcal{O}}(k^2)$	$\tilde{\mathcal{O}}(k)$	Yes	[R-Starikovskaya'17]
$\tilde{\mathcal{O}}(k)$	$\tilde{\mathcal{O}}(k)$	Yes	[Clifford-Kociumaka-Porat'19]

S_k and T_k – space and time/position for streaming k-Mismatch

Streaming WPM

In [R-Starikovskaya'17] using Streaming k-Mismatch:

problem	**space**	**time/position**	**approx**
WPST	$\mathcal{O}(z + S_{\log z})$	$\mathcal{O}(\log^2 z + T_{\log z})$	
SPWT	$\mathcal{O}(z \log_{\frac{1}{1-\varepsilon}} z + S_{\log z})$	$\mathcal{O}(z \log_{\frac{1}{1-\varepsilon}} z + T_{\log z})$	$1-\varepsilon$

Streaming WPM

In [R-Starikovskaya'17] using Streaming k-Mismatch:

problem	**space**	**time/position**	**approx**
WPST	$\mathcal{O}(z + S_{\log z})$	$\mathcal{O}(\log^2 z + T_{\log z})$	
SPWT	$\mathcal{O}(z \log_{\frac{1}{1-\varepsilon}} z + S_{\log z})$	$\mathcal{O}(z \log_{\frac{1}{1-\varepsilon}} z + T_{\log z})$	$1-\varepsilon$

In [R-Starikovskaya'17] using Streaming MultiPattern Matching:

problem	**space**	**time/position**	**appr.**
WPST	$\mathcal{O}(z \log m)$	$\mathcal{O}(1)$	
SPWT	$\mathcal{O}(z(\log_{\frac{1}{1-\varepsilon}} z + \log m))$	$\mathcal{O}(z \log_{\frac{1}{1-\varepsilon}} z)$	$1-\varepsilon$
WPWT	$\mathcal{O}(z(\log_{\frac{1}{1-\varepsilon}} z + \log z \log m))$	$\mathcal{O}(z(\log_{\frac{1}{1-\varepsilon}} z + \log z \log m))$	$1-\varepsilon$

Streaming WPM

In [R-Starikovskaya'17] using Streaming k-Mismatch:

problem	space	time/position	approx
WPST	$\mathcal{O}(z + S_{\log z})$	$\mathcal{O}(\log^2 z + T_{\log z})$	
SPWT	$\mathcal{O}(z \log_{\frac{1}{1-\varepsilon}} z + S_{\log z})$	$\mathcal{O}(z \log_{\frac{1}{1-\varepsilon}} z + T_{\log z})$	$1-\varepsilon$

In [R-Starikovskaya'17] using Streaming MultiPattern Matching:

problem	space	time/position	appr.
WPST	$\mathcal{O}(z \log m)$	$\mathcal{O}(1)$	
SPWT	$\mathcal{O}(z(\log_{\frac{1}{1-\varepsilon}} z + \log m))$	$\mathcal{O}(z \log_{\frac{1}{1-\varepsilon}} z)$	$1-\varepsilon$
WPWT	$\mathcal{O}(z(\log_{\frac{1}{1-\varepsilon}} z + \log z \log m))$	$\mathcal{O}(z(\log_{\frac{1}{1-\varepsilon}} z + \log z \log m))$	$1-\varepsilon$

Lower bound in [R-Starikovskaya'17]

Any streaming algorithm, exact or $(1-\varepsilon)$-approximate, solving WPST, SPWT or WPWT must use $\Omega(z)$ space.

Plan of Presentation

1. Weighted Strings
2. Weighted Pattern Matching and Profile Matching
3. General Weighted Pattern Matching
4. Weighted Indexing
5. On-line and Streaming Weighted Pattern Matching
6. Weighted LCS and SCS

Weighted LCS and SCS

Definition (z-subsequence)

For a string S, a weighted sequence W and a threshold $\frac{1}{z}$, we write $S \subseteq_z W$ if $\mathcal{P}(S, W') \geq \frac{1}{z}$ for some subsequence W' of W.

Weighted LCS and SCS

Definition (z-subsequence)

For a string S, a weighted sequence W and a threshold $\frac{1}{z}$, we write $S \subseteq_z W$ if $\mathcal{P}(S, W') \geq \frac{1}{z}$ for some subsequence W' of W.

Weighted Longest Common Subsequence, [Amir-Gotthilf-Shalom'09]

Input

- W_1, W_2 – weighted strings of length n
- $\frac{1}{z}$ – threshold probability
- $\sigma = \mathcal{O}(1)$ – alphabet size (this presentation)

Output

- A longest string S such that $S \subseteq_z W_1$ and $S \subseteq_z W_2$

Weighted LCS and SCS

Definition (z-subsequence)

For a string S, a weighted sequence W and a threshold $\frac{1}{z}$, we write $W \subseteq_z S$ if $\mathcal{P}(S', W) \geq \frac{1}{z}$ for some subsequence S' of S.

Weighted LCS and SCS

Definition (z-subsequence)

For a string S, a weighted sequence W and a threshold $\frac{1}{z}$, we write $W \subseteq_z S$ if $\mathcal{P}(S', W) \geq \frac{1}{z}$ for some subsequence S' of S.

Weighted Shortest Common Supersequence, [Amir-Gotthilf-Shalom'11]

Input

- W_1, W_2 – weighted strings of length n
- $\frac{1}{z}$ – threshold probability
- $\sigma = \mathcal{O}(1)$ – alphabet size (this presentation)

Output

- A shortest string S such that $W_1 \subseteq_z S$ and $W_2 \subseteq_z S$

Weighted LCS and SCS

- Both problems are NP-complete for $\sigma = 2$; see [CKRRW'11] and [CKPRRSWZ'19]

Weighted LCS and SCS

- Both problems are NP-complete for $\sigma = 2$; see [CKRRW'11] and [CKPRRSWZ'19]

Weighted SCS

Upper bound

- $\mathcal{O}(n^2\sqrt{z}\log z)$ (using Facts 1, 2, 4); see [CKPRRSWZ'19]

Lower bounds

- $\mathcal{O}(n^{2-\varepsilon})$ unless the Strong Exponential Time Hypothesis fails; see [Abboud-Backurs-Williams'15]
- $\mathcal{O}^*(z^{0.5-\varepsilon})$ unless a better algorithm for Subset Sum exists; see [Kociumaka-Pissis-R]

Weighted LCS and SCS

- Both problems are NP-complete for $\sigma = 2$; see [CKRRW'11] and [CKPRRSWZ'19]

Weighted SCS

Upper bound

- $\mathcal{O}(n^2\sqrt{z}\log z)$ (using Facts 1, 2, 4); see [CKPRRSWZ'19]

Lower bounds

- $\mathcal{O}(n^{2-\varepsilon})$ unless the Strong Exponential Time Hypothesis fails; see [Abboud-Backurs-Williams'15]
- $\mathcal{O}^*(z^{0.5-\varepsilon})$ unless a better algorithm for Subset Sum exists; see [Kociumaka-Pissis-R]

Weighted LCS

- Cannot be solved in $\mathcal{O}(n\,f(z))$ time or $\mathcal{O}(n^{f(z)})$ time unless $P = NP$; see [CKPRRSWZ'19]

Conclusion

1. Weighted Strings
2. Weighted Pattern Matching and Profile Matching
3. General Weighted Pattern Matching
4. Weighted Indexing
5. On-line and Streaming Weighted Pattern Matching
6. Weighted LCS and SCS

Open Problems and Further Work

- A Weighted Index with $\mathcal{O}(m + |\mathrm{Occ}_z(P, T)|)$-time queries and $o(nz)$ space?

Open Problems and Further Work

- A Weighted Index with $\mathcal{O}(m + |\mathrm{Occ}_z(P, T)|)$-time queries and $o(nz)$ space?
- An on-line algorithm for General WPM using less space?
- More efficient queries in the Generalized Weighed Index?

Open Problems and Further Work

- A Weighted Index with $\mathcal{O}(m + |\mathrm{Occ}_z(P, T)|)$-time queries and $o(nz)$ space?
- An on-line algorithm for General WPM using less space?
- More efficient queries in the Generalized Weighed Index?
- Automatic selection of parameter z?
- Non-independent probability distributions?

References Presented Today

[BKLPR'16]
C. Barton, T. Kociumaka, C. Liu, S.P. Pissis, R,
Efficient Index for Weighted Sequences, CPM 2016
Full version (with C. Liu) accepted to **Inf. Comput.**
`https://arxiv.org/abs/1704.07625`

[CIPR'19]
P. Charalampopoulos, C.S. Iliopoulos, S.P. Pissis, R
On-line weighted pattern matching, **Inf. Comput.** 266, 2019

[Kociumaka-Pissis-R'16]
Pattern Matching and Consensus Problems on Weighted Sequences and Profiles, ISAAC 2016
Full version in **Theory Comput. Syst.** 63(3), 2019

[R-Starikovskaya]
Streaming k-Mismatch with Error Correcting and Applications, DCC 2017
Full version: `https://arxiv.org/abs/1607.05626`

References Presented Today

[Barton-Liu-Pissis'16]
On-Line Pattern Matching on Uncertain Sequences and Applications,
COCOA 2016

[Barton-Liu-Pissis'18]
Fast Average-Case Pattern Matching on Weighted Sequences,
Int. J. Found. Comput. Sci. 29(8), 2018

[CKRRW'11]
M. Cygan, M. Kubica, R, W. Rytter, T. Waleń,
Polynomial-Time Approximation Algorithms for Weighted LCS Problem,
CPM 2011
Full version in **Discr. Appl. Math.** 204, 2016

[CKPRRSWZ'19]
P. Charalampopoulos, T. Kociumaka, S.P. Pissis, R, W. Rytter, J. Straszyński,
T. Waleń, W. Zuba,
Weighted Shortest Common Supersequence Problem Revisited, SPIRE 2019

Other References – WPM

[ACIKZ'06]
A. Amir, E. Chencinski, C.S. Iliopoulos, T. Kopelowitz, H. Zhang,
Property matching and weighted matching, CPM 2006
Full version in **Theor. Comput. Sci.** 395 (2-3), 2008

[Amir-Gotthilf-Shalom'09]
Weighted LCS, IWOCA 2009
Full version in **J. Discrete Algorithms** 8(3), 2010

[Amir-Gotthilf-Shalom'11]
Weighted Shortest Common Supersequence, SPIRE 2011

[Barton-Liu-Pissis'15]
Linear-Time Computation of Prefix Table for Weighted Strings, WORDS 2015
Full version (with C. Liu) in **Theor. Comput. Sci.** 656, 2016

[BPTS'16]
S. Biswas, M. Patil, S.V. Thankachan, R. Shah,
Probabilistic Threshold Indexing for Uncertain Strings, EDBT 2016

[CIMT'04]
M. Christodoulakis, C.S. Iliopoulos, L. Mouchard, K. Tsichlas,
Pattern matching on weighted sequences, CompBioNets 2004

Other References – WPM

[IMPPTT'06]
C.S. Iliopoulos, C. Makris, Y. Panagis, K. Perdikuri, E. Theodoridis, A.K. Tsakalidis,
The weighted suffix tree: An efficient data structure for handling molecular weighted sequences and its applications,
Fundam. Inform. 71 (2-3), 2006

[Iliopoulos-Rahman'08]
Faster index for property matching,
Inf. Process. Lett. 105 (6), 2008

[Juan-Liu-Wang'09]
Errata for "Faster index for property matching",
Inf. Process. Lett. 109 (18), 2009

[Pizzi-Ukkonen'08]
Fast profile matching algorithms – A survey,
Theor. Comput. Sci. 395(2-3), 2008

[Rajasekaran-Jin-Spouge'02]
The Efficient Computation of Position-Specific Match Scores with the Fast Fourier Transform, **J. Comp. Biol.** 9(1): 23-33 (2002)

Other References – General Tools

[Breslauer-Galil'11]
Real-Time Streaming String-Matching, CPM 2011
Full version in **ACM Trans. Algorithms** 10(4), 2014

[Clifford-Efremenko-Porat-Porat'08]
A Black Box for Online Approximate Pattern Matching, CPM 2008
Full version in **Inf. Comput.** 209(4), 2011

[CFPSS'16]
R. Clifford, A. Fontaine, E. Porat, B. Sach, T. Starikovskaya,
The k-mismatch problem revisited, SODA 2016

[Clifford-Kociumaka-Porat'19]
The streaming k-mismatch problem, SODA 2019

[Hui'92]
Color set size problem with application to string matching, CPM 1992

[Muthukrishnan'02]
Efficient algorithms for document retrieval problems, SODA 2002

[Porat-Porat'09]
Exact and approximate pattern matching in the streaming model, FOCS 2009

Thank You

Thank you for your attention!