Faster 2D Pattern Matching With k Mismatches

Jonas Ellert, Paweł Gawrychowski, Adam Górkiewicz, Tatiana Starikovskaya

Pattern matching with mismatches

1D pattern: 1D text:

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

Pattern matching with mismatches

1D pattern: 1D text:

`a b c a b`

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

Pattern matching with mismatches

1D pattern:

a b c a b

1D text:

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

– – ✓ – – ✓ – – – – – ✓ – – – – –

Pattern matching with mismatches

1D pattern:

a b c a b

1D text:

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

– – ✓ – – ✓ – – – – – ✓ – – – – –

2D pattern:

a b c
d e a
f h d

2D text:

z z a b z z z z z
z z z e a b c z z
z z a h d e a z z
z z z z f h d z z
a b c z z z a b c
d e a b c z d z a
f h d e a z f h d
z z f h d z z z z
z z z z z z z z z

Pattern matching with mismatches

1D pattern:

a b c a b

1D text:

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

– – ✓ – – ✓ – – – – – ✓ – – – – –

2D pattern:

a b c
d e a
f h d

2D text:

z z a b z z z z z
z z z e a b c z z
z z a h d e a z z
z z z z f h d z z
a b c z z z a b c
d e a b c z d z a
f h d e a z f h d
z z f h d z z z z
z z z z z z z z z

Pattern matching with mismatches

1D pattern:

```
a b c a b
```

1D text:

```
a b a b c a b c a b b a b c a b a
– – ✓ – – ✓ – – – – – ✓ – – – – –
```

2D pattern:

```
a b c
d e a
f h d
```

2D text:

```
z z a b z z z z z    – – – – – – – – –
z z z e a b c z z    – – – – ✓ – – – –
z z a h d e a z z    – – – – – – – – –
z z z z f h d z z    – – – – – – – – –
a b c z z z a b c    ✓ – – – – – – – –
d e a b c z d z a    – – ✓ – – – – – –
f h d e a z f h d    – – – – – – – – –
z z f h d z z z z    – – – – – – – – –
z z z z z z z z z    – – – – – – – – –
```

Pattern matching with mismatches

1D pattern:

a b c a b

1D text:

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

– – ✓ – – ✓ – – – – – ✓ – – – – –

What if we allow up to k mismatches?

for example, $k = 3$ mismatches

2D pattern:

a	b	c
d	e	a
f	h	d

2D text:

z	z	a	b	z	z	z	z	z
z	z	z	e	a	b	c	z	z
z	z	a	h	d	e	a	z	z
z	z	z	z	f	h	d	z	z
a	b	c	z	z	z	a	b	c
d	e	a	b	c	z	d	z	a
f	h	d	e	a	z	f	h	d
z	z	f	h	d	z	z	z	z
z	z	z	z	z	z	z	z	z

–	–	–	–	–	–	–	–	–
–	–	–	–	✓	–	–	–	–
–	–	–	–	–	–	–	–	–
–	–	–	–	–	–	–	–	–
✓	–	–	–	–	–	–	–	–
–	–	✓	–	–	–	–	–	–
–	–	–	–	–	–	–	–	–
–	–	–	–	–	–	–	–	–
–	–	–	–	–	–	–	–	–

Pattern matching with mismatches

1D pattern:

```
a b c a b
```

1D text:

```
a b a b c a b c a b b a b c a b a
– – ✓ – – ✓ – – – – – ✓ – – – – –
```

2D pattern:

```
a b c
d e a
f h d
```

2D text:

```
z z a b z z z z z    – – – – – – – – –
z z z e a b c z z    – – – – ✓ – – – –
z z a h d e a z z    – – – – – – – – –
z z z z f h d z z    – – – – – – – – –
a b c z z z a b c    ✓ – – – – – – – –
d e a b c z d z a    – – ✓ – – – – – –
f h d e a z f h d    – – – – – – – – –
z z f h d z z z z    – – – – – – – – –
z z z z z z z z z    – – – – – – – – –
```

What if we allow up to k mismatches?

for example, $k = 3$ mismatches

Pattern matching with mismatches

1D pattern:

a b c a b

1D text:

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

– – ✓ – – ✓ – – – – – ✓ – – – – –

What if we allow up to k mismatches?

for example, $k = 3$ mismatches

2D pattern:

a b c
d e a
f h d

2D text:

z z a b z z z z z
z z z e a b c z z
z z a h d e a z z
z z z z f h d z z
a b c z z z a b c
d e a b c z d z a
f h d e a z f h d
z z f h d z z z z
z z z z z z z z z

– – – – – – – – –
– – – – ✓ – – – –
– – – – – – – – –
– – – – – – – – –
✓ – – – – – – – –
– – ✓ – – – – – –
– – – – – – – – –
– – – – – – – – –
– – – – – – – – –

Pattern matching with mismatches

1D pattern:

```
a b c a b
```

1D text:

```
a b a b c a b c a b b a b c a b a
– – ✓ – – ✓ – – – – – ✓ – – – – –
```

2D pattern:

```
a b c
d e a
f h d
```

2D text:

```
z z a b z z z z z     – – – – – – – – –
z z z e a b c z z     – – – – ✓ – – – –
z z a h d e a z z     – – – – – – – – –
z z z z f h d z z     – – – – – – – – –
a b c z z z a b c     ✓ – – – – – – – –
d e a b c z d z a     – – ✓ – – – – – –
f h d e a z f h d     – – – – – – – – –
z z f h d z z z z     – – – – – – – – –
z z z z z z z z z     – – – – – – – – –
```

What if we allow up to k mismatches?

for example, $k = 3$ mismatches

Pattern matching with mismatches

1D pattern:

a b c a b

1D text:

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

3 – 0 – – 0 – – 1 – – 0 – – – – –

What if we allow up to k mismatches?

for example, $k = 3$ mismatches

2D pattern:

a b c

d e a

f h d

2D text:

z z a b z z z z z – – – – – – – – –

z z z e a b c z z – – – – ✓ – – – –

z z a h d e a z z – – – – – – – – –

z z z z f h d z z – – – – – – – – –

a b c z z z a b c ✓ – – – – – – – –

d e a b c z d z a – – ✓ – – – – – –

f h d e a z f h d – – – – – – – – –

z z f h d z z z z – – – – – – – – –

z z z z z z z z z – – – – – – – – –

Pattern matching with mismatches

1D pattern:

```
a b c a b
```

1D text:

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

3 – 0 – – 0 – – 1 – – 0 – – – – –
```

What if we allow up to k mismatches?

for example, $k = 3$ mismatches

2D pattern:

```
a b c
d e a
f h d
```

2D text:

```
z z a b z z z z z    – – – – – – – – –
z z z e a b c z z    – – – – ✓ – – – –
z z a h d e a z z    – – – – – – – – –
z z z z f h d z z    – – – – – – – – –
a b c z z z a b c    ✓ – – – – – – – –
d e a b c z d z a    – – ✓ – – – – – –
f h d e a z f h d    – – – – – – – – –
z z f h d z z z z    – – – – – – – – –
z z z z z z z z z    – – – – – – – – –
```

Pattern matching with mismatches

1D pattern:

a b c a b

1D text:

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

3 – 0 – – 0 – – 1 – – 0 – – – – –

What if we allow up to k mismatches?

for example, $k = 3$ mismatches

2D pattern:

a b c
d e a
f h d

2D text:

```
z z a b z z z z z    – – – – – – – – –
z z z e a b c z z    – – – – ✓ – – – –
z z a h d e a z z    – – – – – – – – –
z z z z f h d z z    – – – – – – – – –
a b c z z z a b c    ✓ – – – – – – – –
d e a b c z d z a    – – ✓ – – – – – –
f h d e a z f h d    – – – – – – – – –
z z f h d z z z z    – – – – – – – – –
z z z z z z z z z    – – – – – – – – –
```

Pattern matching with mismatches

1D pattern: a b c a b

1D text: a b a b c a b c a b b a b c a b a

3 – 0 – – 0 – – 1 – – 0 – – – – –

What if we allow up to k mismatches?

for example, $k = 3$ mismatches

2D pattern:

a	b	c
d	e	a
f	h	d

2D text:

z	z	a	b	z	z	z	z	z
z	z	z	e	a	b	c	z	z
z	z	a	h	d	e	a	z	z
z	z	z	z	f	h	d	z	z
a	b	c	z	z	z	a	b	c
d	e	a	b	c	z	d	z	a
f	h	d	e	a	z	f	h	d
z	z	f	h	d	z	z	z	z
z	z	z	z	z	z	z	z	z

–	–	3	–	–	–	–	–	–
–	–	–	–	0	–	–	–	–
–	–	–	–	–	–	–	–	–
–	–	–	–	–	–	–	–	–
0	–	–	–	–	–	1	–	–
–	–	0	–	–	–	–	–	–
–	–	–	–	–	–	–	–	–
–	–	–	–	–	–	–	–	–
–	–	–	–	–	–	–	–	–

Pattern matching with mismatches

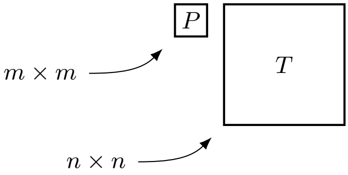
Assume that $n = \left\lfloor \frac{3m}{2} \right\rfloor$, otherwise solve $\mathcal{O}(n/m)$ instances of size $\left\lfloor \frac{3m}{2} \right\rfloor$ (in 1D),

or $\mathcal{O}(n^2/m^2)$ instances of size $\left\lfloor \frac{3m}{2} \right\rfloor \times \left\lfloor \frac{3m}{2} \right\rfloor$ (in 2D).

Pattern matching with mismatches

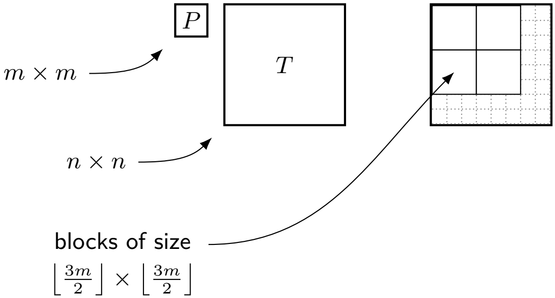
Assume that $n = \left\lfloor \frac{3m}{2} \right\rfloor$, otherwise solve $\mathcal{O}(n/m)$ instances of size $\left\lfloor \frac{3m}{2} \right\rfloor$ (in 1D),

or $\mathcal{O}(n^2/m^2)$ instances of size $\left\lfloor \frac{3m}{2} \right\rfloor \times \left\lfloor \frac{3m}{2} \right\rfloor$ (in 2D).



Pattern matching with mismatches

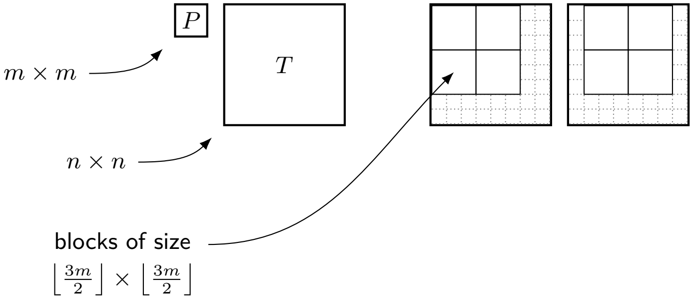
Assume that $n = \left\lfloor \frac{3m}{2} \right\rfloor$, otherwise solve $\mathcal{O}(n/m)$ instances of size $\left\lfloor \frac{3m}{2} \right\rfloor$ (in 1D),
or $\mathcal{O}(n^2/m^2)$ instances of size $\left\lfloor \frac{3m}{2} \right\rfloor \times \left\lfloor \frac{3m}{2} \right\rfloor$ (in 2D).



Pattern matching with mismatches

Assume that $n = \left\lfloor \frac{3m}{2} \right\rfloor$, otherwise solve $\mathcal{O}(n/m)$ instances of size $\left\lfloor \frac{3m}{2} \right\rfloor$ (in 1D),

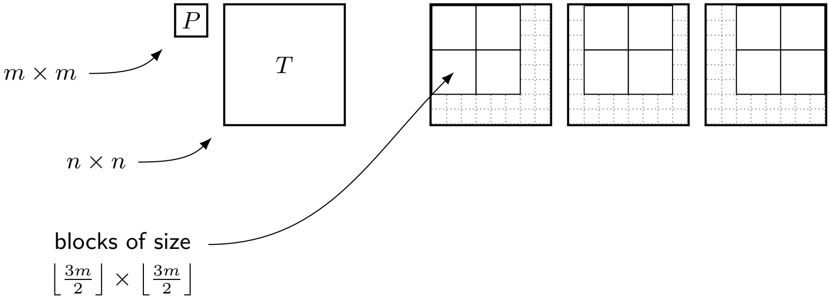
or $\mathcal{O}(n^2/m^2)$ instances of size $\left\lfloor \frac{3m}{2} \right\rfloor \times \left\lfloor \frac{3m}{2} \right\rfloor$ (in 2D).



Pattern matching with mismatches

Assume that $n = \left\lfloor \frac{3m}{2} \right\rfloor$, otherwise solve $\mathcal{O}(n/m)$ instances of size $\left\lfloor \frac{3m}{2} \right\rfloor$ (in 1D),

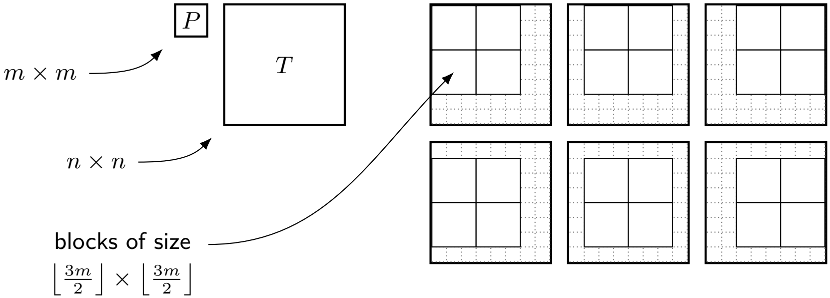
or $\mathcal{O}(n^2/m^2)$ instances of size $\left\lfloor \frac{3m}{2} \right\rfloor \times \left\lfloor \frac{3m}{2} \right\rfloor$ (in 2D).



Pattern matching with mismatches

Assume that $n = \left\lfloor \frac{3m}{2} \right\rfloor$, otherwise solve $\mathcal{O}(n/m)$ instances of size $\left\lfloor \frac{3m}{2} \right\rfloor$ (in 1D),

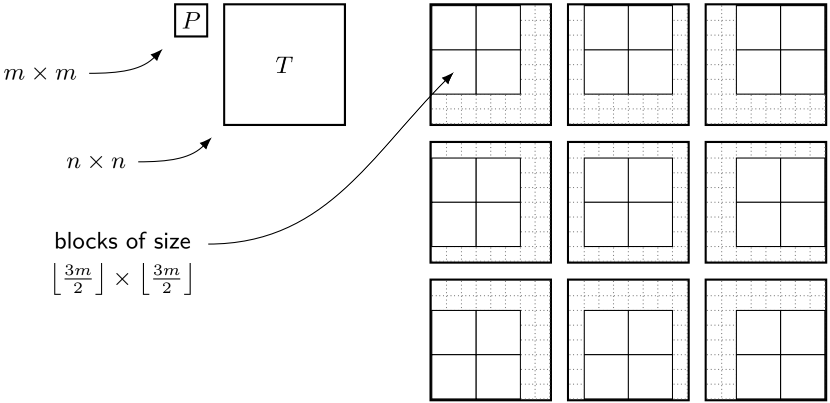
or $\mathcal{O}(n^2/m^2)$ instances of size $\left\lfloor \frac{3m}{2} \right\rfloor \times \left\lfloor \frac{3m}{2} \right\rfloor$ (in 2D).



Pattern matching with mismatches

Assume that $n = \left\lfloor \frac{3m}{2} \right\rfloor$, otherwise solve $\mathcal{O}(n/m)$ instances of size $\left\lfloor \frac{3m}{2} \right\rfloor$ (in 1D),

or $\mathcal{O}(n^2/m^2)$ instances of size $\left\lfloor \frac{3m}{2} \right\rfloor \times \left\lfloor \frac{3m}{2} \right\rfloor$ (in 2D).



Pattern matching with mismatches

Assume that $n = \left\lfloor \frac{3m}{2} \right\rfloor$, otherwise solve $\mathcal{O}(n/m)$ instances of size $\left\lfloor \frac{3m}{2} \right\rfloor$ (in 1D),

or $\mathcal{O}(n^2/m^2)$ instances of size $\left\lfloor \frac{3m}{2} \right\rfloor \times \left\lfloor \frac{3m}{2} \right\rfloor$ (in 2D).

Pattern matching with mismatches

Assume that $n = \left\lfloor \frac{3m}{2} \right\rfloor$, otherwise solve $\mathcal{O}(n/m)$ instances of size $\left\lfloor \frac{3m}{2} \right\rfloor$ (in 1D),

or $\mathcal{O}(n^2/m^2)$ instances of size $\left\lfloor \frac{3m}{2} \right\rfloor \times \left\lfloor \frac{3m}{2} \right\rfloor$ (in 2D).

		1D time	2D time	
exact PM	[Knuth, Morris, Pratt 77]	$\mathcal{O}(m)$	$\mathcal{O}(m^2)$	[Bird 77]

Pattern matching with mismatches

Assume that $n = \left\lfloor \frac{3m}{2} \right\rfloor$, otherwise solve $\mathcal{O}(n/m)$ instances of size $\left\lfloor \frac{3m}{2} \right\rfloor$ (in 1D),
or $\mathcal{O}(n^2/m^2)$ instances of size $\left\lfloor \frac{3m}{2} \right\rfloor \times \left\lfloor \frac{3m}{2} \right\rfloor$ (in 2D).

		1D time	2D time	
exact PM	[Knuth, Morris, Pratt 77]	$\mathcal{O}(m)$	$\mathcal{O}(m^2)$	[Bird 77]
∞-mismatch	[Fischer, Paterson 74]	$\tilde{\mathcal{O}}(m \cdot \lvert\Sigma\rvert)$	$\tilde{\mathcal{O}}(m^2 \cdot \lvert\Sigma\rvert)$	[Fischer, Paterson 74]

Pattern matching with mismatches

Assume that $n = \left\lfloor \frac{3m}{2} \right\rfloor$, otherwise solve $\mathcal{O}(n/m)$ instances of size $\left\lfloor \frac{3m}{2} \right\rfloor$ (in 1D),
or $\mathcal{O}(n^2/m^2)$ instances of size $\left\lfloor \frac{3m}{2} \right\rfloor \times \left\lfloor \frac{3m}{2} \right\rfloor$ (in 2D).

		1D time	2D time	
exact PM	[Knuth, Morris, Pratt 77]	$\mathcal{O}(m)$	$\mathcal{O}(m^2)$	[Bird 77]
∞-mismatch	[Fischer, Paterson 74]	$\tilde{\mathcal{O}}(m \cdot \lvert\Sigma\rvert)$	$\tilde{\mathcal{O}}(m^2 \cdot \lvert\Sigma\rvert)$	[Fischer, Paterson 74]
k-mismatch	[Landau, Vishkin 86]	$\mathcal{O}(mk)$	$\mathcal{O}(m^2 k)$	[Amir, Landau 91]

Pattern matching with mismatches

Assume that $n = \left\lfloor \frac{3m}{2} \right\rfloor$, otherwise solve $\mathcal{O}(n/m)$ instances of size $\left\lfloor \frac{3m}{2} \right\rfloor$ (in 1D),
or $\mathcal{O}(n^2/m^2)$ instances of size $\left\lfloor \frac{3m}{2} \right\rfloor \times \left\lfloor \frac{3m}{2} \right\rfloor$ (in 2D).

		1D time	2D time	
exact PM	[Knuth, Morris, Pratt 77]	$\mathcal{O}(m)$	$\mathcal{O}(m^2)$	[Bird 77]
∞-mismatch	[Fischer, Paterson 74]	$\tilde{\mathcal{O}}(m \cdot \lvert\Sigma\rvert)$	$\tilde{\mathcal{O}}(m^2 \cdot \lvert\Sigma\rvert)$	[Fischer, Paterson 74]
k-mismatch	[Landau, Vishkin 86]	$\mathcal{O}(mk)$	$\mathcal{O}(m^2 k)$	[Amir, Landau 91]
	[Amir, Lewenstein, Porat 04]	$\tilde{\mathcal{O}}(m\sqrt{k})$	$\tilde{\mathcal{O}}(m^2\sqrt{k})$?	

Pattern matching with mismatches

Assume that $n = \left\lfloor \frac{3m}{2} \right\rfloor$, otherwise solve $\mathcal{O}(n/m)$ instances of size $\left\lfloor \frac{3m}{2} \right\rfloor$ (in 1D),

or $\mathcal{O}(n^2/m^2)$ instances of size $\left\lfloor \frac{3m}{2} \right\rfloor \times \left\lfloor \frac{3m}{2} \right\rfloor$ (in 2D).

		1D time	2D time	
exact PM	[Knuth, Morris, Pratt 77]	$\mathcal{O}(m)$	$\mathcal{O}(m^2)$	[Bird 77]
∞-mismatch	[Fischer, Paterson 74]	$\tilde{\mathcal{O}}(m \cdot \|\Sigma\|)$	$\tilde{\mathcal{O}}(m^2 \cdot \|\Sigma\|)$	[Fischer, Paterson 74]
k-mismatch	[Landau, Vishkin 86]	$\mathcal{O}(mk)$	$\mathcal{O}(m^2 k)$	[Amir, Landau 91]
	[Amir, Lewenstein, Porat 04]	$\tilde{\mathcal{O}}(m\sqrt{k})$	$\tilde{\mathcal{O}}(m^2\sqrt{k})$?	
	[Clifford et al. 16]	$\tilde{\mathcal{O}}(m + k^2)$	$\tilde{\mathcal{O}}(m^2 + k^2)$?	

Pattern matching with mismatches

Assume that $n = \left\lfloor \frac{3m}{2} \right\rfloor$, otherwise solve $\mathcal{O}(n/m)$ instances of size $\left\lfloor \frac{3m}{2} \right\rfloor$ (in 1D),

or $\mathcal{O}(n^2/m^2)$ instances of size $\left\lfloor \frac{3m}{2} \right\rfloor \times \left\lfloor \frac{3m}{2} \right\rfloor$ (in 2D).

		1D time	2D time	
exact PM	[Knuth, Morris, Pratt 77]	$\mathcal{O}(m)$	$\mathcal{O}(m^2)$	[Bird 77]
∞-mismatch	[Fischer, Paterson 74]	$\tilde{\mathcal{O}}(m \cdot \lvert\Sigma\rvert)$	$\tilde{\mathcal{O}}(m^2 \cdot \lvert\Sigma\rvert)$	[Fischer, Paterson 74]
k-mismatch	[Landau, Vishkin 86]	$\mathcal{O}(mk)$	$\mathcal{O}(m^2 k)$	[Amir, Landau 91]
	[Amir, Lewenstein, Porat 04]	$\tilde{\mathcal{O}}(m\sqrt{k})$	$\tilde{\mathcal{O}}(m^2\sqrt{k})$?	
	[Clifford et al. 16]	$\tilde{\mathcal{O}}(m + k^2)$	$\tilde{\mathcal{O}}(m^2 + k^2)$?	
	[Gawrychowski, Uznanski 18]	$\tilde{\mathcal{O}}(m + k\sqrt{m})$	$\tilde{\mathcal{O}}(m^2 + km)$?	

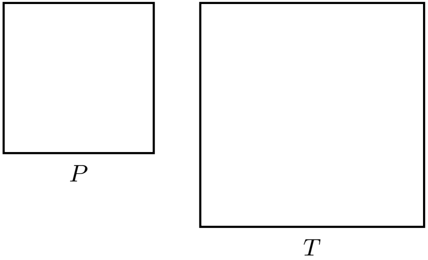
Pattern matching with mismatches

Assume that $n = \left\lfloor \frac{3m}{2} \right\rfloor$, otherwise solve $\mathcal{O}(n/m)$ instances of size $\left\lfloor \frac{3m}{2} \right\rfloor$ (in 1D),
or $\mathcal{O}(n^2/m^2)$ instances of size $\left\lfloor \frac{3m}{2} \right\rfloor \times \left\lfloor \frac{3m}{2} \right\rfloor$ (in 2D).

		1D time	2D time	
exact PM	[Knuth, Morris, Pratt 77]	$\mathcal{O}(m)$	$\mathcal{O}(m^2)$	[Bird 77]
∞-mismatch	[Fischer, Paterson 74]	$\tilde{\mathcal{O}}(m \cdot \lvert\Sigma\rvert)$	$\tilde{\mathcal{O}}(m^2 \cdot \lvert\Sigma\rvert)$	[Fischer, Paterson 74]
k-mismatch	[Landau, Vishkin 86]	$\mathcal{O}(mk)$	$\mathcal{O}(m^2k)$	[Amir, Landau 91]
	[Amir, Lewenstein, Porat 04]	$\tilde{\mathcal{O}}(m\sqrt{k})$	$\tilde{\mathcal{O}}(m^2\sqrt{k})$?	
	[Clifford et al. 16]	$\tilde{\mathcal{O}}(m+k^2)$	$\tilde{\mathcal{O}}(m^2+k^2)$?	$\boxed{\tilde{\mathcal{O}}(m^2+k^{5/4}m)}$
	[Gawrychowski, Uznanski 18]	$\tilde{\mathcal{O}}(m+k\sqrt{m})$	$\tilde{\mathcal{O}}(m^2+km)$?	

Almost solving the problem...

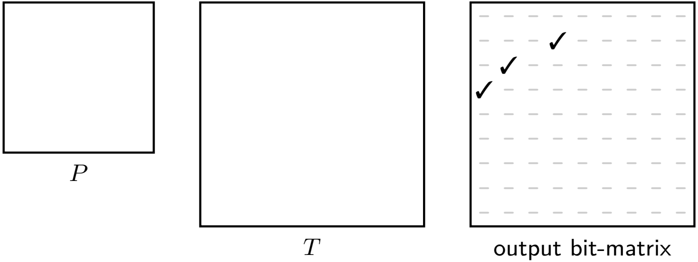
...is easy! Adapt Karloff's algorithm to 2D and compute the following in $\tilde{\mathcal{O}}(m^2)$ time:



$\Longrightarrow$ From now on, consider set $\mathcal{C}$ of candidate alignments ($=$ marked positions).

Almost solving the problem...

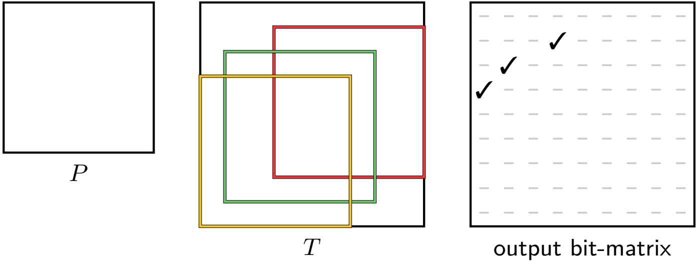
...is easy! Adapt Karloff's algorithm to 2D and compute the following in $\tilde{\mathcal{O}}(m^2)$ time:



$\Longrightarrow$ From now on, consider set $\mathcal{C}$ of candidate alignments (= marked positions).

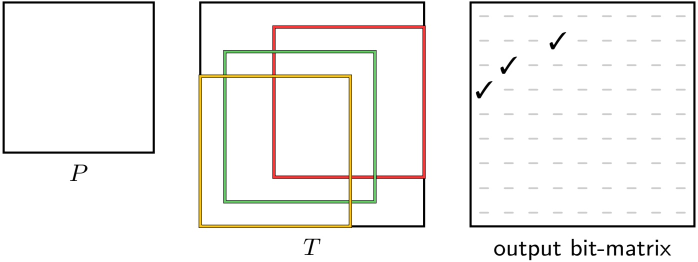
Almost solving the problem...

...is easy! Adapt Karloff's algorithm to 2D and compute the following in $\tilde{\mathcal{O}}(m^2)$ time:



Almost solving the problem...

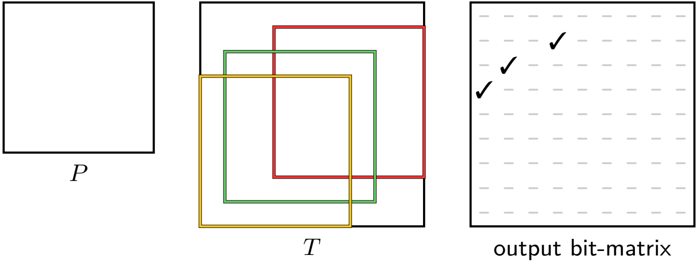
...is easy! Adapt Karloff's algorithm to 2D and compute the following in $\tilde{\mathcal{O}}(m^2)$ time:



- every k-mismatch occurrence is marked

Almost solving the problem...

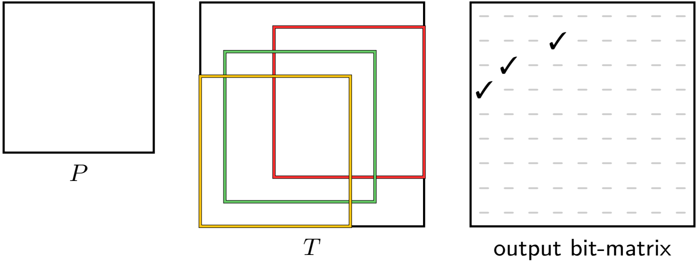
...is easy! Adapt Karloff's algorithm to 2D and compute the following in $\tilde{\mathcal{O}}(m^2)$ time:



- every k-mismatch occurrence is marked
- every marked position is a $2k$-mismatch occurrence

Almost solving the problem...

...is easy! Adapt Karloff's algorithm to 2D and compute the following in $\tilde{\mathcal{O}}(m^2)$ time:



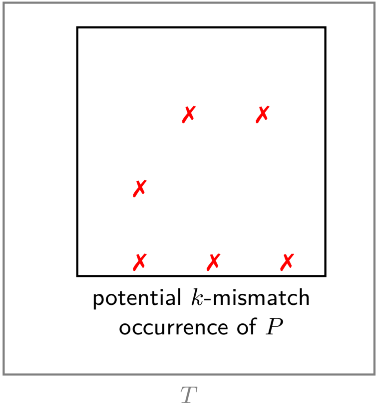
- every k-mismatch occurrence is marked
- every marked position is a $2k$-mismatch occurrence

$\implies$ From now on, consider set $\mathcal{C}$ of candidate alignments ($=$ marked positions).

Verifying candidate alignments

For a given candidate alignment, count up to k mismatches in $\mathcal{O}(k)$ time:

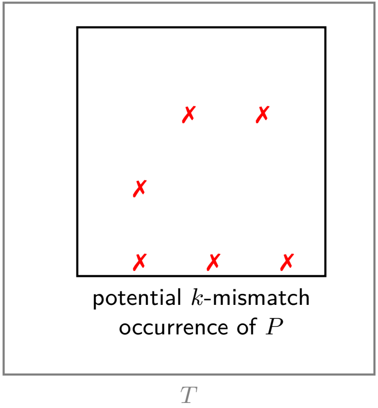
(use simple reduction to 1D and standard data structures like suffix tree)



Verifying candidate alignments

For a given candidate alignment, count up to k mismatches in $\mathcal{O}(k)$ time:

(use simple reduction to 1D and standard data structures like suffix tree)



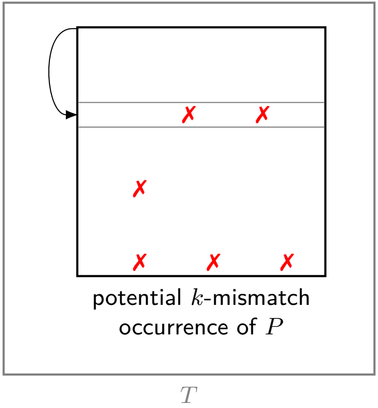
2D "kangaroo jumping":

- jump to next row that contains a mismatch in $\mathcal{O}(1)$ time

Verifying candidate alignments

For a given candidate alignment, count up to k mismatches in $\mathcal{O}(k)$ time:

(use simple reduction to 1D and standard data structures like suffix tree)



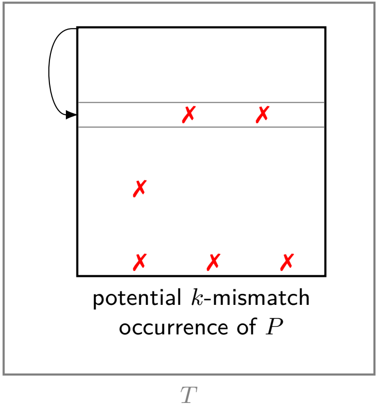
2D “kangaroo jumping”:

- jump to next row that contains a mismatch in $\mathcal{O}(1)$ time

Verifying candidate alignments

For a given candidate alignment, count up to k mismatches in $\mathcal{O}(k)$ time:

(use simple reduction to 1D and standard data structures like suffix tree)



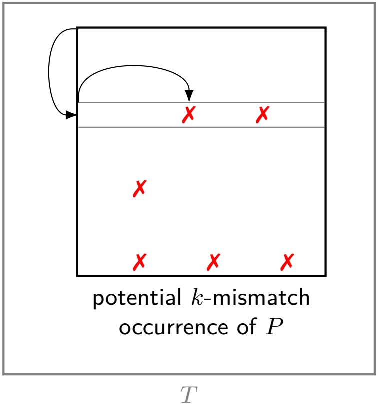
2D "kangaroo jumping":

- jump to next row that contains a mismatch in $\mathcal{O}(1)$ time
- jump to next mismatch within row in $\mathcal{O}(1)$ time

Verifying candidate alignments

For a given candidate alignment, count up to k mismatches in $\mathcal{O}(k)$ time:

(use simple reduction to 1D and standard data structures like suffix tree)



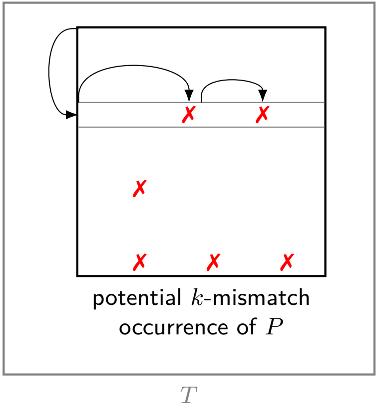
2D "kangaroo jumping":

- jump to next row that contains a mismatch in $\mathcal{O}(1)$ time
- jump to next mismatch within row in $\mathcal{O}(1)$ time

Verifying candidate alignments

For a given candidate alignment, count up to k mismatches in $\mathcal{O}(k)$ time:

(use simple reduction to 1D and standard data structures like suffix tree)



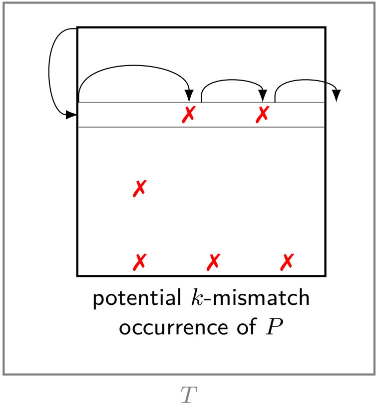
2D "kangaroo jumping":

- jump to next row that contains a mismatch in $\mathcal{O}(1)$ time
- jump to next mismatch within row in $\mathcal{O}(1)$ time

Verifying candidate alignments

For a given candidate alignment, count up to k mismatches in $\mathcal{O}(k)$ time:

(use simple reduction to 1D and standard data structures like suffix tree)



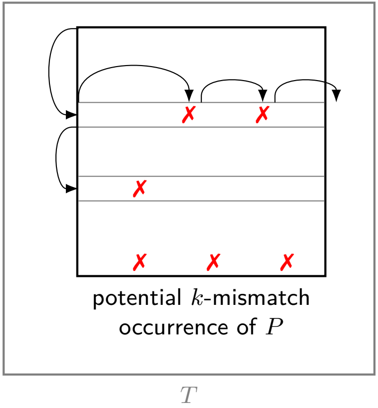
2D "kangaroo jumping":

- jump to next row that contains a mismatch in $\mathcal{O}(1)$ time
- jump to next mismatch within row in $\mathcal{O}(1)$ time

Verifying candidate alignments

For a given candidate alignment, count up to k mismatches in $\mathcal{O}(k)$ time:

(use simple reduction to 1D and standard data structures like suffix tree)



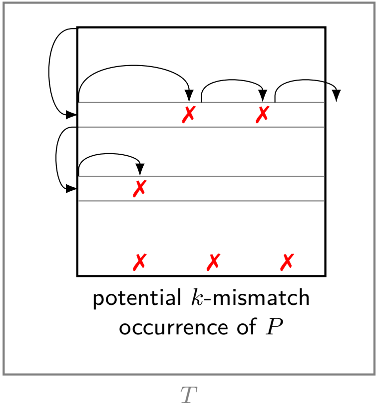
2D "kangaroo jumping":

- jump to next row that contains a mismatch in $\mathcal{O}(1)$ time
- jump to next mismatch within row in $\mathcal{O}(1)$ time

Verifying candidate alignments

For a given candidate alignment, count up to k mismatches in $\mathcal{O}(k)$ time:

(use simple reduction to 1D and standard data structures like suffix tree)



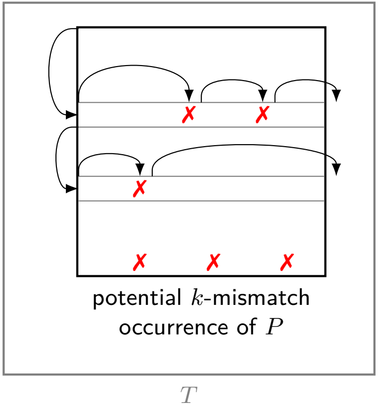
2D "kangaroo jumping":

- jump to next row that contains a mismatch in $\mathcal{O}(1)$ time
- jump to next mismatch within row in $\mathcal{O}(1)$ time

Verifying candidate alignments

For a given candidate alignment, count up to k mismatches in $\mathcal{O}(k)$ time:

(use simple reduction to 1D and standard data structures like suffix tree)



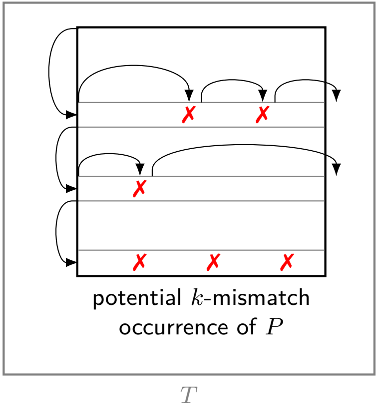
2D "kangaroo jumping":

- jump to next row that contains a mismatch in $\mathcal{O}(1)$ time
- jump to next mismatch within row in $\mathcal{O}(1)$ time

Verifying candidate alignments

For a given candidate alignment, count up to k mismatches in $\mathcal{O}(k)$ time:

(use simple reduction to 1D and standard data structures like suffix tree)



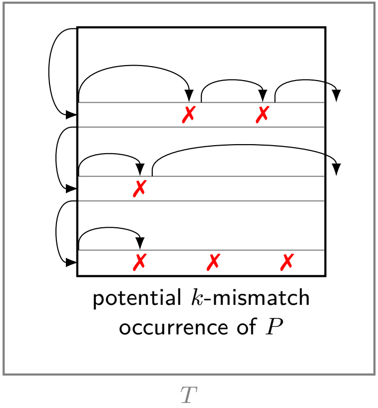
2D "kangaroo jumping":

- jump to next row that contains a mismatch in $\mathcal{O}(1)$ time
- jump to next mismatch within row in $\mathcal{O}(1)$ time

Verifying candidate alignments

For a given candidate alignment, count up to k mismatches in $\mathcal{O}(k)$ time:

(use simple reduction to 1D and standard data structures like suffix tree)



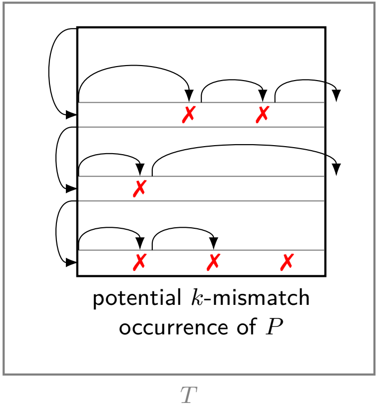
2D "kangaroo jumping":

- jump to next row that contains a mismatch in $\mathcal{O}(1)$ time
- jump to next mismatch within row in $\mathcal{O}(1)$ time

Verifying candidate alignments

For a given candidate alignment, count up to k mismatches in $\mathcal{O}(k)$ time:

(use simple reduction to 1D and standard data structures like suffix tree)



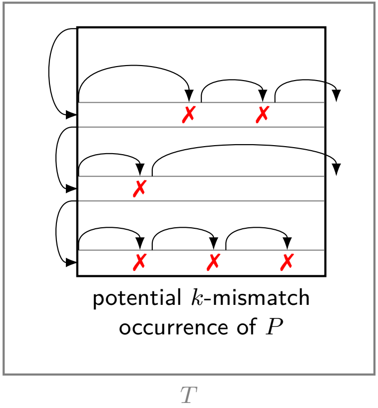
2D "kangaroo jumping":

- jump to next row that contains a mismatch in $\mathcal{O}(1)$ time
- jump to next mismatch within row in $\mathcal{O}(1)$ time

Verifying candidate alignments

For a given candidate alignment, count up to k mismatches in $\mathcal{O}(k)$ time:

(use simple reduction to 1D and standard data structures like suffix tree)



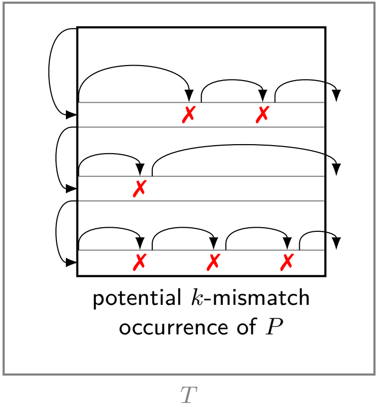
2D "kangaroo jumping":

- jump to next row that contains a mismatch in $\mathcal{O}(1)$ time
- jump to next mismatch within row in $\mathcal{O}(1)$ time

Verifying candidate alignments

For a given candidate alignment, count up to k mismatches in $\mathcal{O}(k)$ time:

(use simple reduction to 1D and standard data structures like suffix tree)



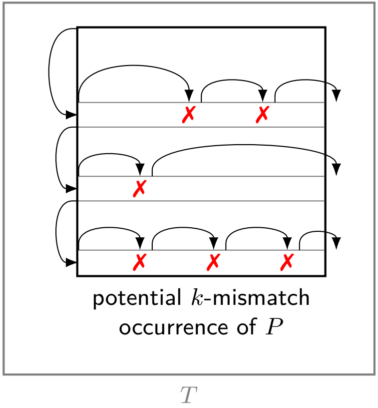
2D "kangaroo jumping":

- jump to next row that contains a mismatch in $\mathcal{O}(1)$ time
- jump to next mismatch within row in $\mathcal{O}(1)$ time

Verifying candidate alignments

For a given candidate alignment, count up to k mismatches in $\mathcal{O}(k)$ time:

(use simple reduction to 1D and standard data structures like suffix tree)



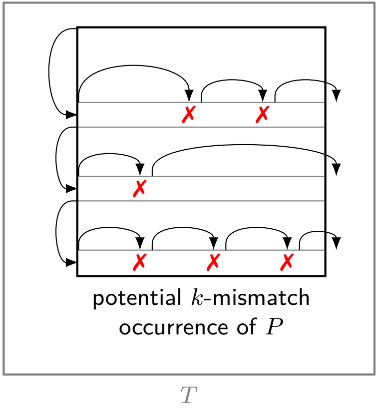
2D "kangaroo jumping":

- jump to next row that contains a mismatch in $\mathcal{O}(1)$ time
- jump to next mismatch within row in $\mathcal{O}(1)$ time
- at most $\mathcal{O}(k)$ steps due to previous filtering

Verifying candidate alignments

For a given candidate alignment, count up to k mismatches in $\mathcal{O}(k)$ time:

(use simple reduction to 1D and standard data structures like suffix tree)



2D "kangaroo jumping":

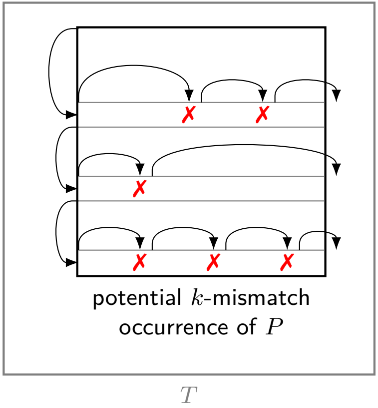
- jump to next row that contains a mismatch in $\mathcal{O}(1)$ time
- jump to next mismatch within row in $\mathcal{O}(1)$ time
- at most $\mathcal{O}(k)$ steps due to previous filtering

$\Longrightarrow$ run for all candidate alignment in $\mathcal{O}(|\mathcal{C}| \cdot k)$ time

Verifying candidate alignments

For a given candidate alignment, count up to k mismatches in $\mathcal{O}(k)$ time:

(use simple reduction to 1D and standard data structures like suffix tree)



2D "kangaroo jumping":

- jump to next row that contains a mismatch in $\mathcal{O}(1)$ time
- jump to next mismatch within row in $\mathcal{O}(1)$ time
- at most $\mathcal{O}(k)$ steps due to previous filtering

$\Longrightarrow$ run for all candidate alignment in $\mathcal{O}(|\mathcal{C}| \cdot k)$ time

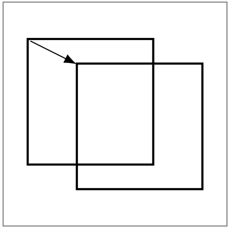
$\Longrightarrow$ if $|\mathcal{C}| = \mathcal{O}(m^2/k + m)$, then overall $\mathcal{O}(m^2 + km)$ time

Many candidates imply periodicity

We're good unless $|\mathcal{C}| = \Omega(m^2/k + m)$. For now, assume that there are no mismatches.

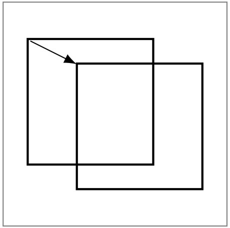
Many candidates imply periodicity

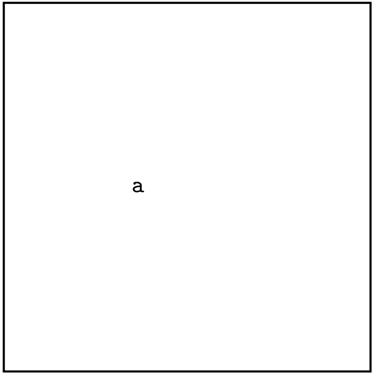
We're good unless $|\mathcal{C}| = \Omega(m^2/k + m)$. For now, assume that there are no mismatches.



Many candidates imply periodicity

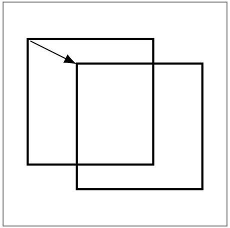
We're good unless $|\mathcal{C}| = \Omega(m^2/k + m)$. For now, assume that there are no mismatches.

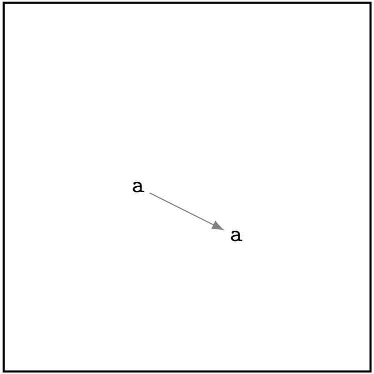




Many candidates imply periodicity

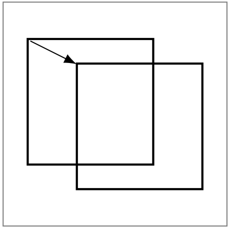
We're good unless $|\mathcal{C}| = \Omega(m^2/k + m)$. For now, assume that there are no mismatches.

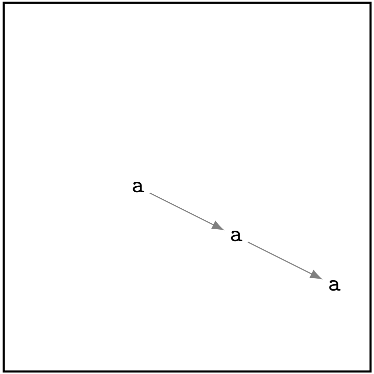




Many candidates imply periodicity

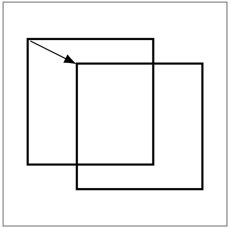
We're good unless $|\mathcal{C}| = \Omega(m^2/k + m)$. For now, assume that there are no mismatches.

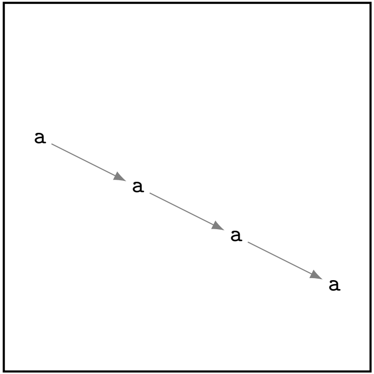




Many candidates imply periodicity

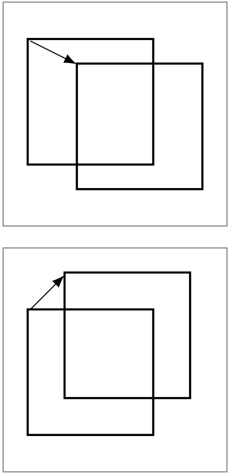
We're good unless $|\mathcal{C}| = \Omega(m^2/k + m)$. For now, assume that there are no mismatches.

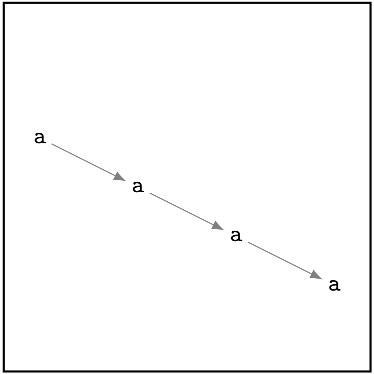




Many candidates imply periodicity

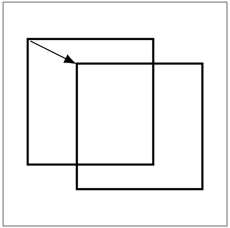
We're good unless $|\mathcal{C}| = \Omega(m^2/k + m)$. For now, assume that there are no mismatches.

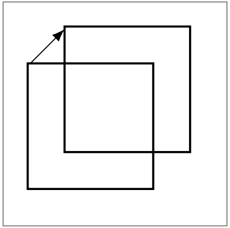


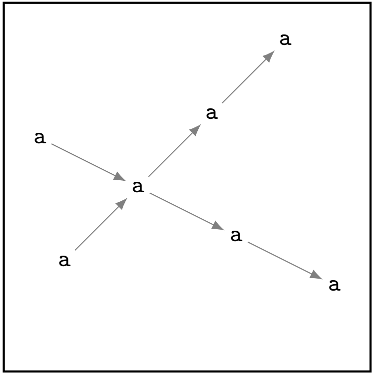


Many candidates imply periodicity

We're good unless $|\mathcal{C}| = \Omega(m^2/k + m)$. For now, assume that there are no mismatches.

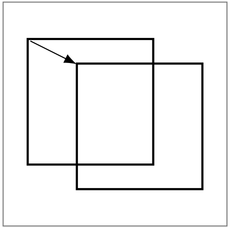


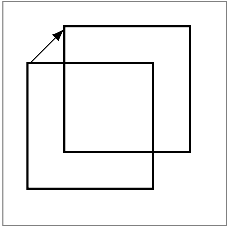


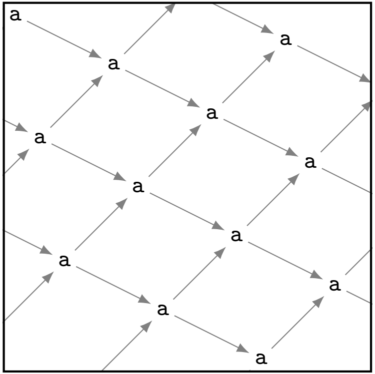


Many candidates imply periodicity

We're good unless $|\mathcal{C}| = \Omega(m^2/k + m)$. For now, assume that there are no mismatches.

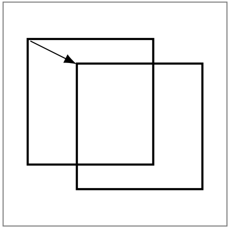


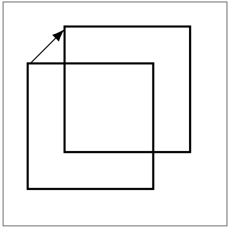


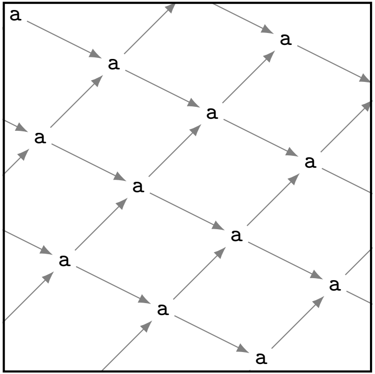


Many candidates imply periodicity

We're good unless $|\mathcal{C}| = \Omega(m^2/k + m)$. For now, assume that there are no mismatches.



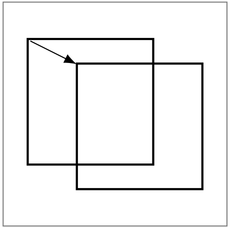


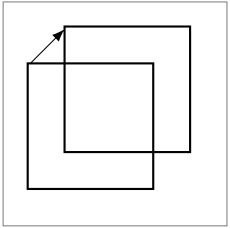


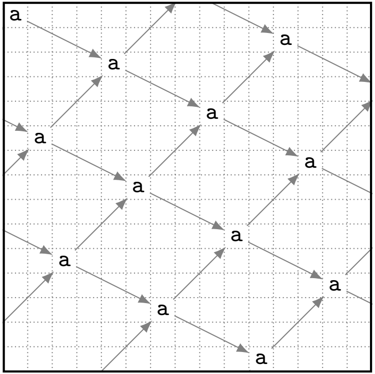
- if self-overlaps with no mismatches, then pattern is repeating "diamond"

Many candidates imply periodicity

We're good unless $|\mathcal{C}| = \Omega(m^2/k + m)$. For now, assume that there are no mismatches.



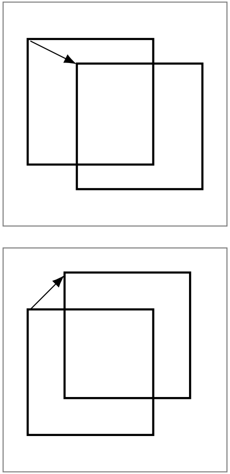


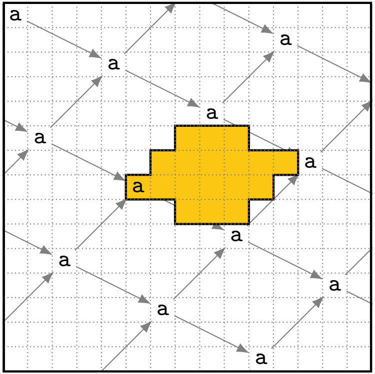


- if self-overlaps with no mismatches, then pattern is repeating "diamond"

Many candidates imply periodicity

We're good unless $|\mathcal{C}| = \Omega(m^2/k + m)$. For now, assume that there are no mismatches.

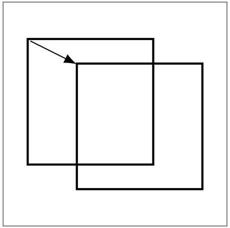


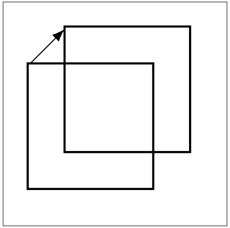


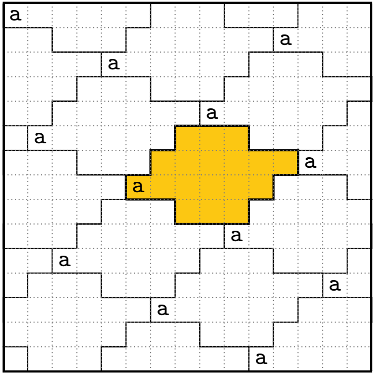
- if self-overlaps with no mismatches, then pattern is repeating "diamond"

Many candidates imply periodicity

We're good unless $|\mathcal{C}| = \Omega(m^2/k + m)$. For now, assume that there are no mismatches.



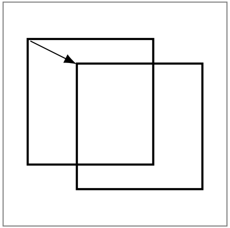


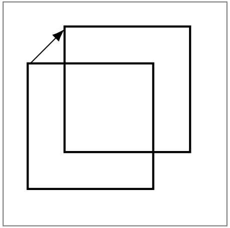


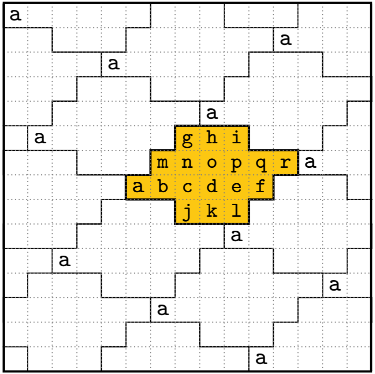
- if self-overlaps with no mismatches, then pattern is repeating "diamond"

Many candidates imply periodicity

We're good unless $|\mathcal{C}| = \Omega(m^2/k + m)$. For now, assume that there are no mismatches.



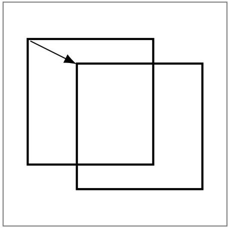


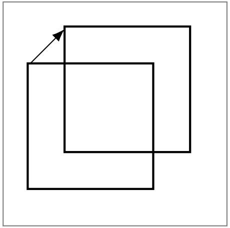


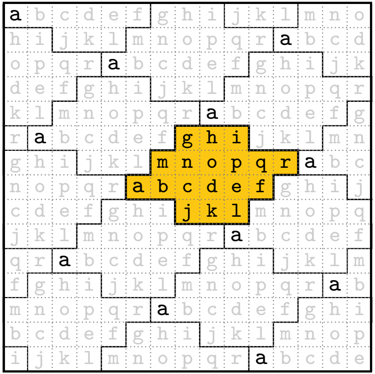
- if self-overlaps with no mismatches, then pattern is repeating "diamond"

Many candidates imply periodicity

We're good unless $|\mathcal{C}| = \Omega(m^2/k + m)$. For now, assume that there are no mismatches.



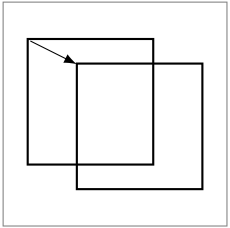


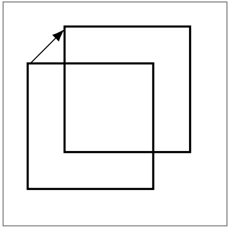


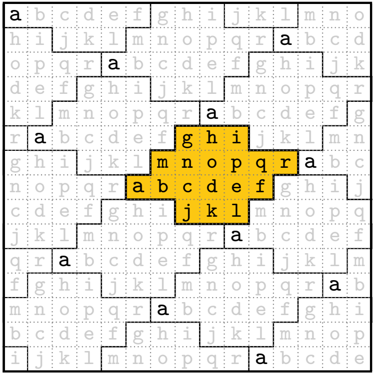
- if self-overlaps with no mismatches, then pattern is repeating "diamond"

Many candidates imply periodicity

We're good unless $|\mathcal{C}| = \Omega(m^2/k + m)$. For now, assume that there are no mismatches.



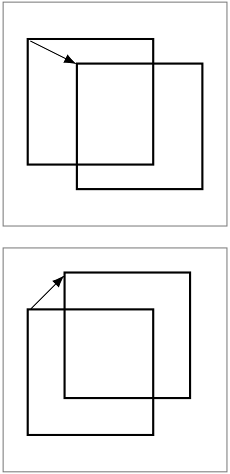


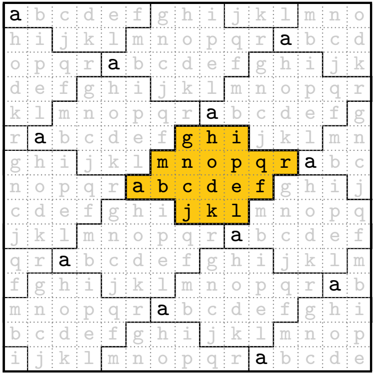


- if self-overlaps with no mismatches, then pattern is repeating "diamond"
- due to number $|\mathcal{C}|$ of candidates, diamond is of size $\mathcal{O}(\min(m, k))$

Many candidates imply periodicity

We're good unless $|\mathcal{C}| = \Omega(m^2/k + m)$. For now, assume that there are no mismatches.

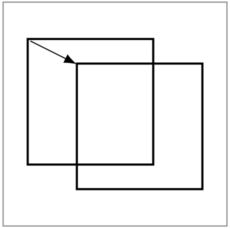


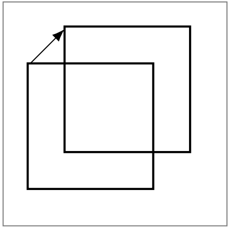


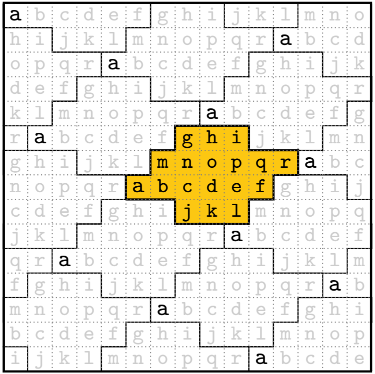
- if self-overlaps with no mismatches, then pattern is repeating "diamond"
- due to number $|\mathcal{C}|$ of candidates, diamond is of size $\mathcal{O}(\min(m, k))$
- diamond is nearly rectangular

Many candidates imply periodicity

We're good unless $|\mathcal{C}| = \Omega(m^2/k + m)$. For now, assume that there are no mismatches.





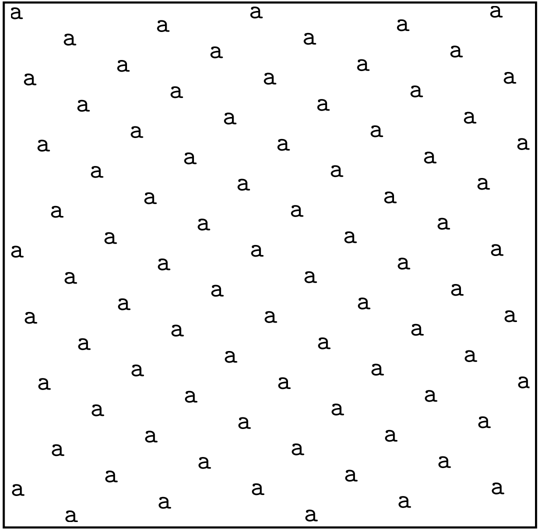


- if self-overlaps with no mismatches, then pattern is repeating "diamond"
- due to number $|\mathcal{C}|$ of candidates, diamond is of size $\mathcal{O}(\min(m, k))$
- diamond is nearly rectangular

$\implies$ at most $\mathcal{O}(\min(m, k))$ single symbol "subsequences"

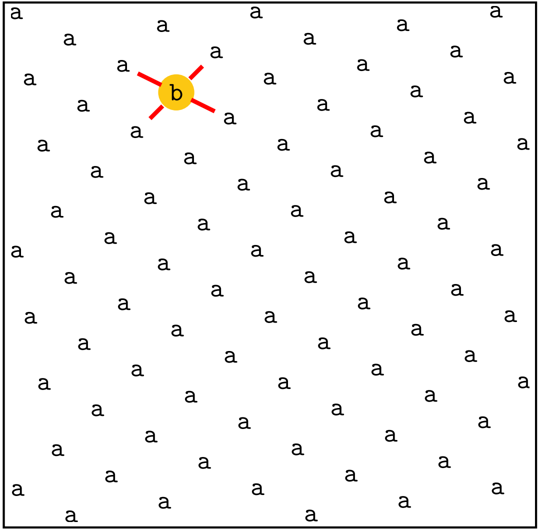
Splitting pattern subsequences

- diamond leads to $\mathcal{O}(\min(m, k))$ subsequences



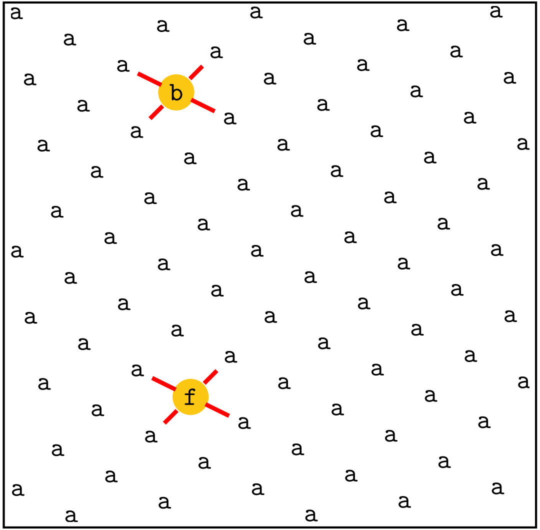
Splitting pattern subsequences

- diamond leads to $\mathcal{O}(\min(m,k))$ subsequences



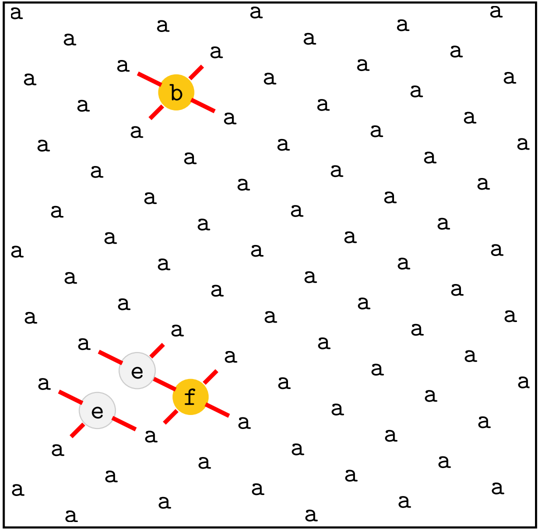
Splitting pattern subsequences

- diamond leads to $\mathcal{O}(\min(m, k))$ subsequences



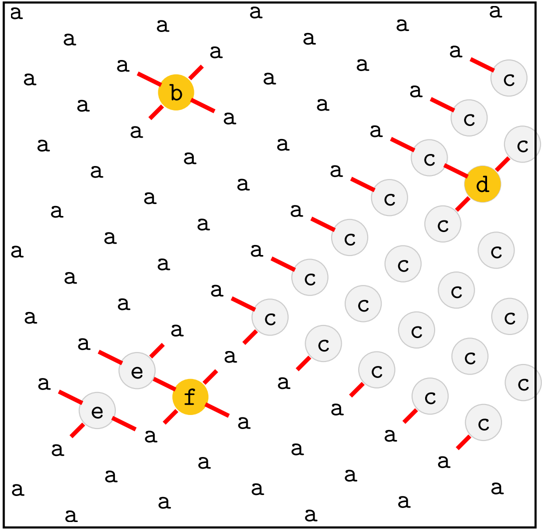
Splitting pattern subsequences

- diamond leads to $\mathcal{O}(\min(m, k))$ subsequences



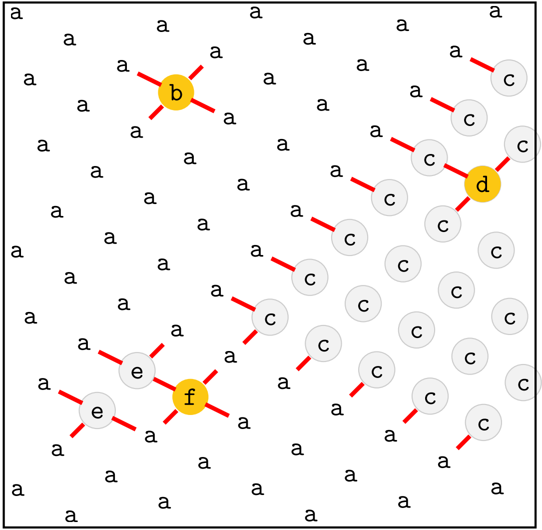
Splitting pattern subsequences

- diamond leads to $\mathcal{O}(\min(m, k))$ subsequences



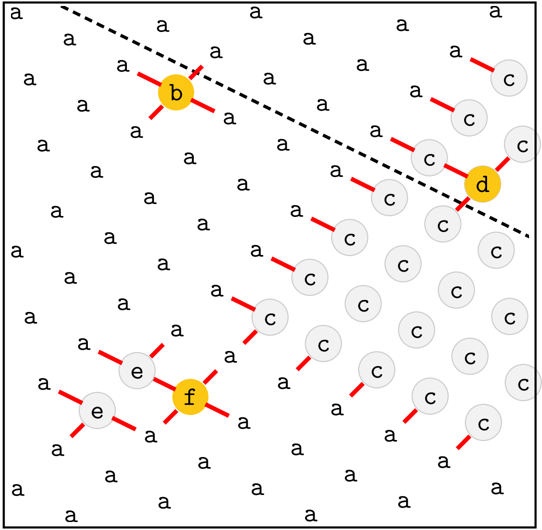
Splitting pattern subsequences

- diamond leads to $\mathcal{O}(\min(m,k))$ subsequences
- $\mathcal{O}(k)$ mismatches of adjacent symbols (with respect to subsequences)



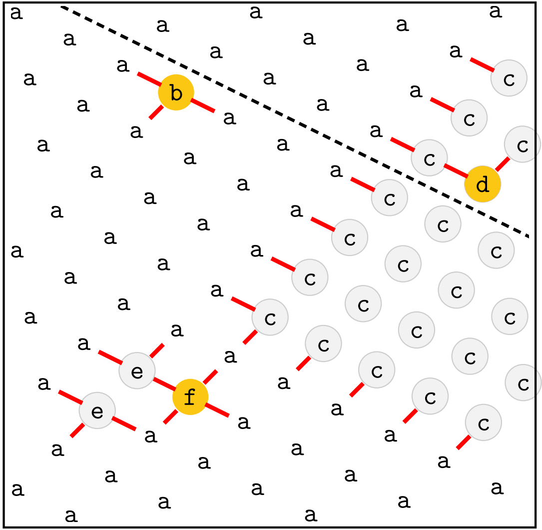
Splitting pattern subsequences

- diamond leads to $\mathcal{O}(\min(m,k))$ subsequences
- $\mathcal{O}(k)$ mismatches of adjacent symbols (with respect to subsequences)



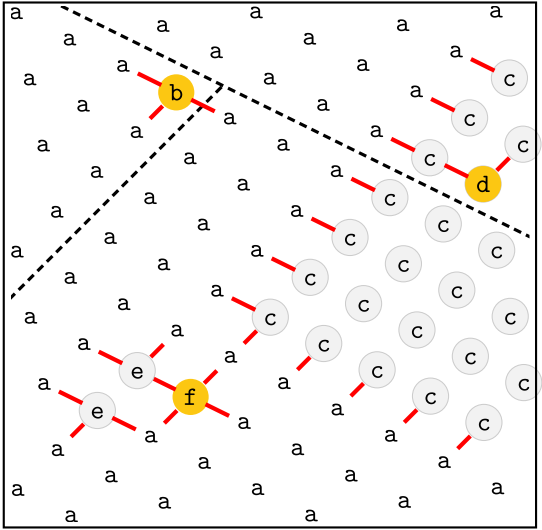
Splitting pattern subsequences

- diamond leads to $\mathcal{O}(\min(m, k))$ subsequences
- $\mathcal{O}(k)$ mismatches of adjacent symbols (with respect to subsequences)



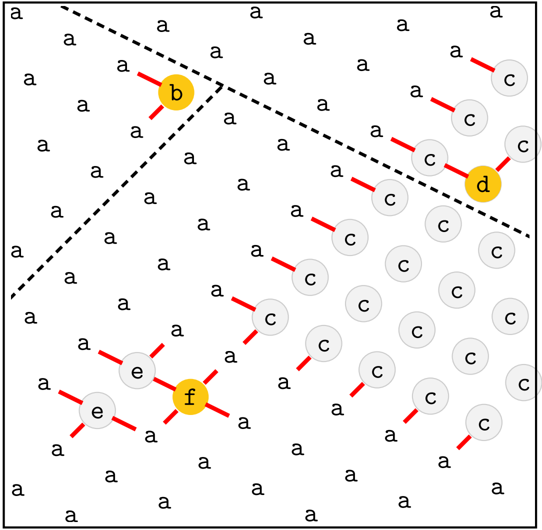
Splitting pattern subsequences

- diamond leads to $\mathcal{O}(\min(m, k))$ subsequences
- $\mathcal{O}(k)$ mismatches of adjacent symbols (with respect to subsequences)



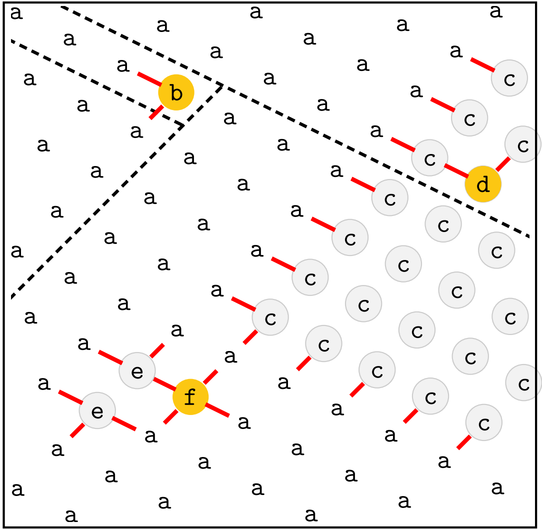
Splitting pattern subsequences

- diamond leads to $\mathcal{O}(\min(m,k))$ subsequences
- $\mathcal{O}(k)$ mismatches of adjacent symbols (with respect to subsequences)



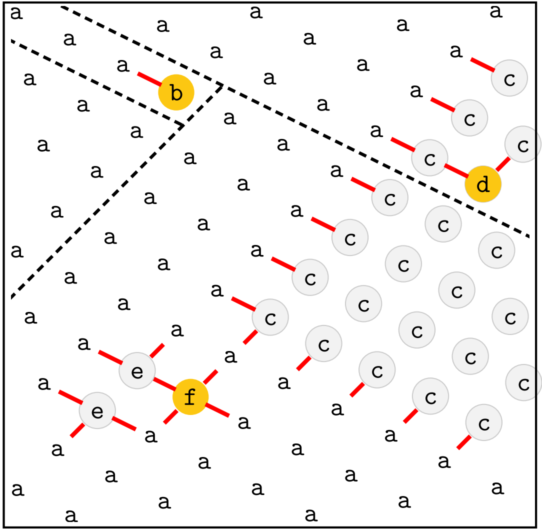
Splitting pattern subsequences

- diamond leads to $\mathcal{O}(\min(m, k))$ subsequences
- $\mathcal{O}(k)$ mismatches of adjacent symbols (with respect to subsequences)



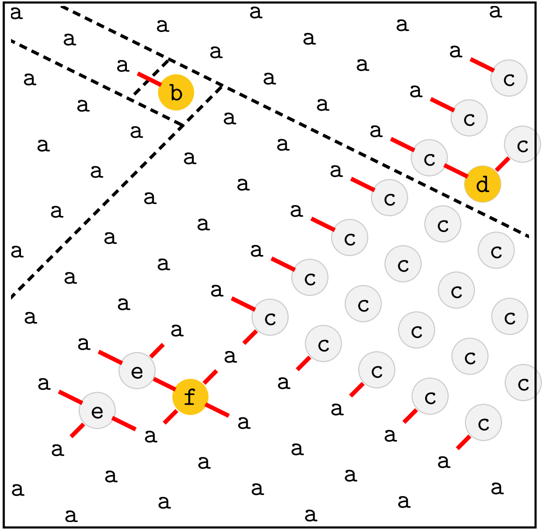
Splitting pattern subsequences

- diamond leads to $\mathcal{O}(\min(m, k))$ subsequences
- $\mathcal{O}(k)$ mismatches of adjacent symbols (with respect to subsequences)



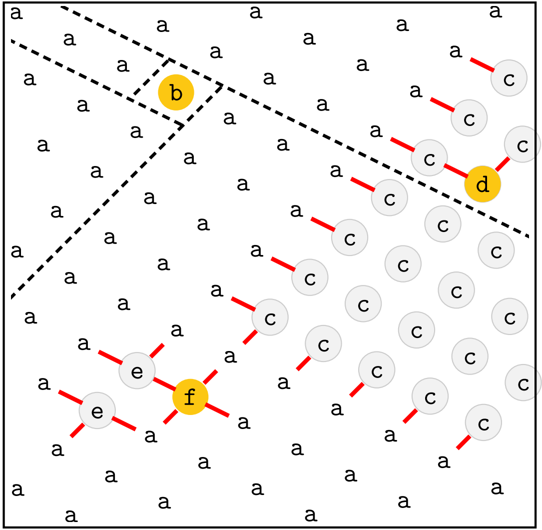
Splitting pattern subsequences

- diamond leads to $\mathcal{O}(\min(m, k))$ subsequences
- $\mathcal{O}(k)$ mismatches of adjacent symbols (with respect to subsequences)



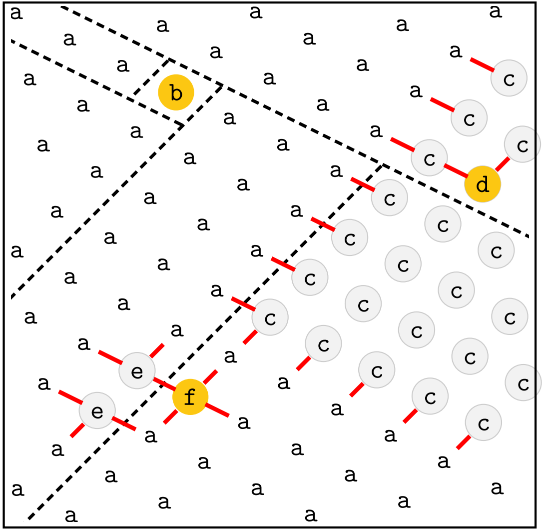
Splitting pattern subsequences

- diamond leads to $\mathcal{O}(\min(m, k))$ subsequences
- $\mathcal{O}(k)$ mismatches of adjacent symbols (with respect to subsequences)



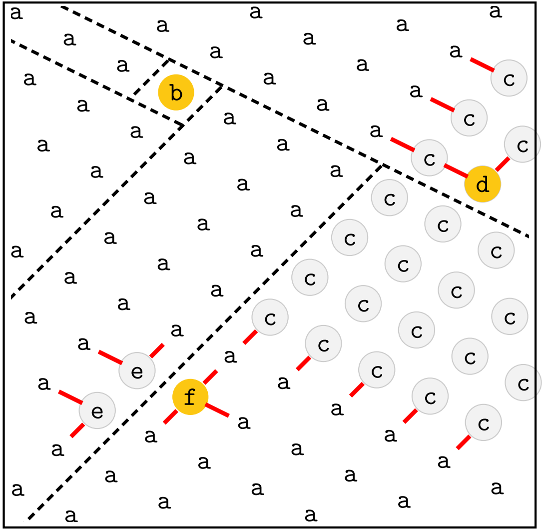
Splitting pattern subsequences

- diamond leads to $\mathcal{O}(\min(m, k))$ subsequences
- $\mathcal{O}(k)$ mismatches of adjacent symbols (with respect to subsequences)



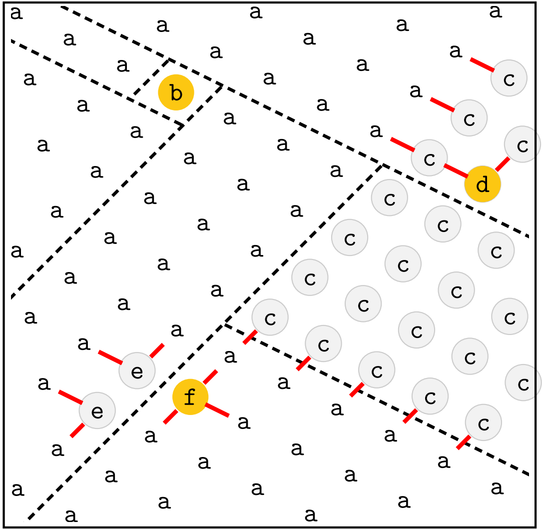
Splitting pattern subsequences

- diamond leads to $\mathcal{O}(\min(m, k))$ subsequences
- $\mathcal{O}(k)$ mismatches of adjacent symbols (with respect to subsequences)



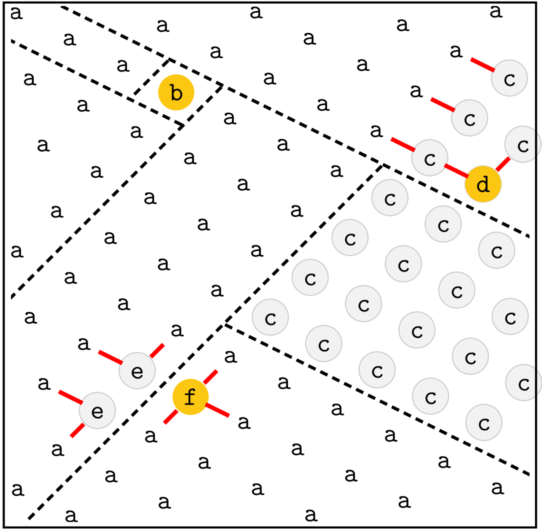
Splitting pattern subsequences

- diamond leads to $\mathcal{O}(\min(m, k))$ subsequences
- $\mathcal{O}(k)$ mismatches of adjacent symbols (with respect to subsequences)



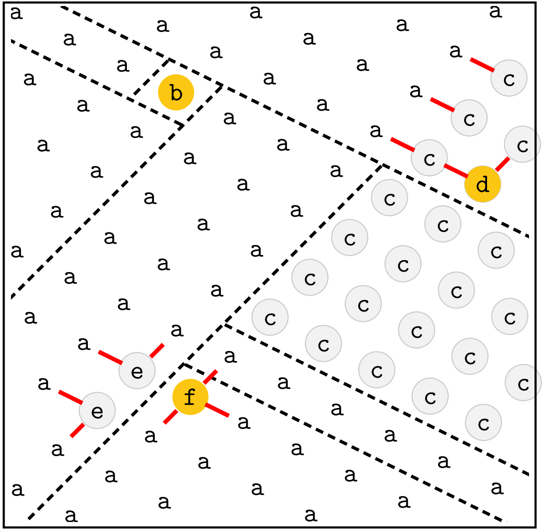
Splitting pattern subsequences

- diamond leads to $\mathcal{O}(\min(m, k))$ subsequences
- $\mathcal{O}(k)$ mismatches of adjacent symbols (with respect to subsequences)



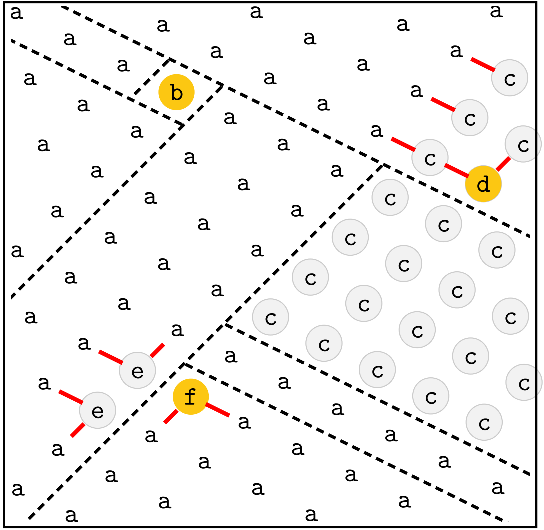
Splitting pattern subsequences

- diamond leads to $\mathcal{O}(\min(m, k))$ subsequences
- $\mathcal{O}(k)$ mismatches of adjacent symbols (with respect to subsequences)



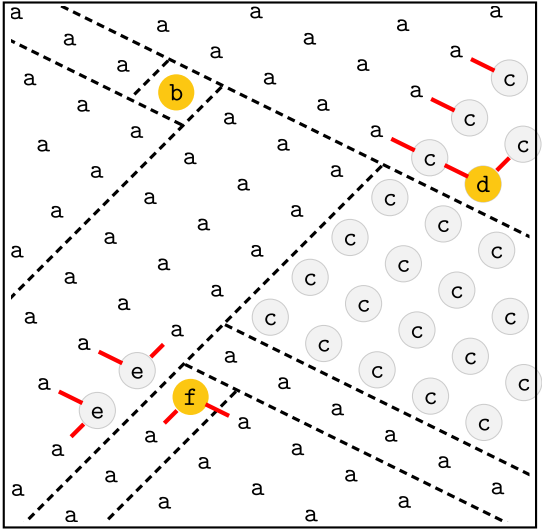
Splitting pattern subsequences

- diamond leads to $\mathcal{O}(\min(m, k))$ subsequences
- $\mathcal{O}(k)$ mismatches of adjacent symbols (with respect to subsequences)



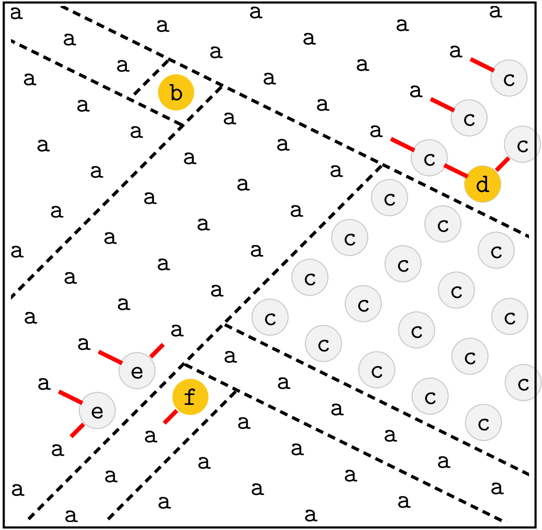
Splitting pattern subsequences

- diamond leads to $\mathcal{O}(\min(m, k))$ subsequences
- $\mathcal{O}(k)$ mismatches of adjacent symbols (with respect to subsequences)



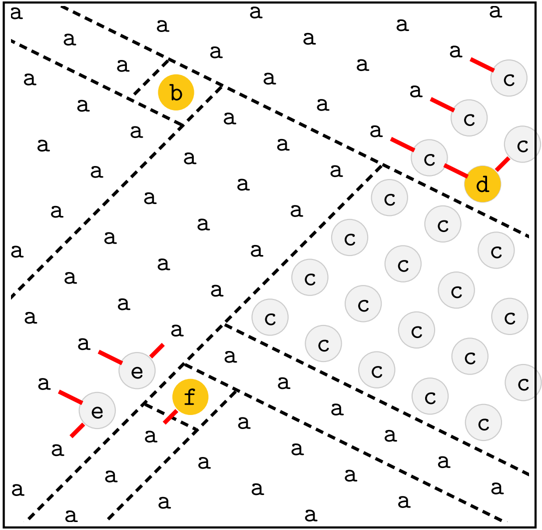
Splitting pattern subsequences

- diamond leads to $\mathcal{O}(\min(m, k))$ subsequences
- $\mathcal{O}(k)$ mismatches of adjacent symbols (with respect to subsequences)



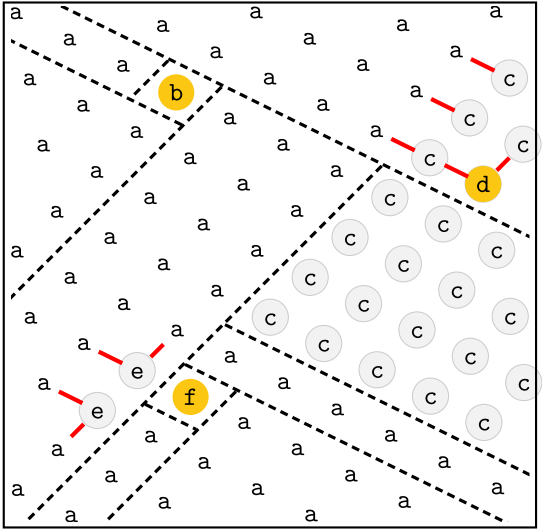
Splitting pattern subsequences

- diamond leads to $\mathcal{O}(\min(m,k))$ subsequences
- $\mathcal{O}(k)$ mismatches of adjacent symbols (with respect to subsequences)



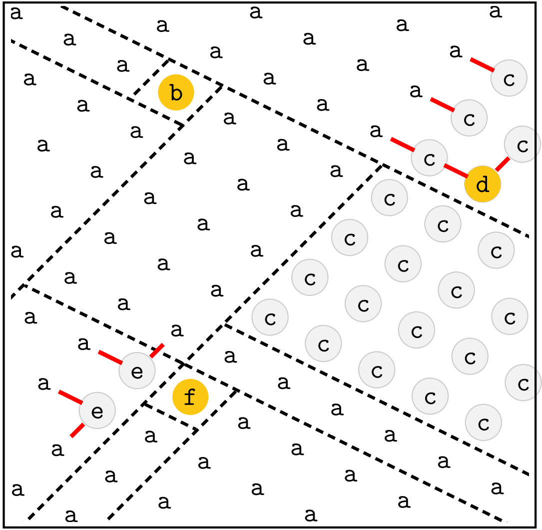
Splitting pattern subsequences

- diamond leads to $\mathcal{O}(\min(m,k))$ subsequences
- $\mathcal{O}(k)$ mismatches of adjacent symbols (with respect to subsequences)



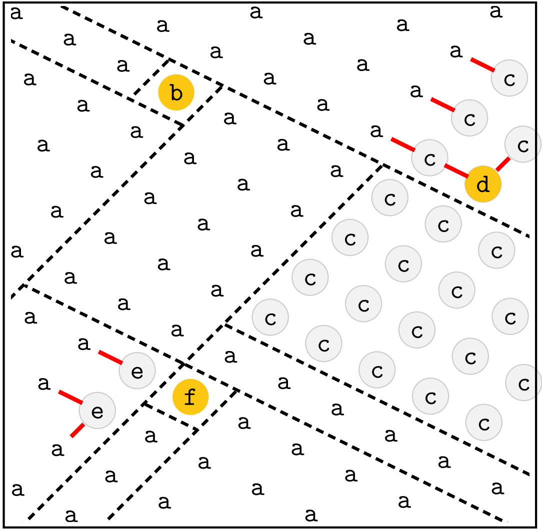
Splitting pattern subsequences

- diamond leads to $\mathcal{O}(\min(m, k))$ subsequences
- $\mathcal{O}(k)$ mismatches of adjacent symbols (with respect to subsequences)



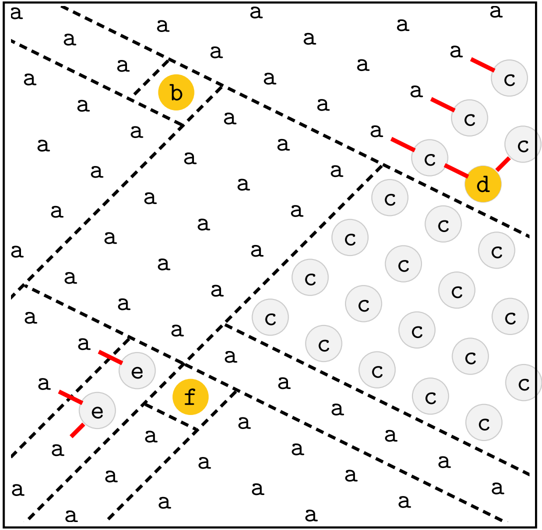
Splitting pattern subsequences

- diamond leads to $\mathcal{O}(\min(m,k))$ subsequences
- $\mathcal{O}(k)$ mismatches of adjacent symbols (with respect to subsequences)



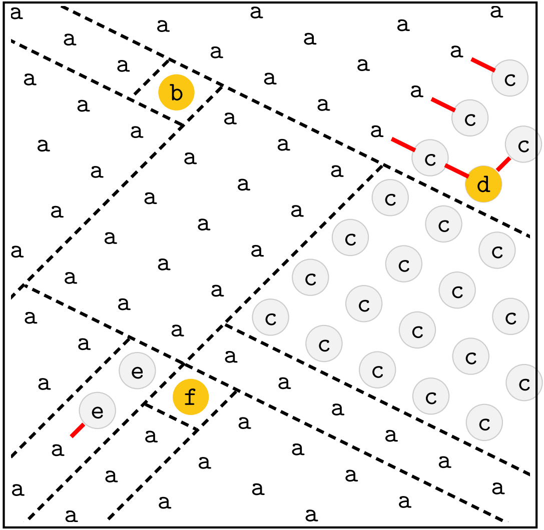
Splitting pattern subsequences

- diamond leads to $\mathcal{O}(\min(m,k))$ subsequences
- $\mathcal{O}(k)$ mismatches of adjacent symbols (with respect to subsequences)



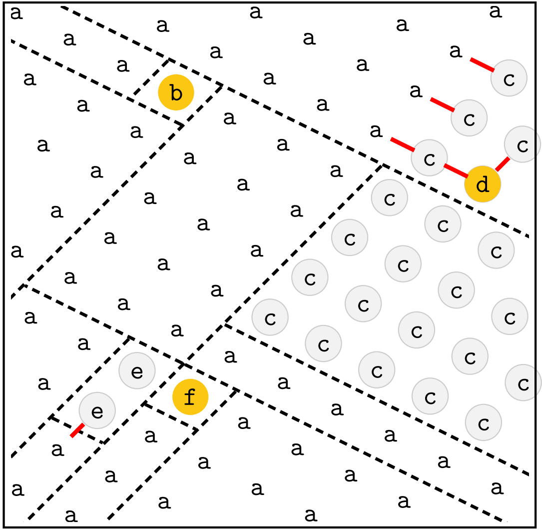
Splitting pattern subsequences

- diamond leads to $\mathcal{O}(\min(m,k))$ subsequences
- $\mathcal{O}(k)$ mismatches of adjacent symbols (with respect to subsequences)



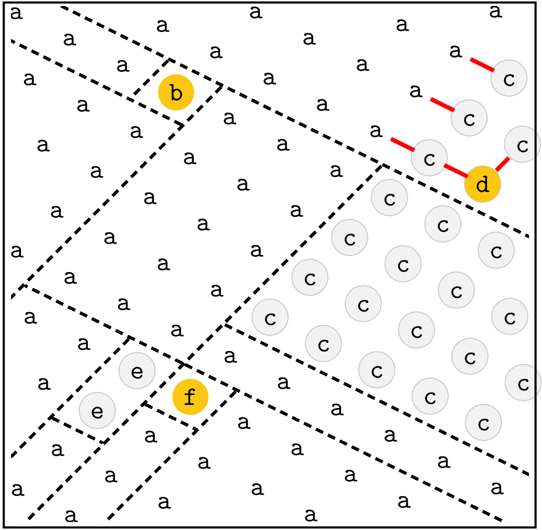
Splitting pattern subsequences

- diamond leads to $\mathcal{O}(\min(m,k))$ subsequences
- $\mathcal{O}(k)$ mismatches of adjacent symbols (with respect to subsequences)



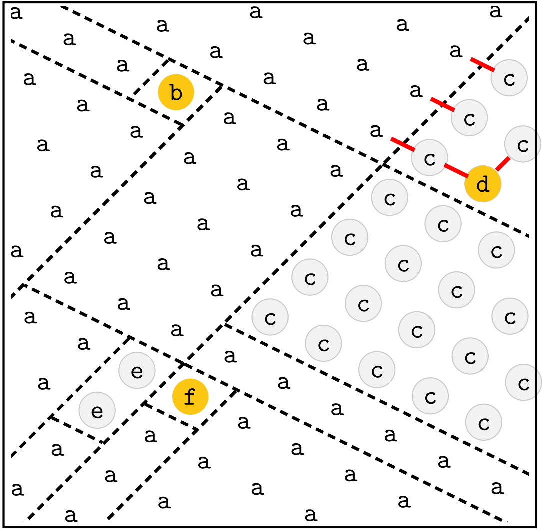
Splitting pattern subsequences

- diamond leads to $\mathcal{O}(\min(m, k))$ subsequences
- $\mathcal{O}(k)$ mismatches of adjacent symbols (with respect to subsequences)



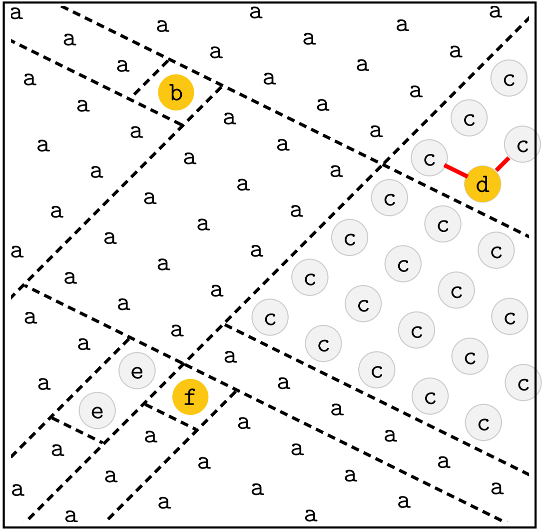
Splitting pattern subsequences

- diamond leads to $\mathcal{O}(\min(m,k))$ subsequences
- $\mathcal{O}(k)$ mismatches of adjacent symbols (with respect to subsequences)



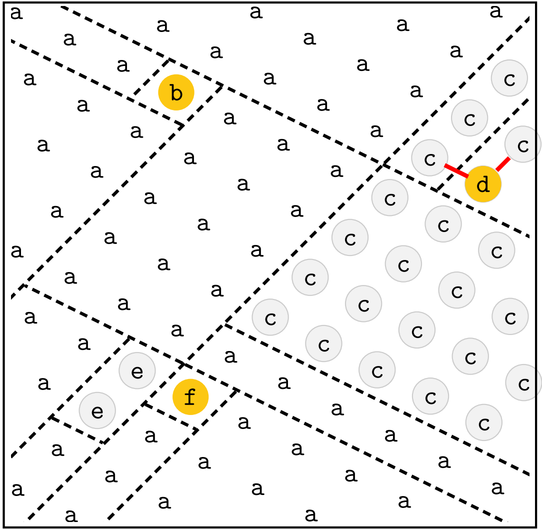
Splitting pattern subsequences

- diamond leads to $\mathcal{O}(\min(m, k))$ subsequences
- $\mathcal{O}(k)$ mismatches of adjacent symbols (with respect to subsequences)



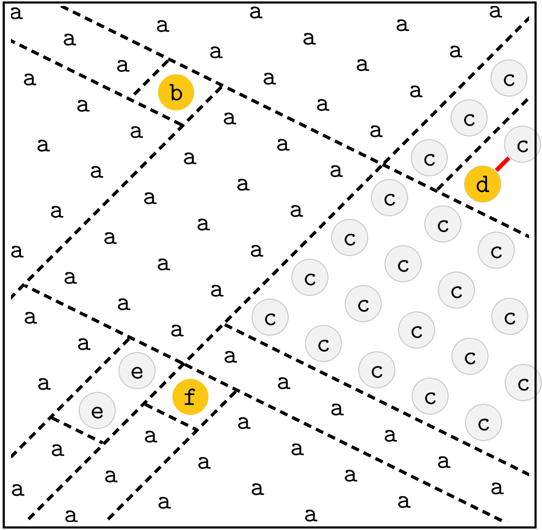
Splitting pattern subsequences

- diamond leads to $\mathcal{O}(\min(m,k))$ subsequences
- $\mathcal{O}(k)$ mismatches of adjacent symbols (with respect to subsequences)



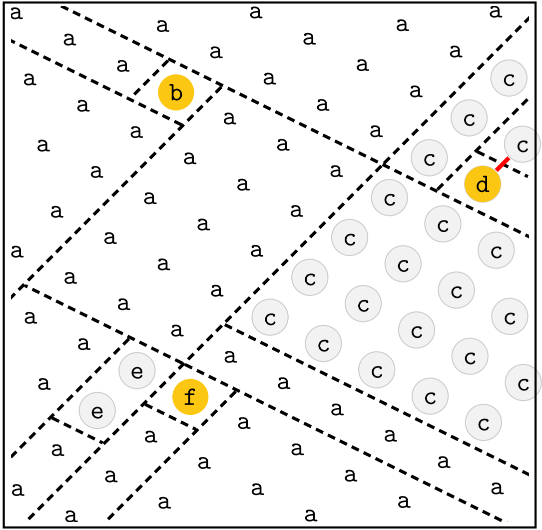
Splitting pattern subsequences

- diamond leads to $\mathcal{O}(\min(m,k))$ subsequences
- $\mathcal{O}(k)$ mismatches of adjacent symbols (with respect to subsequences)



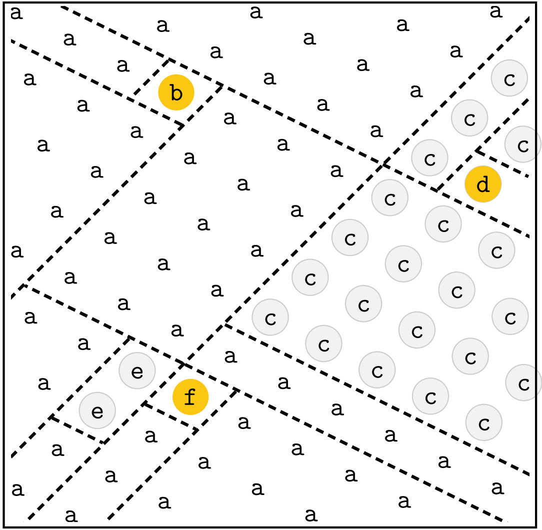
Splitting pattern subsequences

- diamond leads to $\mathcal{O}(\min(m,k))$ subsequences
- $\mathcal{O}(k)$ mismatches of adjacent symbols (with respect to subsequences)



Splitting pattern subsequences

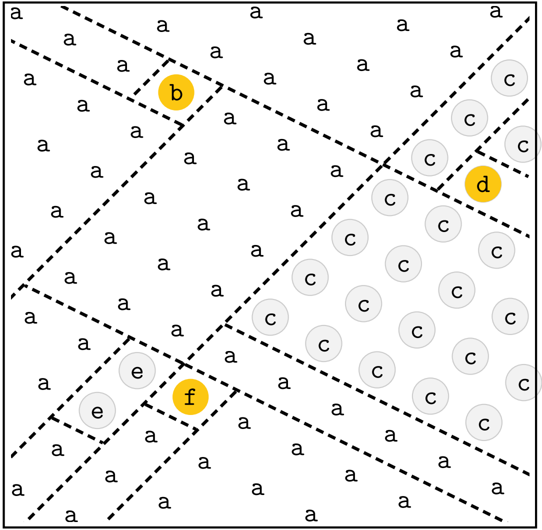
- diamond leads to $\mathcal{O}(\min(m, k))$ subsequences
- $\mathcal{O}(k)$ mismatches of adjacent symbols (with respect to subsequences)



Splitting pattern subsequences

- diamond leads to $\mathcal{O}(\min(m,k))$ subsequences
- $\mathcal{O}(k)$ mismatches of adjacent symbols (with respect to subsequences)

$\implies$ $\mathcal{O}(k)$ single symbol subsequences, each of which is intersection of rectangle and parallelogram

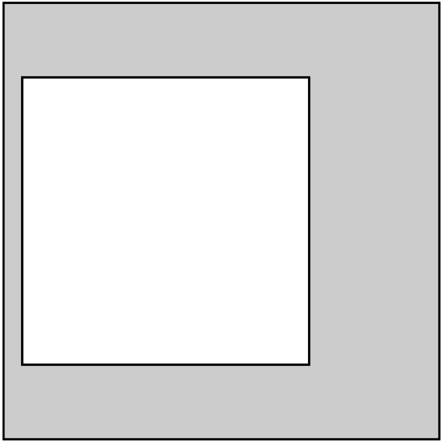


Partitioning the text

- ideally also partition text into $\mathcal{O}(k)$ single symbol subsequences

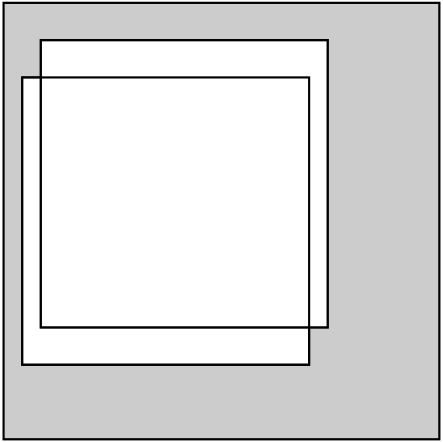
Partitioning the text

- ideally also partition text into $\mathcal{O}(k)$ single symbol subsequences



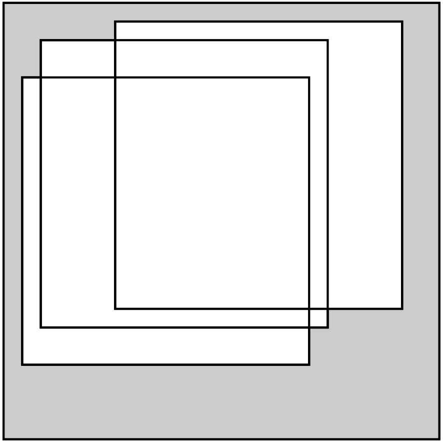
Partitioning the text

- ideally also partition text into $\mathcal{O}(k)$ single symbol subsequences



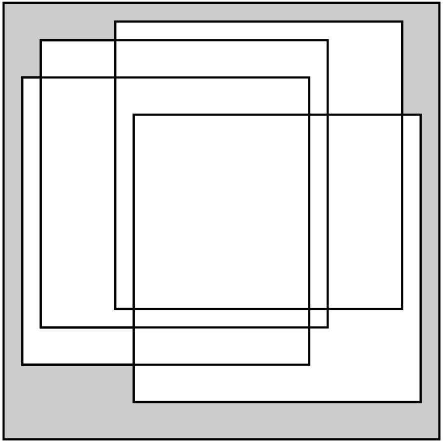
Partitioning the text

- ideally also partition text into $\mathcal{O}(k)$ single symbol subsequences



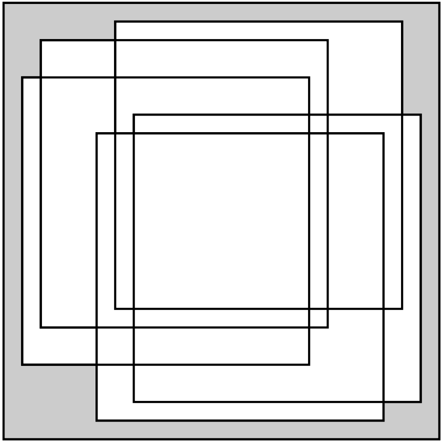
Partitioning the text

- ideally also partition text into $\mathcal{O}(k)$ single symbol subsequences



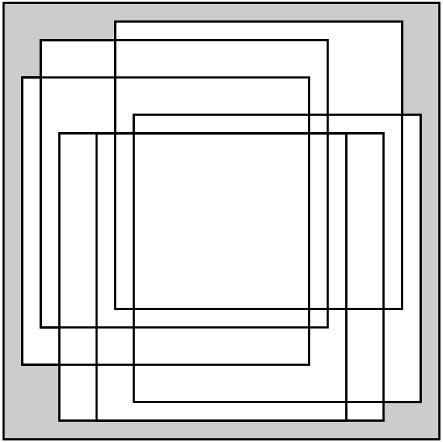
Partitioning the text

- ideally also partition text into $\mathcal{O}(k)$ single symbol subsequences



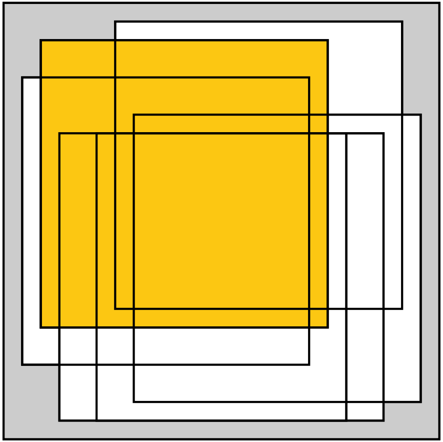
Partitioning the text

- ideally also partition text into $\mathcal{O}(k)$ single symbol subsequences



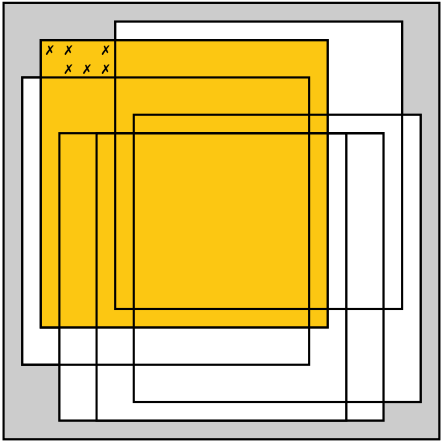
Partitioning the text

- ideally also partition text into $\mathcal{O}(k)$ single symbol subsequences



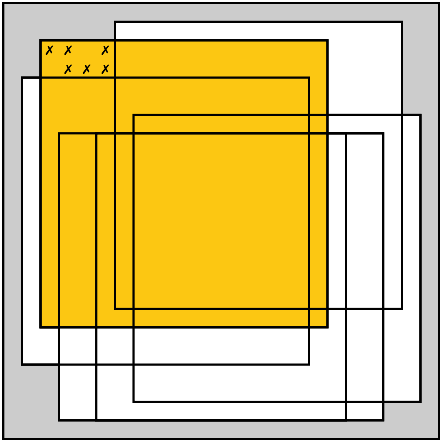
Partitioning the text

- ideally also partition text into $\mathcal{O}(k)$ single symbol subsequences



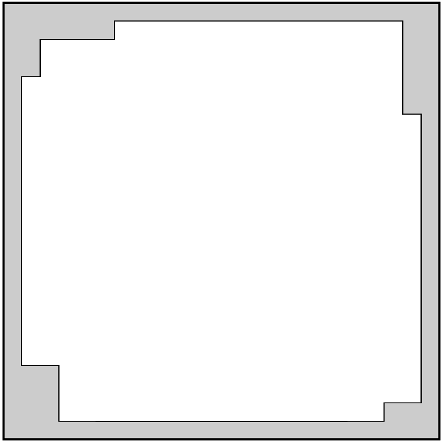
Partitioning the text

- ideally also partition text into $\mathcal{O}(k)$ single symbol subsequences
- but this is not always possible



Partitioning the text

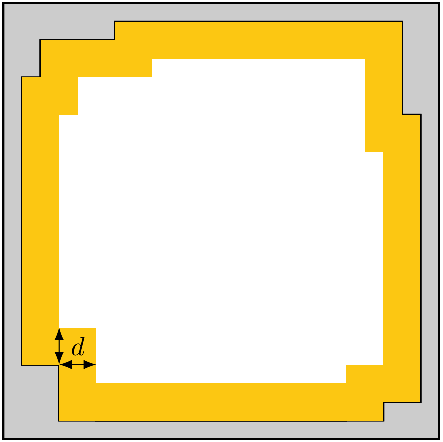
- ideally also partition text into $\mathcal{O}(k)$ single symbol subsequences
- but this is not always possible



Partitioning the text

- ideally also partition text into $\mathcal{O}(k)$ single symbol subsequences
- but this is not always possible

Instead: d-peripheral string

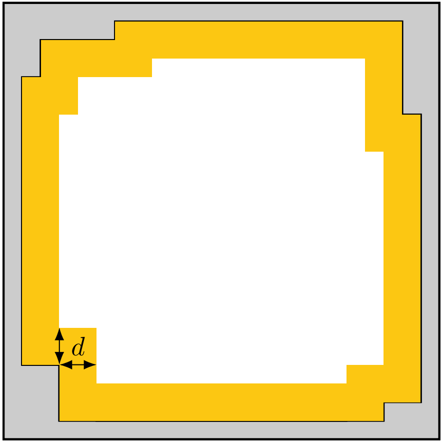


Partitioning the text

- ideally also partition text into $\mathcal{O}(k)$ single symbol subsequences
- but this is not always possible

Instead: d-peripheral string

- count mismatches in $\mathcal{O}(m^2 + md\sqrt{k})$ time



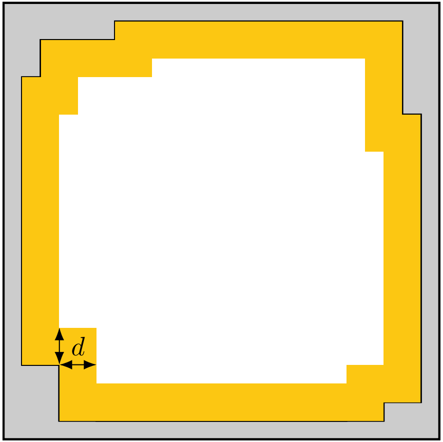
Partitioning the text

- ideally also partition text into $\mathcal{O}(k)$ single symbol subsequences
- but this is not always possible

Instead: d-peripheral string

- count mismatches in $\mathcal{O}(m^2 + md\sqrt{k})$ time

and $\mathcal{O}(mk/d)$ single symbol subsequences



Partitioning the text

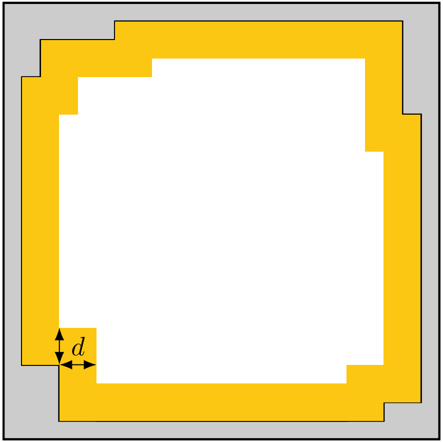
- ideally also partition text into $\mathcal{O}(k)$ single symbol subsequences
- but this is not always possible

Instead: d-peripheral string

- count mismatches in $\mathcal{O}(m^2 + md\sqrt{k})$ time

and $\mathcal{O}(mk/d)$ single symbol subsequences

- count mismatches in $\mathcal{O}(m^2 + mk^2/d)$ time



Partitioning the text

- ideally also partition text into $\mathcal{O}(k)$ single symbol subsequences
- but this is not always possible

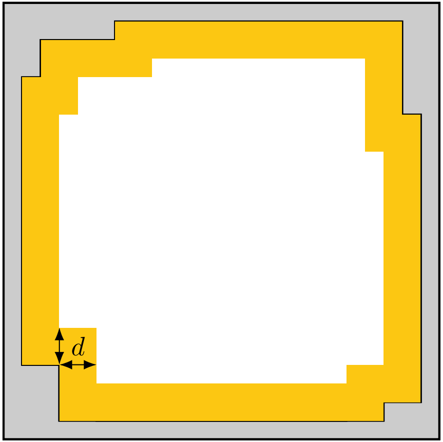
Instead: d-peripheral string

- count mismatches in $\mathcal{O}(m^2 + md\sqrt{k})$ time

and $\mathcal{O}(mk/d)$ single symbol subsequences

- count mismatches in $\mathcal{O}(m^2 + mk^2/d)$ time

$\Rightarrow$ use $d = \Theta(k^{3/4})$ for $\mathcal{O}(m^2 + mk^{5/4})$ time



Summary

Summary

- 2D pattern matching with up to k mismatches in $\mathcal{O}(m^2 + mk^{5/4})$ time

Summary

- 2D pattern matching with up to k mismatches in $\mathcal{O}(m^2 + mk^{5/4})$ time
- many k-mismatch occurrences imply strong 2D periodicity

Summary

- 2D pattern matching with up to k mismatches in $\mathcal{O}(m^2 + mk^{5/4})$ time
- many k-mismatch occurrences imply strong 2D periodicity

Open questions:

Summary

- 2D pattern matching with up to k mismatches in $\mathcal{O}(m^2 + mk^{5/4})$ time
- many k-mismatch occurrences imply strong 2D periodicity

Open questions:

- Can we get $\mathcal{O}(m^2 + mk)$ time?

Summary

- 2D pattern matching with up to k mismatches in $\mathcal{O}(m^2 + mk^{5/4})$ time
- many k-mismatch occurrences imply strong 2D periodicity

Open questions:

- Can we get $\mathcal{O}(m^2 + mk)$ time?
- Other notions of approximate occurrences (e.g., edit distance)?

Summary

- 2D pattern matching with up to k mismatches in $\mathcal{O}(m^2 + mk^{5/4})$ time
- many k-mismatch occurrences imply strong 2D periodicity

Open questions:

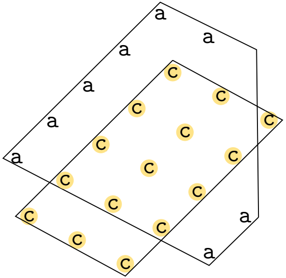
- Can we get $\mathcal{O}(m^2 + mk)$ time?
- Other notions of approximate occurrences (e.g., edit distance)?
- Lower bounds?

Matching subsequences

- compute, for each candidate alignment in $\mathcal{C}$, the aligned matches between each pattern subsequence and each text subsequence

Matching subsequences

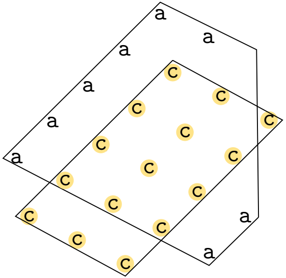
- compute, for each candidate alignment in $\mathcal{C}$, the aligned matches between each pattern subsequence and each text subsequence



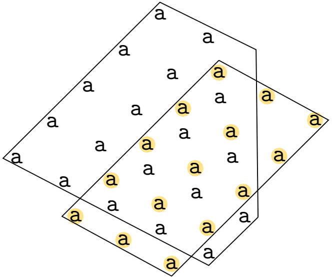
no matches

Matching subsequences

- compute, for each candidate alignment in $\mathcal{C}$, the aligned matches between each pattern subsequence and each text subsequence



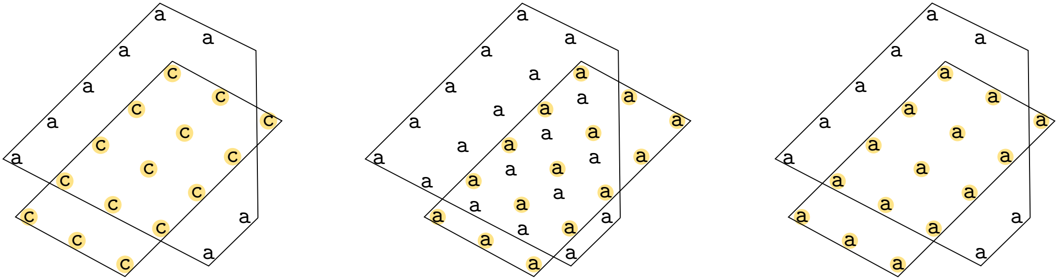
no matches



no matches

Matching subsequences

- compute, for each candidate alignment in $\mathcal{C}$, the aligned matches between each pattern subsequence and each text subsequence



no matches | no matches | matches = intersection