Knuth-Morris-Pratt deterministic string matching

Text: HOCUSPOCUSABRABRACADABRA $T[1..n]$
Pattern: ABRACADABRA $P[1..m]$

ABRACADABRA
ABRACADABRA
ABRACADABRA

Once we've matched $T[i]$ to some $P[j]$, we shouldn't need to compare $T[i]$ to anything else

The next shift is smallest s such that $T[s..i-1]$ are a shorter prefix of the pattern.

~~For now, magic~~

But what is this?

$P[i]$	A	B	R	A	C	A	D	A	B	R	A
$fail[i]$	0	1	1	1	2	1	2	1	2	3	4

Failure function for the string ABRACADABRA

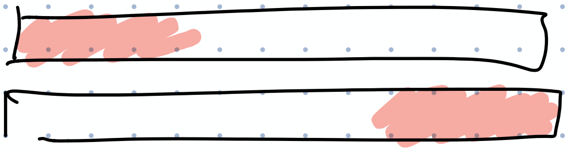
```
KnuthMorrisPratt(T[1..n], P[1..m]):
  j ← 1
  for i ← 1 to n
      while j > 0 and T[i] ≠ P[j]
          j ← fail[j]
      if j = m        ⟨⟨Found it!⟩⟩
          return i − m + 1
      j ← j + 1
  return None
```

Running time: Crudely $O(nm)$

\# increment $i \leq n$
\# increment $j \leq n$
\# decrease $j \leq$ \# increment $j \leq n$

$O(n)$ time

Prefix :
Suffix :



proper = not the whole string

Border = both a nonempty proper prefix and a suffix of P[1..i-1]

ABACABADABACABA

Borders are nested!

Border of a string w is either
longest border of w
or a border of a border of w

> $P[1\,..\,\mathit{fail}[j]-1]$ is the longest proper prefix of $P[1\,..\,j-1]$
> that is also a suffix of $T[1\,..\,i-1]$.

= longest border of P[1..j-1]

> $P[1\,..\,\mathit{fail}[j]-1]$ is the longest proper prefix of $P[1\,..\,j-1]$
> that is also a suffix of $\mathbf{P[1\,..\,j-1]}$.

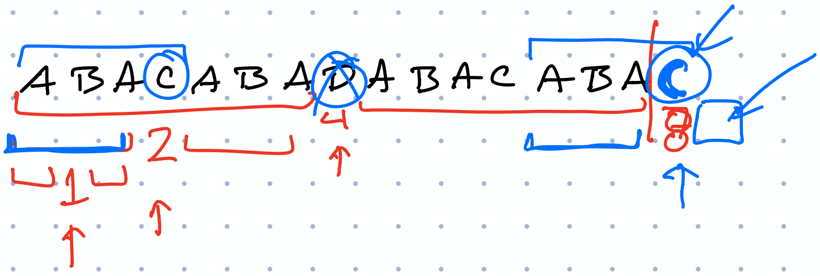
$P[i]$	A	B	R	A	C	A	D	A	B	R	A
$\mathit{fail}[i]$	0	1	1	1	2	1	2	1	2	3	4

Failure function for the string ABRACADABRA

Naive $O(m^3)$
DP $O(m^2)$

```
ComputeFailure(P[1..m]):
  j ← 0
  for i ← 1 to m
      fail[i] ← j        (∗)
      while j > 0 and P[i] ≠ P[j]
          j ← fail[j]
      j ← j + 1
```

→ $O(m)$ time



$$\mathit{fail}^c(i-1) = \begin{cases} \mathit{fail}(\mathit{fail}^{c-1}(i-1)) & \text{if } c>0 \\ i-1 & \text{if } c=0 \end{cases}$$

$$\mathit{fail}[i] = \begin{cases} 0 & \text{if } i = 0, \\ \max\limits_{c \geq 1} \left\{ \mathit{fail}^c[i-1] + 1 \;\middle|\; P[i-1] = P[\mathit{fail}^c[i-1]] \right\} & \text{otherwise.} \end{cases}$$

```
KnuthMorrisPratt(T[1..n], P[1..m]):
  j ← 1
  for i ← 1 to n
      while j > 0 and T[i] ≠ P[j]
          j ← fail[j]
      if j = m        ⟨⟨Found it!⟩⟩
          return i − m + 1
      j ← j + 1
  return None
```

```
ComputeFailure(P[1..m]):
  j ← 0
  for i ← 1 to m
      fail[i] ← j        (∗)
      while j > 0 and P[i] ≠ P[j]
          j ← fail[j]
      j ← j + 1
```

	j								i							
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	$	A	B	A	C	A	B	A	D	A	B	A	C	A	B	A
Fail:	X	0	1	1	2	1	2	3	4	1	2	3	4	5	6	7