RENAMING

MOTIVATION: Parallel Pattern Matching.

Naive algorithm, KMP, the wave—all inherently serial.

Want: a method that uses “local” matches.


Rename pairs of symbols as a single symbol (consistently).
Pattern Renaming:

\[ P = p_1 p_2 \ldots p_{m-1} p_m \]

Let \( P^0 = P \)

Consider: \( <p_i, p_j> \quad <p_3, p_4> \ldots <p_{2i-1}, p_{2i}> \ldots <p_{m-1}, p_m> \)

There are \( \frac{m}{2} \) such pairs.

Number the different pairs by different numbers from \( \Sigma_1 = \{1, \ldots, \frac{m}{2}\} \)

Create new pattern

\[ P' = p'_1 p'_2 \ldots p'_\frac{m}{2} \]

where \( p'_i \in \Sigma_1 \)

\( i = 1, \ldots, \frac{m}{2} \)
Now rename all pairs \( \langle p_{2i}, p'_{2i} \rangle \)

\[ i = 1, \ldots, \frac{m}{2^i} \]

by elements from \( \Sigma_2 = \{1, 2, \ldots, \frac{m}{2^i} \} \)

Proceed for \( \log m \) iterations where

in iteration \( i+1 \):

rename all pairs \( \langle p_{2j-1}, p'_{2j} \rangle \)

\[ j = 1, \ldots, \frac{m}{2^i} \]

by elements from \( \Sigma_{i+1} = \{1, 2, \ldots, \frac{m}{2^{i+1}} \} \)

After iteration \( \log m \): Pattern reduced to a single symbol "1".
Time: Renaming can be done in linear time at every step by radix sorting the pairs.

Total Pattern Preprocessing Time:
\[ \sum_{i=0}^{\log m} \frac{m}{2^i} = O(m). \]

Text Processing:

We would like a similar renaming but do not know where pattern starts.

So unlike pattern, we need to rename at every location.
We have $\log m$ steps.

At step $j$:

For $i = 1$ to $n$

if $<t_{i}^{j-1}, t_{i+2^{j-1}}^{j-1}>$ is one of the pairs that where renamed in pattern step $j$ then $t_{i}^{j}$ is the name of that pair in $P_{j}$.

Otherwise $t_{i}^{j} \leftarrow B$

end

After Step $\log m$:

$\exists$ occurrence of $P$ in location $i$ of $T$ iff

$t_{i}^{\log m} = 1.$
EXAMPLE:
\[ T = \text{ababababc} \text{caabaca} \]
\[ p = \text{babc} \]

Pattern Renaming:
step 1: \langle b, a \rangle \langle b, c \rangle
\[ p^1 = 1 \ 2 \]
step 2: \langle 1, 2 \rangle
\[ p^2 = 1 \]

Text Scanning:
step 1:
\[ \langle ab \rangle \langle ba \rangle \langle ab \rangle \langle ba \rangle \langle ab \rangle \langle ba \rangle \langle ab \rangle \langle bc \rangle \langle cc \rangle \langle ca \rangle \langle ab \rangle \langle ba \rangle \langle ab \rangle \langle bc \rangle \langle ca \rangle \]

\[ T^1 = B 1 B 1 B 1 B 2 B 0 3 1 B 2 B 8 \]
step 2:
\[ \langle ab \rangle \langle 11 \times ab \rangle \langle 11 \times ab \rangle \langle 12 \rangle \langle 08 \rangle \langle 20 \rangle \langle ab \rangle \langle 81 \rangle \langle 08 \rangle \langle 12 \rangle \langle 08 \rangle \]

\[ T^2 = B B B B B B 1 B B B B B B 1 B \]

"Rnk"
Time for text scanning:
Checking if a pair \( <x,y> \) of text is one of the pattern pairs can be done in time \( O(\log m) \).

This is done for every text location:
\[ O(n \log m) \]
For each of the \( \log m \) steps:
\[ O(n \log^2 m) \]

Can be done by \( n \) processors in parallel in time
\[ O(\log^2 m) \]

But can be done better serially...
1. In time $O(n \log m)$
   convert pattern alphabet to $\{1, \ldots, m\}$
   convert text alphabet to $\{1, \ldots, m, B\}$

2. At every one of the $\log m$ steps:
   Radix sort text & pattern pairs
   \[
   \text{(Time: } O(n+m))
   \]
   Then merge
   \[
   \text{(Time: } O(n+m))
   \]

Total Time per Step: $O(n)$

Total Algorithm Time ($\log m$ steps):
$O(m \log m)$