5 Parallel String Matching

10 March 2021

Sebastian Wild

version 2021-03-10 10:47

Outline

5 Parallel String Matching

5.1 Elementary Tricks

5.2 Periodicity

5.3 String Matching by Duels

Parallelizing string matching

- We have seen a plethora of string matching methods
- But all efficient methods seem inherently sequential
  Indeed, they became efficient only after building on knowledge from previous steps!

Sounds like the *opposite* of parallel!

$\rightsquigarrow$ This unit:
- How well can we parallelize string matching?
- What new ideas can help?

Here: string matching = find *all* occurrences of P in T (more natural problem for parallel)
always assume $m \le n$

5.1 Elementary Tricks

Embarrassingly Parallel

- A problem is called *"embarrassingly parallel"*
  if it can immediately be split into *many, small subtasks*
  that can be solved completely *independently* of each other
- Typical example: sum of two large matrices (all entries independent)

⇝ best case for parallel computation (simply assign each processor one subtask)

- Sorting is not embarrassingly parallel
  - no obvious way to define many *small* (=efficiently solvable) subproblems
  - but: some subtasks of our algorithms are, e. g., comparing all elements with pivot

Clicker Question

Is the string-matching problem "embarrassingly parallel"?

- **A** Yes
- **B** No
- **C** Only for $n \gg m$
- **D** Only for $n \approx m$

`sli.do/comp526`

Click on "Polls" tab

Elementary parallel string matching

Subprojects in string matching:

- string matching = check all guesses $i = 0, \ldots, n - m - 1$
- checking one guess is a subtask!

Elementary parallel string matching

Subproblems in string matching:

- string matching = check all guesses $i = 0, \ldots, n - m - 1$
- checking one guess is a subtask!

Approach 1:

- Check all guesses in parallel all n-m guesses

$\rightsquigarrow$ **Time**: $\Theta(m)$ using sequential checks
$\Theta(\log m)$ on CREW-PRAM ($\rightsquigarrow$ see tutorials)
$\Theta(1)$ on CRCW-PRAM ($\rightsquigarrow$ see tutorials)

$\rightsquigarrow$ **Work**: $\Theta((n - m)m)$ $\rightsquigarrow$ not great ...

Elementary parallel string matching

Subproblems in string matching:

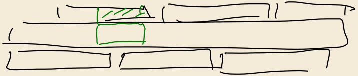
- string matching = check all guesses $i = 0, \ldots, n-m-1$
- checking one guess is a subtask!

Approach 1:

- Check all guesses in parallel

$\rightsquigarrow$ **Time:** $\Theta(m)$ using sequential checks
$\Theta(\log m)$ on CREW-PRAM ($\rightsquigarrow$ see tutorials)
$\Theta(1)$ on CRCW-PRAM ($\rightsquigarrow$ see tutorials)

$\rightsquigarrow$ **Work:** $\Theta((n-m)m)$ $\rightsquigarrow$ not great . . .



Approach 2:

- Divide T into **overlapping** blocks of $2m$ characters:
  $T[0..2m)$, $T[m..3m)$, $T[2m..4m)$, $T[3m..5m)$. . .
- Find matches inside blocks in parallel, using efficient sequential method
  $\rightsquigarrow$ $\Theta(2m+m) = \Theta(m)$ each

$\rightsquigarrow$ **Time:** $\Theta(m)$ **Work:** $\Theta(\frac{n}{m} \cdot m) = \Theta(n)$

Clicker Question

Is the string-matching problem "embarrassingly parallel"?

A Yes

B No

C Only for $n \gg m$

D Only for $n \approx m$

`sli.do/comp526`

Click on "Polls" tab

Clicker Question

Is the string-matching problem “embarrassingly parallel”?

- **A** ~~Yes~~
- **B** ~~No~~
- **C** Only for $n \gg m$ $\checkmark$
- **D** ~~Only for $n \approx m$~~

sli.do/comp526

Click on “Polls” tab

Elementary parallel matching – Discussion

- very simple methods
- could even run distributed with access to part of T
- parallel speedup only for $m \ll n$

Goal:

- methods with better parallel time! $\rightsquigarrow$ higher speedup

$\rightsquigarrow$ must genuinely parallelize the matching process! (and the preprocessing of the pattern)

$\rightsquigarrow$ need new ideas

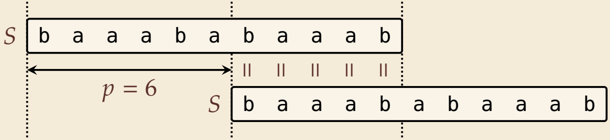
5.2 Periodicity

Periodicity of Strings

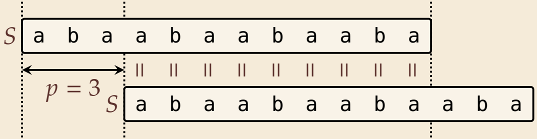
- $S = S[0..n-1]$ has *period p* iff $\forall i \in [0..n-p) : S[i] = S[i+p]$
- $p = 0$ and any $p \geq n$ are trivial periods   but these are not very interesting . . .

Examples:

- $S =$ baaababaaab has period 6:



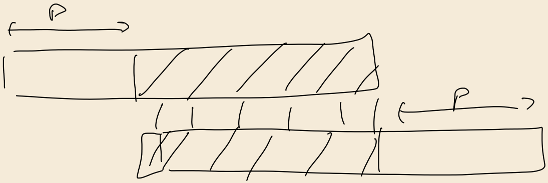
- $S =$ abaabaabaaba has period 3:



Periodicity and KMP

Lemma 5.1 (Periodicity = Longest Overlap)

$p \in [1..n]$ is the *shortest* period in $S = S[0..n-1]$
iff $S[0..n-p)$ is the longest prefix that is also a suffix of $S[p..n)$.



Periodicity and KMP

Lemma 5.1 (Periodicity = Longest Overlap)

$p \in [1..n]$ is the *shortest* period in $S = S[0..n-1]$
iff $S[0..n-p)$ is the longest prefix that is also a suffix of $S[p..n)$.

$S[0..n-1]$ has minimal period $p \iff fail[n] = n - p$