Algorithms in Systems Engineering
IE170

Lecture 21

Dr. Ted Ralphs
References for Today’s Lecture

• Required reading
  – CLRS Chapter 32
The Knuth-Morris-Pratt Algorithm

• By being a little smarter, we can get a string-matching algorithm linear run time.

• The Knuth-Morris-Pratt Algorithm is based on the pre-computation a prefix function for the pattern.

• The prefix function “looks ahead” and tells us something about the validity of upcoming shifts by examining how well the current shift matches the pattern.

• Example:
The Naive KMP Algorithm

- In this simplified algorithm, we keep track of two pointers $s$ and $r$.
- The pointer $s$ is the start of the shift we are investigating.
- At all times, we maintain $r$ such that $P[0..r - s] = T[s..r]$.

The Simple KMP Algorithm

1. Set $s = 0$ and $r = -1$.
2. While $s < n - m$
   - If $r - s = m - 1$, then $s$ is a valid shift. Increase $s$ by at least 1 until $P[0..r - s] = T[s..r]$ and continue.
   - If $P[r - s + 1] = T[r + 1]$, then increase $r$ by 1 and continue.
   - If $P[r - s + 1] \neq T[r + 1]$, then increase $s$ by at least 1 until $P[0..r - s] = T[s..r]$ and continue.

- How do we implement the first and third options in Step 2?
- What is the running time of this algorithm?
Improving the Simple KMP Algorithm

• Suppose that the string $P[0..q]$ matches $T[s..s + q]$, but $P[q + 1] \neq T[s + q + 1]$.

• **Question**: What is the smallest $k$ such that $P[0..q - k] = T[s + k, s + q]$?

• In other words, what is the first shift after $s$ that isn’t necessarily invalid based on our knowledge of $T[s..s + q]$?

• The answer has to do with how the pattern overlaps itself.

• We can easily determine the answer for a particular instance, but how do we do so efficiently?
The Prefix Function

- A **prefix** of a string $S$ is any substring beginning with the first character of $S$, i.e., $S[0..q]$.

- A **suffix** is defined similarly as any substring of $S$ that ends with the last character of $S$.

- Note that the empty string is a both prefix and a suffix for any string.

- Define $\pi[q]$ to be the length of the longest prefix of $P$ that is a suffix of $P[0..q]$.

- The answer to the question from the previous slide is then $\rho(q) = q - \pi[q]$ (why?).

- Note that the value of $\rho[q]$ depends only on $P$, not on $T$.

- We can calculate $\rho[0..m - 1]$ easily in $\Theta(m)$ time.

- What do we do with this?
The Algorithm

- In the KMP algorithm, we keep track of two pointers $s$ and $r$.
- The pointer $s$ is the start of the shift we are investigating.
- At all times, we maintain $r$ such that $P[0..r-s] = T[s..r]$.
- For notational convenience, set $\rho(-1) = 1$.

The KMP Algorithm

1. Set $s = 0$ and $r = -1$.
2. While $s < n - m$
   - If $r - s = m - 1$, then $s$ is a valid shift. Increase $s$ by $\rho(r - s)$ and continue.
   - If $P[r - s + 1] = T[r + 1]$, then increase $r$ by 1 and continue.
   - If $P[r - s + 1] \neq T[r + 1]$, then increase $s$ by $\rho(r - s)$ and continue.

- Now what is the running time of the algorithm?