String Matching

Hsuan-Tien Lin

Dept. of CSIE, NTU

June 19, 2012

String Matching

find the position that *pat* (first) shows up in *string*

for $i \leftarrow 0$ to $len(string) - 1$ **do**
  if *pat* matches $strings[i, i + len(pat) - 1]$
    matching found
  end if
end for
matching not found

- the IF takes $O(m)$ for $m = len(pat)$
  —can use heuristic on comparing *begin* and *end* first, but still $O(m)$ in the worst case
- so total $O(n * m)$ for $n = len(string)$

Slow (Naive) Pattern Matching

$i, j \leftarrow 0$
while $i < len(string)$ and $j < len(pat)$ **do**
  if $pat[j] == string[i]$
    $i \leftarrow i + 1$, $j \leftarrow j + 1$ (continue matching)
  else
    $i \leftarrow i - j + 1$
    $j \leftarrow 0$
    (fail and totally go back)
  end if
end while
check matching status

see demo

"Jump" Pattern Matching

$i, j \leftarrow 0$
while $i < len(string)$ and $j < len(pat)$ **do**
  if $pat[j] == string[i]$
    $i \leftarrow i + 1$, $j \leftarrow j + 1$ (continue matching)
  else
    $i \leftarrow i - \min(jump, j) + 1$
    $j \leftarrow 0$
    (fail and go to next possible starting point)
  end if
end while
check matching status

see demo

Fast (Knuth-Morris-Pratt) Pattern Matching

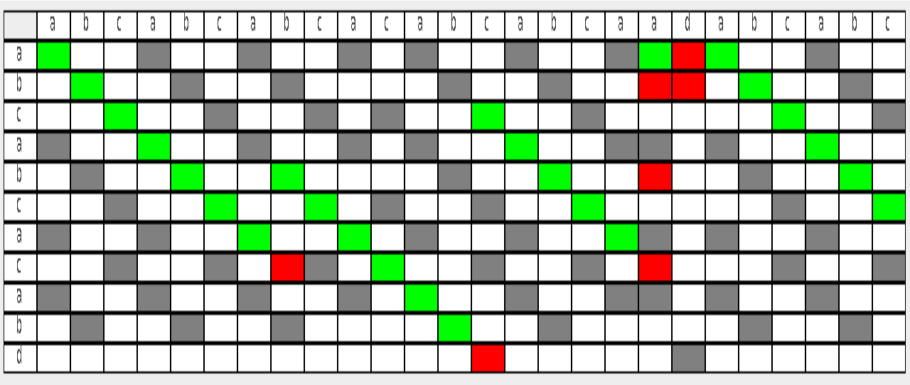
$i, j \leftarrow 0$
while $i < len(string)$ and $j < len(pat)$ **do**
  if $pat[j] == string[i]$
    $i \leftarrow i + 1$, $j \leftarrow j + 1$ (continue matching)
  else
    $i \leftarrow i$
    decrease j such that $pat[0, j-1]$ matches $string[i-j, i-1]$
    (fail but continue partially)
  end if
end while
check matching status

see demo

Donald Knuth 高德納

- Ph.D., Caltech Math
- Professor Emeritus, Stanford
- 1974 ACM A. M. Turing Award
  (who is Turing and what is Turing Award?)
- 1995 IEEE John von Neumann Medal
  (who is von Neumann?)

For his major contributions to the analysis of algorithms and the design of programming languages, and in particular for his contributions to "The Art of Computer Programming" through his well-known books in a continuous series by this title



- number of increase i = $O(len(string))$ = number of increase j
- number of decrease j = $O($ number of increase $j)$ because $j \geq 0$
- total: $O(len(string))$ IF the decrease step is $O(1)$