Pattern Matching (Lecture 19A)

I. Problem: Given a text t and a pattern p, find the location of p in t.

II. Example: “grep” function in UNIX.

III. Trivial Algorithm: match p against t starting from the first character of t from left to right. If fails, shift p one character to the right, starting from the second character of t, and continue matching. The time complexity of this algorithm is $O(|p||t|)$.

IV. We want to solve the pattern matching problem in $O(|p| + |t|)$ time. This was proved to be possible in the 70’s by Cook, Knuth-Morris-Pratt, Boyer-Moore.

V. Idea: when failing a match, can we shift p more steps to the right, instead of just one step? The answer: Yes.

Skip Table

- Assuming

  $$p[0..i-1] = t[k..k+i-1], \quad \text{but} \quad p[i] \neq t[k+i],$$

  we would like to find the smallest d such that

  $$p[0..i-d] = t[k+d..k+i].$$

  Note that $d \leq i+1$. If no smaller d works, we always set $d = i+1$, and continue to do the matching from there.

- Notice a crucial point: except for $t[k+i]$, all above depends only on p! The desired value d depends only on p and on the mismatching character $t[k+i]$—call it c.

Skip Table continues

- Define $KMPskip(p, i, c)$ to be the smallest integer d, $0 \leq d \leq i$ such that

  $$p[i - d] = c$$

  and

  $$p[j] = p[j + d]$$

  for each $0 \leq j \leq i-d-1$, and $KMPskip(p, i, c) = i + 1$ if no such d exists.

- This $KMPskip$ function can be computed in linear time, $O(|p|)$ and it is only computed once in the beginning.

Two Examples: $KMPskip$ tables for ABCD and for XYXYZ.

c	p[0]=A	p[1]=B	p[2]=C	p[3]=D
A	0	1	2	3
B	1	0	3	4
C	1	2	0	4
D	1	2	3	0
other	1	2	3	4

c	p[0] =X	p[1] =Y	p[2] =X	p[3] =Y	p[4] =Z
X	0	1	0	3	2
Y	1	0	3	0	5
Z	1	2	3	4	0
other	1	2	3	4	5

The Knuth-Morris-Pratt Algorithm

- Input: text t and pattern p.

- Compute the $KMPskip$ table for p.

- Start to match p against t from left to right.

- If a mismatch happens, *i.e.* $p[0..i-1] = t[k..k+i-1]$ but $p[i] \neq t[k+i]$, then use the $KMPskip$ table to move the pattern to the right d places.

- By the way $KMPskip$ table is defined, we know that $t[k+d..k+i]$ match correctly to $p[0..i-d]$ after the move. So we continue to do the matching starting from $t[k+i+1]$ and $p[i+1-d]$.

- Repeat the above until a match is found or we reach the end of text.

Time Complexity of the Knuth-Morris-Pratt Algorithm

The time complexity of the algorithm depends on

- Time to compute the $KMPskip$ table — $O(|p|)$ time.

- Time to do the matching — $O(|t|)$ time.

So the KMP algorithm is a linear time, $O(|t| + |p|)$, algorithm.