Advanced Algorithm

String Matching

Topics To be Covered

- ✓ String Matching Terminology
- ✓ String Matching Applications
- ✓ Naïve String Matching (Brute-Force Algorithm)
- ✓ Horspool's Algorithm
- ✓ String Matching Using Finite Automata
- ✓ Rabin-Karp Algorithm
- ✓ **Knuth-Morris-Pratt Algorithm**

Knuth-Morris-Pratt Algorithm

This algorithm was conceived by Donald Knuth and Vaughan Pratt and independently by James H.Morris in 1977.

Knuth-Morris-Pratt Algorithm

ABC_ABCDAB_ABCDABCDABDE
ABCDABD
1234567

0	1	2	3	4	5	6	7
	A	B	C	D	A	B	D
-1	0	0	0	0	1	2	0

Knuth-Morris-Pratt Algorithm

ABC_ABCDAB_ABCDABCDABDE
ABCDABD
1234567

0	1	2	3	4	5	6	7
	A	B	C	D	A	B	D
-1	0	0	0	0	1	2	0

Knuth-Morris-Pratt Algorithm

ABC_ABCDAB_ABCDABCDABDE
ABCDABD
1234567

0	1	2	3	4	5	6	7
	A	B	C	D	A	B	D
-1	0	0	0	0	1	2	0

Knuth-Morris-Pratt Algorithm

ABC_ABCDAB_ABCDABCDABDE
ABCDABD
1234567

0	1	2	3	4	5	6	7
	A	B	C	D	A	B	D
-1	0	0	0	0	1	2	0

Knuth-Morris-Pratt Algorithm

ABC_ABCDAB_ABCDABCDABDE
ABCDABD
1234567

0	1	2	3	4	5	6	7
	A	B	C	D	A	B	D
-1	0	0	0	0	1	2	0

Knuth-Morris-Pratt Algorithm

ABC_ABCDAB_ABCDABCDABDE
ABCDABD
1234567

0	1	2	3	4	5	6	7
	A	B	C	D	A	B	D
-1	0	0	0	0	1	2	0

Knuth-Morris-Pratt Algorithm

ABC_ABCDAB_ABCDABCDABDE
ABCDABD
1234567

0	1	2	3	4	5	6	7
	A	B	C	D	A	B	D
-1	0	0	0	0	1	2	0

Knuth-Morris-Pratt Algorithm (Construction of Prefix Table(Pi Table))

Pattern: ABCDABD

```
pi[0] = -1;
int k = -1;
for(int i = 1; i <= m; i++) {
  while(k >= 0 && P[k+1] != P[i])
    k = pi[k];
  pi[i] = ++k;
}
```

0	1	2	3	4	5	6	7
	A	B	C	D	A	B	D
-1	0	0	0	0	1	2	0

Knuth-Morris-Pratt Algorithm

ABC_ABCDAB_ABCDABCDABDE
ABCDABD
1234567

```
int k = 0;
for(int i = 1; i <= n; i++) {
  while(k >= 0 && P[k+1] != T[i])
    k = pi[k];
  k++;
  if(k == m) {
    // P matches T[i-m+1..i]
    k = pi[k];
  }
}
```

0	1	2	3	4	5	6	7
	A	B	C	D	A	B	D
-1	0	0	0	0	1	2	0

Knuth-Morris-Pratt Algorithm

ABC_ABCDAB_ABCDABCDABDE
ABCDABD
1234567

```
int k = 0;
for(int i = 1; i <= n; i++) {
  while(k >= 0 && P[k+1] != T[i])
    k = pi[k];
  k++;
  if(k == m) {
    // P matches T[i-m+1..i]
    k = pi[k];
  }
}
```

0	1	2	3	4	5	6	7
	A	B	C	D	A	B	D
-1	0	0	0	0	1	2	0

Knuth-Morris-Pratt Algorithm

ABC_ABCDAB_ABCDABCDABDE
ABCDABD
1234567

```
int k = 0;
for(int i = 1; i <= n; i++) {
  while(k >= 0 && P[k+1] != T[i])
    k = pi[k];
  k++;
  if(k == m) {
    // P matches T[i-m+1..i]
    k = pi[k];
  }
}
```

0	1	2	3	4	5	6	7
	A	B	C	D	A	B	D
-1	0	0	0	0	1	2	0

Knuth-Morris-Pratt Algorithm

ABC_ABCDAB_ABCDABCDABDE
ABCDABD
1234567

```
int k = 0;
for(int i = 1; i <= n; i++) {
  while(k >= 0 && P[k+1] != T[i])
    k = pi[k];
  k++;
  if(k == m) {
    // P matches T[i-m+1..i]
    k = pi[k];
  }
}
```

0	1	2	3	4	5	6	7
	A	B	C	D	A	B	D
-1	0	0	0	0	1	2	0

Knuth-Morris-Pratt Algorithm

ABC_ABCDAB_ABCDABCDABDE
ABCDABD
123456 7

```
int k = 0;
for(int i = 1; i <= n; i++) {
  while(k >= 0 && P[k+1] != T[i])
    k = pi[k];
  k++;
  if(k == m) {
    // P matches T[i-m+1..i]
    k = pi[k];
  }
}
```

0	1	2	3	4	5	6	7
	A	B	C	D	A	B	D
-1	0	0	0	0	1	2	0

Knuth-Morris-Pratt Algorithm

ABC_ABCDAB_ABCDABCDABDE
ABCDABD
1234567

```
int k = 0;
for(int i = 1; i <= n; i++) {
  while(k >= 0 && P[k+1] != T[i])
    k = pi[k];
  k++;
  if(k == m) {
    // P matches T[i-m+1..i]
    k = pi[k];
  }
}
```

0	1	2	3	4	5	6	7
	A	B	C	D	A	B	D
-1	0	0	0	0	1	2	0

Knuth-Morris-Pratt Algorithm

ABC_ABCDAB_ABCDABCDABDE
ABCDABD
1234567

```
int k = 0;
for(int i = 1; i <= n; i++) {
  while(k >= 0 && P[k+1] != T[i])
    k = pi[k];
  k++;
  if(k == m) {
    // P matches T[i-m+1..i]
    k = pi[k];
  }
}
```

0	1	2	3	4	5	6	7
	A	B	C	D	A	B	D
-1	0	0	0	0	1	2	0

Knuth-Morris-Pratt Algorithm (Complexity)

A linear time (!) algorithm that solves the string matching problem by preprocessing P in $\Theta(m)$ time

- Main idea is to skip some comparisons by using the previous comparison result

Overall Complexity of KMP algorithm is *O*(n+m)

Try Yourself (Example-1)

b	a	c	b	a	b	a	b	a	b	a	c	a	a	b

a	b	a	b	a	c	a

Try Yourself (Example-1)

b	a	c	b	a	b	a	b	a	b	a	c	a	a	b

a	b	a	b	a	c	a

Try Yourself (Example-2)

Complete the Pi Table given below

0	1	2	3	4	5	6	7	8	9	10
	a	b	a	b	a	b	a	b	c	a
-1										