Knuth Morris Pratt Algorithm

Dr. Mattox Beckman

University of Illinois at Urbana-Champaign
Department of Computer Science

Näive Matching

```
0 void naiveMatching() {
1   for (int i = 0; i < n; i++) {
2     bool found = true;
3     for (int j = 0; j < m && found; j++)
4        if (i + j >= n || P[j] != T[i + j])
5           found = false;  // mismatch!
6     if (found) // if P[0..m-1] == T[i..i+m-1]
7        printf("P is found at index %d in T\n", i);
8 } }
```

Example 1

```
i 01234567890123456789012345678901234567890123456789
T This is a test
P is a
  ^
```

Example 1

```
i 01234567890123456789012345678901234567890123456789
T This is a test
P  is a
   ^
```

Example 1

```
i 01234567890123456789012345678901234567890123456789
T This is a test
P   is a
    ^
```

Example 1

```
i 01234567890123456789012345678901234567890123456789
T This is a test
P   is a
     ^
```

Example 1

```
i 01234567890123456789012345678901234567890123456789
T This is a test
P   is a
      ^
```

Example 1

```
i 01234567890123456789012345678901234567890123456789
T This is a test
P   is a
       *
```

Example 1

```
i 01234567890123456789012345678901234567890123456789
T This is a test
P    is a
     ^
```

Example 1

```
i 01234567890123456789012345678901234567890123456789
T This is a test
P     is a
      ^
```

Example 1

```
i 01234567890123456789012345678901234567890123456789
T This is a test
P      is a
       ^
```

Example 1

```
i 01234567890123456789012345678901234567890
T aaaaaaaaaaxaaaaaaaaaaaaaaaaaaaaaaaaaaaaab
P aaaaaaaaaab
```

The KMP Algorithm

```
i 012345678901234567890123456789012345678901
T aaaaaaaaaaxaaaaaaaaaaaaaaaaaaaaaaaaaaaaaab
P aaaaaaaaaab
           *
```

Example 1

```
i 01234567890123456789012345678901234567890
T aaaaaaaaaaxaaaaaaaaaaaaaaaaaaaaaaaaaaaaab
P            aaaaaaaaaab
             ^
```

The idea

```
P   i s _ a
b  -1 0 0 0

P   a a a a b b b a a a a
b  -1 0 1 2 3 0 0 0 1 2 3

P   a a b b a a b b c c c
b  -1 0 1 0 0 1 2 3 4 0 0
```

The Code

```
0 void kmpPreprocess() {
1   int i = 0, j = -1; b[0] = -1;
2   while (i < m) {
3     while (j >= 0 && P[i] != P[j]) j = b[j];
4     i++; j++;
5     b[i] = j;
6 } }
```

Code, part 2

```
0 void kmpSearch() {
1   int i = 0, j = 0;
2   while (i < n) {
3     while (j >= 0 && T[i] != P[j])
4         j = b[j]; // different, reset j using b
5     i++; j++;
6     if (j == m) { // a match found when j == m
7     printf("P is found at index %d in T\n", i - j);
8     j = b[j];
9 } } }
```