Lecture 4

	worst case	average case
Naive	$O(mn)$	$O(n)$
KMP	$O(m+n)$	$O(m+n)$

Can we do better?

- No/Yes, in the worst case. $n + \frac{cn}{m} - O(1)$ where $2 \le c \le \frac{8}{3}$
- Yes, in the average case.

Boyer-Moore Algorithm

Comparisons from right to left.

How to shift the pattern?

Two heuristics: bad-character heuristic, good-suffix heuristic

Example

```
       T   abcaababcabcabaa..
       P         aabcaaab

bad-character      aabcaaab
good-suffix           aabcaaab
```

Best case: text b^n, pattern a^m. $O(n/m)$ comparisons

Bad-Character Heuristic

1. If the bad character doesn't occur in the pattern, then the pattern can be moved completely past the bad character in the text.
2. If the rightmost occurrence of the bad character in the pattern is to the left of the current position, then shift so that the bad character is aligned with the rightmost occurrence.
3. If the rightmost occurrence is to the right of the current position, this heuristic makes no proposal. In this case the good-suffix heuristic always proposes a shift.

We need to compute Last-occurrence function δ.

Use characters as indices: it needs conversion from char's to int's or use of a big array (in C).

```
procedure Last-Occurrence($P, m, \Sigma$)
    for each char $a \in \Sigma$ do $\delta[a] = 0$ od
    for $j = 1$ to $m$ do $\delta[P[j]] = j$ od
    return $\delta$
end
```

Example: $P = abcaab$

- $\delta[a] = 0, 1, 4, 5$
- $\delta[b] = 0, 2, 6$
- $\delta[c] = 0, 3$
- $\delta[d] = 0$

Running time $O(|\Sigma| + m)$.

Good-Suffix Heuristic

It shifts so that the good suffix of the text is aligned with its next occurrence (from right) in the pattern.

For each position i, compute the position of the next occurrence of $P[i..m]$.

$O(m)$ time using Prefix function f twice. See CLR (1st ed.) for the code.

Worst case running time of Boyer-Moore is $O(mn + |\Sigma|)$.
Example: text b^n, pattern b^m.

Average running time of Boyer-Moore?

What is the algorithm of choice in practice?

Horspool Algorithm

Horspool version of the Boyer-Moore algorithm

1. Remove the good-suffix heuristic.
2. We can use any character from the bad character to the last text character in current alignment. Use the rightmost text character.

Eexample:

```
T = abcacbcadcdacbbada
P = acbcda
       acbcda
        acbcda
          acbcda
               acbcda
                acbcda
                     acbcda
```

$d[a] = m - \delta[a]$

```
d: a b c d
   5 3 2 1
```

```
procedure Horspool($T, P$)
    for each char $a \in \Sigma$ do $d[a] = m$ od
    for $i = 1$ to $m - 1$ do $d[P[i]] = m - i$ od
    $i = m$
    while $i \leq n$ do
        $k = i$
        $j = m$
        while $j > 0$ and $T[k] == P[j]$ do
            $k = k - 1$
            $j = j - 1$
        od
        if $j == 0$ then print “occurrence at $k + 1$” fi
        $i = i + d[T[i]]$
    od
end
```

Lemma 1 *The expected value S of each shift (i.e., $d[T[i]]$) is $q(1-(1-1/q)^m)$, where $q = |\Sigma|$.*

- Average running time: approximately $O(n/|\Sigma|)$
- Asymptotically best expected time is $\Theta(n(\log m)/m)$. Yao proved the lower bound, and there are such algorithms.

Compare $n/|\Sigma|$ vs $n(\log m)/m$.

- ◦ English, programming languagues – Horspool
- ◦ Binary or DNA sequences – $n(\log m)/m$