Strings

In Java, (and other languages - Python, Scala!, C#) strings are immutable.

Algorithms such as:

```
reverse (String s)
X       s → Prefix + c
        return  c + reverse (Prefix)
```

} $RT = O(n^2)$

↳ O(n) algorithm: [StringBuilder class
[toCharArray() — convert to character array.

In general:

do not write: `if (s.equals("Find")) ....`

(see LP4 Driver).

- ok only during testing phase
- should not be used in production code.

Avoid: `HashMap<String, X>`.

String Matching

Input: Text $T[1..n]$, Pattern $P[1..m]$

Output: All occurrences of P in T. — valid shifts

shifts $s_1, s_2, \ldots s_k$: $T[s_i+1 \ldots s_i+m] = P[1..m]$.

Ex: $T =$ a b a b a b

$P =$ a b a : shifts = 0, 2

Naive Algorithm

```
for s ← 0 to n-m do
    if P.equals(T.substring(s, s+m)) then
       P[1..m] = T[s+1...s+m]
       s is a valid shift
```

} $RT = O(mn)$

Better algorithms for string matching

1. Rabin-Karp algorithm: use ideas from hashing to reduce RT.

$h(P[1..m]) = x$

$h(T[s+1 \dots s+m]) = y$

if $x \neq y$ then it is not a valid shift

else Use naive algorithm on $P[1..m], T[s+1 \dots s+m]$

RT = $O(mn)$ worst case, $O(m+n)$ most of the times

From $h(T[s+1 \dots s+m])$, calculate $h(T[s+2 \dots s+m+1])$ in $O(1)$ time.

---

h: Consider $T[s+1 \dots s+m]$ as a polynomial in some base b — mod p (for some random, large prime p)

$s \leftarrow 0$ $a \leftarrow h(P)$, $t \leftarrow h(T[1..m])$

mult $\leftarrow b^{m-1} \pmod p$

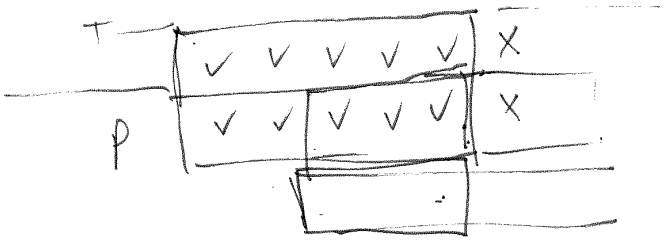
while $s < n-m$ do

if $a = t$ then Naive matcher $(P[1..m], T[s+1 \dots s+m])$

$t \leftarrow (t - T[s] \times mult) * b + T[s+m+1] \pmod p$

2. Finite Automata matcher - next class. RT = $O(m+n)$. space: $O(m \cdot \Sigma)$ Σ - size of alphabet.

3. Knuth-Morris-Pratt (KMP) Algorithm



Prefix function Π: $\Pi[q]$ = Length of Longest prefix of $P[1..q]$ that is also a suffix of $P[1..q]$

```
KMP(T[1..n], P[1..m], Π[1..m]) // Given prefix function Π
    q ← 0          // Length of current match.
    for i ← 1 to n do   // Looking at T[i].
        while q>0 and P[q+1] ≠ T[i] do
            q ← Π[q]
        if P[q+1] = T[i] then q++
        if q = m then s = i-m is a valid shift
```