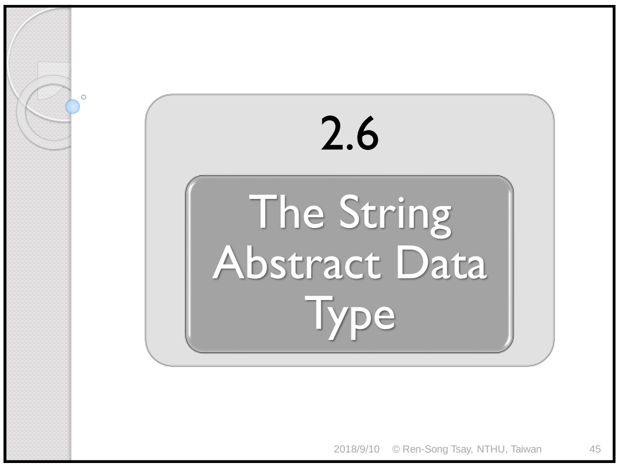
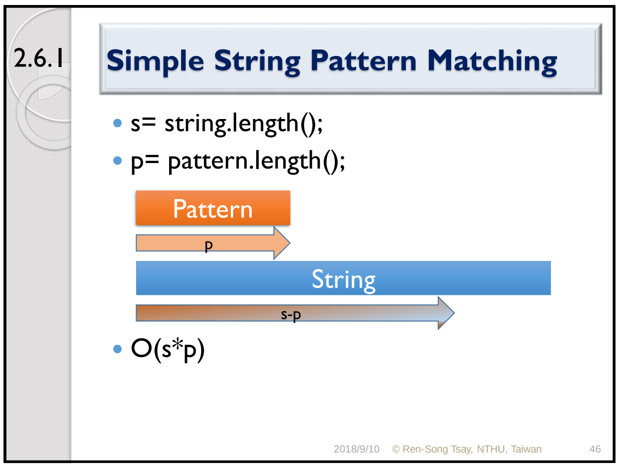
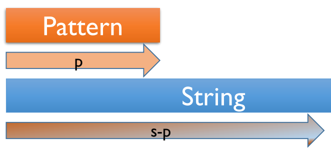


2.6 The String Abstract Data Type

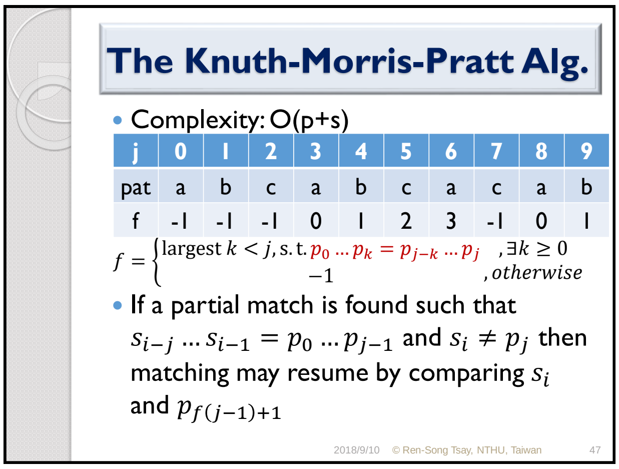


2.6.1 Simple String Pattern Matching

- s= string.length();
- p= pattern.length();



- O(s*p)



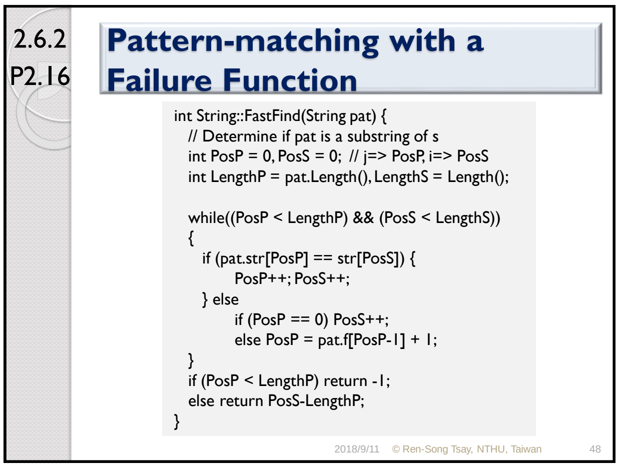
The Knuth-Morris-Pratt Alg.

- Complexity: O(p+s)

j	0	1	2	3	4	5	6	7	8	9
pat	a	b	c	a	b	c	a	c	a	b
f	-1	-1	-1	0	1	2	3	-1	0	1

$$f = \begin{cases} \text{largest } k < j, \text{s.t.} \, p_0 \ldots p_k = p_{j-k} \ldots p_j & , \exists k \geq 0 \\ -1 & , otherwise \end{cases}$$

- If a partial match is found such that $s_{i-j} \ldots s_{i-1} = p_0 \ldots p_{j-1}$ and $s_i \neq p_j$ then matching may resume by comparing s_i and $p_{f(j-1)+1}$



2.6.2
P2.16

Pattern-matching with a Failure Function

```
int String::FastFind(String pat) {
   // Determine if pat is a substring of s
   int PosP = 0, PosS = 0;  // j=> PosP, i=> PosS
   int LengthP = pat.Length(), LengthS = Length();

   while((PosP < LengthP) && (PosS < LengthS))
   {
      if (pat.str[PosP] == str[PosS]) {
             PosP++; PosS++;
      } else
             if (PosP == 0) PosS++;
             else PosP = pat.f[PosP-1] + 1;
   }
   if (PosP < LengthP) return -1;
   else return PosS-LengthP;
}
```

2018/9/11 © Ren-Song Tsay, NTHU, Taiwan 48