Information Science
11: String of Characters

Naonori Kakimura

垣村尚徳

kakimura@global.c.u-tokyo.ac.jp

2016/12/19

PEAK
Programs in English at Komaba

Schedule

- No class on Dec 26

- Remaining sessions
  - Jan 7 **(Sat)**
  - Jan 13, P2 [Make-up]
    - Answers of the exercises(Jan 7) and Q&A sessions

- Final Exam (90min.) on Jan 23 P5

Today’s Contents

- Review of Lec9 exercises
  - Complexity’s exercises
    - There are some incorrect/unclear parts
- String: a string of characters
  - How to use it in Ruby
  - Operations for strings
- Searching a keyword in a string
  - match
  - Submatch
  - Recursive definition
- Random numbers (another slide)
- Exercises

Review of Exercise1:

- A certain store has two software *A* and *B* to process experimental data. It is known that for an input of size N,
  - *A* runs in $O(N^2)$ time, and
  - *B* runs in $O(N \log_2(N))$ time.

- When we process 1000-record test data, *A* takes 1 second, while *B* takes 10 seconds.
- The target data has 1-million records.
- Which software is better to process the data? Explain why?

Take the condition into account

- *A* runs in $O(N^2)$ time, ・・・ $T_A = C_A N^2$
- *B* runs in $O(N \log_2(N))$ time ・・・ $T_B = C_B N \log_2(N)$
  - proportional to the values

- When we process 1000-record test data, *A* takes 1 second, while *B* takes 10 seconds.
  - A: $C_A (1000)^2 = 1[\text{sec}]$
  - B: $C_B (1000) \log_2(1000) = 10[\text{sec}]$

- Q. What is the time for $1000{,}000 = 1000^2$
  - A: $C_A (1000^2)^2$?
  - B: $C_B (1000^2) \log_2(1000)^2$?

Ans.: Take the condition into account

- *A* runs in $O(N^2)$ time, ・・・ $T_A = C_A N^2$
- *B* runs in $O(N \log_2(N))$ time ・・・ $T_B = C_B N \log_2(N)$

➢ When we process 1000-record test data, *A* takes 1 second while *B* takes 10 seconds.

- A: $C_A (1000)^2 = 1[\text{sec}]$
- B: $C_B (1000) \log_2(1000) = 10[\text{sec}]$

➢ Q. What is the time for $1000{,}000 = 1000^2$

- A: $C_A (1000^2)^2 = (1000)^2 C_A (1000)^2 = 1000{,}000\ [\text{sec}]$

- B: $C_B (1000^2) \log_2(1000)^2$
  $= 2000\ C_B (1000) \log_2(1000) = 20{,}000[\text{sec}]$

B is faster !

Exercise2: Counting Data

1. **(Past Exam 2010)** Suppose that an array a has size n and contains m kinds of positive integers. We want to store all the distinct integers of a to b, and also return the frequencies of occurrence in array c. For example, if a=[3,1,4,1,5,9,2,6,5,3], then n is 10 and m is 7. In this case, b contains [3,1,4,5,9,2,6], and c contains [2,2,1,2,1,1,1].

   (a) The following program is a program to compute b and c from a. Describe the computational complexity $\boxed{\text{using } n \text{ and } m.}$ Note that the parameters b and c are supposed to be arrays of size m. We suppose that each entry in array b is initialized to be 0.

```
def intcount(a, b, c)
   for i in 0..(a.length()-1)
      x = a[i]
      j = 0
      while b[j] != 0 && b[j] != x
         j = j + 1
      end
      if b[j] == 0
         b[j] = x
         c[j] = 1
      else
         c[j] = c[j] + 1
      end
   end
end
```

Example: Counting Data

- Before execution

Begin with arrays of 0's

a=[3,1,4,3,5,9,2,6,5,3]

b=[0,0,0,0,0,0,0] (m=7)
c=[0,0,0,0,0,0,0]

- At the end of each iteration

i=0 b=[3,0,0,0,0,0,0]
c=[1,0,0,0,0,0,0]

$\longrightarrow$

i=1 b=[3,1,0,0,0,0,0]
c=[1,1,0,0,0,0,0]

$\longrightarrow$

i=2 b=[3,1,4,0,0,0,0]
c=[1,1,1,0,0,0,0]

$\rightarrow$

i=3 b=[3,1,4,0,0,0,0]
c=[2,1,1,0,0,0,0]

```
def intcount(a, b, c)
   for i in 0..(a.length()-1)
      x = a[i]
      j = 0
      while b[j] != 0 && b[j] != x
         j = j + 1
      end
      if b[j] == 0
         b[j] = x
         c[j] = 1
      else
         c[j] = c[j] + 1
      end
   end
end
```

Look at a[i]

Check if some b[j] == a[i]
until b[j] becomes 0

If ended by b[j]==0,
b doesn't have a[i], and
make a new entry in b

If ended by b[j]==a[i],
increment c[j] by one

Example: Counting Data

Worst case Complexity in terms of n & m

- Time complexity
  - #(iteration of i) ×#(operations for the while) = O(nm)

    n(=length of a) (the worst case for blue part) = O(m)

- At the end of each iteration

i=0 b=[3,0,0,0,0,0,0]
c=[1,0,0,0,0,0,0]

$\rightarrow$

i=1 b=[3,1,0,0,0,0,0]
c=[1,1,0,0,0,0,0]

$\longrightarrow$

i=2 b=[3,1,4,0,0,0,0]
c=[1,1,1,0,0,0,0]

$\rightarrow$

i=3 b=[3,1,4,0,0,0,0]
c=[2,1,1,0,0,0,0]

```
def intcount(a, b, c)
  for i in 0..(a.length()-1)
    x = a[i]
    j = 0
    while b[j] != 0 && b[j] != x
      j = j + 1
    end
    if b[j] == 0
      b[j] = x
      c[j] = 1
    else
      c[j] = c[j] + 1
    end
  end
end
```

Look at a[i]

Check if some b[j] == a[i] until b[j] becomes 0

If ended by b[j]==0, b doesn't have a[i], and make a new entry in b

If ended by b[j]==a[i], increment c[j] by one

What if a is sorted

(b) Suppose that a is sorted, that is, elements in a is ordered in nondecreasing order. Modifying the above program, make a new function intcount(a,b,c) that runs in $O(n)$ time.

Ex. a=[10, 10, 9, 8, 8, 6, 6, 6, 3, 3, 2, 2, 1]

Hint: suffices to detect the change of numbers in a

```
def intcount(a, b, c)
    Write something
  for i in 0..(a.length()-1)
    Write something
  end
end
```

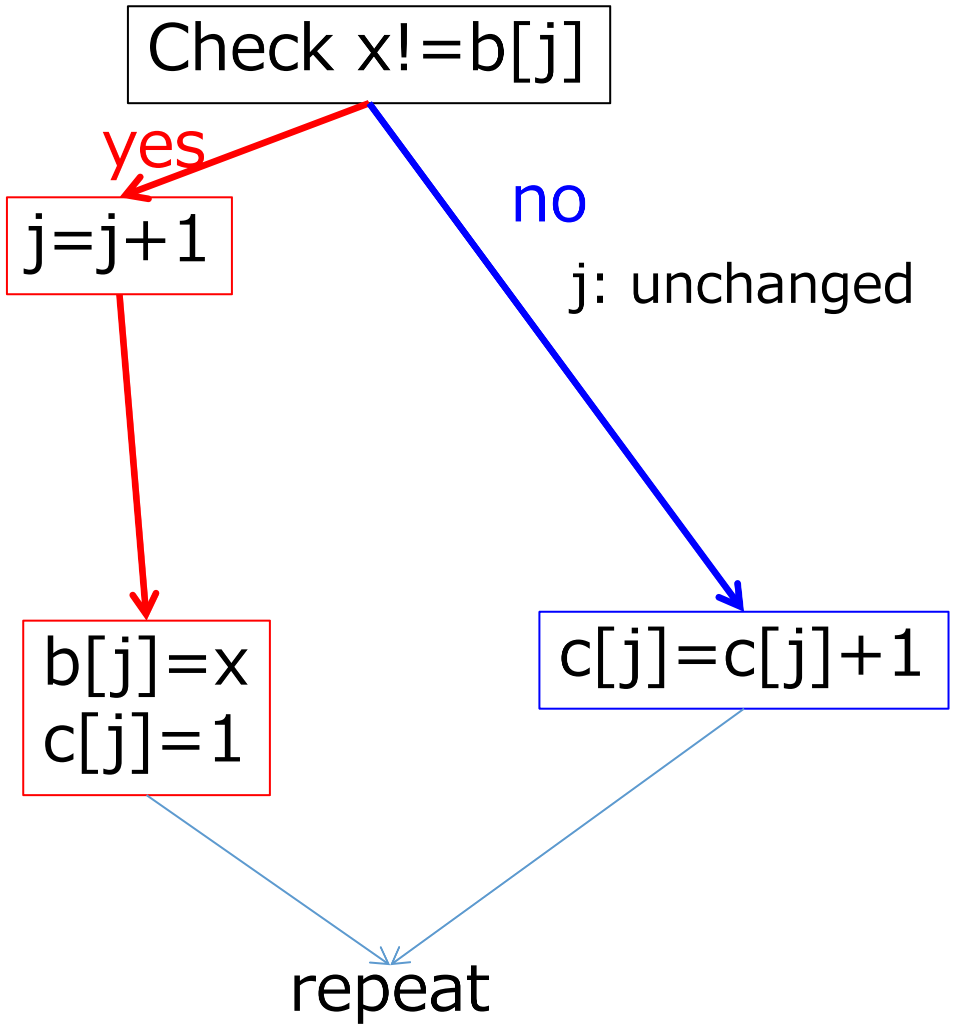
b=[0, 0, 0, 0, 0, 0, 0]

$\uparrow$

Keep current index j of b

Repeatedly
  read a from a[0] to the end
  If a[i]== b[j] increment c[j]
  Otherwise,
  move to next b[j] and c[j]=1

Solution



```
def intcount(a, b, c)
   j = 0
   b[0] = a[0]
   for i in 0..(a.length()-1)
      x = a[i]
      if b[j] != x
         j = j + 1
         b[j] = x
         c[j] = 1
      else
         c[j] = c[j] + 1
      end
   end
end
```

Today’s Contents

- Review of Lec9 exercises
  - Complexity’s exercises
- String: a string of characters
  - How to use it in Ruby
  - Operations for strings
- Searching a keyword in a string
  - match
  - Submatch
  - Recursive definition
- Random numbers (another slide)
- Exercises

String

- A sequence of characters
  - an array of character
  - Put " " covering something in Ruby

```
irb(main):003:0> "hello"
=> "hello"
irb(main):004:0> hello
(error: Ruby tries to find a variable hello)
irb(main):003:0> "Hi, " + "hello"
=> "Hi, hello"
```

"" means a string

Concatenate strings

String of Numbers and Integers

```
irb(main):003:0> "123"
=> "123"
irb(main):004:0> 123
=> 123
irb(main):010:0> 123+123                Sum of integers
=> 246
irb(main):011:0> "123"+"123"       Concatenate strings
=> "123123"
irb(main):012:0> "123"+123
TypeError: can't convert Fixnum into String
  from (irb):12:in `+'
  from (irb):12
  from :0
```

Variables can Contain a String

Operations for strings

```
irb(main):003:0> s = "abra"
=> "abra"
irb(main):004:0> t = "cadabra"
=> "cadabra"
irb(main):005:0> u = s + t
=> "abracadabra"

irb(main):005:0> u = s - t
(error message)
```

concatenation

No operation “-”
is define for string

Functions for Strings

```
s = "abra", t = "cadabra"
```

```
irb(main):009:0> "abra".length()
=> 4
irb(main):010:0> (s + t).length()
=> 11
irb(main):011:0> s[0..0]
=> "a"
irb(main):012:0> s[1..2]
=> "br"
irb(main):013:0> t[1..(t.length()-1)]
=> "adabra"
```

Length of string

Same as s[0]

the 0th element in s

(like an array)

the 1st to 2nd element in s

the 1st to the last element in t

Transform Integers to String

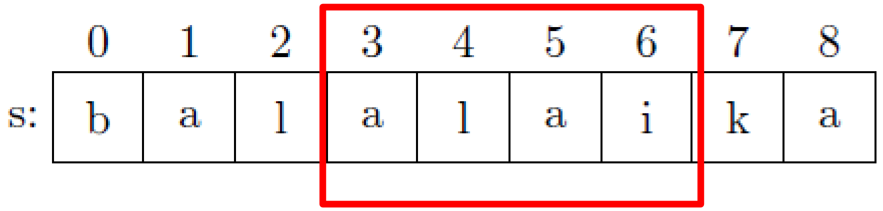
```
irb(main):013:0> "123"+123.to_s()
=> "123123"                        Change an integer to a string
irb(main):013:0> "123".to_i() + 123
=> 246                       Change a string to integers

irb(main):014:0> i = 10
=> 10
irb(main):015:0> i.to_s()
=> "10"                            Change integers to a string
irb(main):016:0> (i+1).to_s()
=> "11"
```

Today's Contents

- Review of Lec9 exercises
  - Complexity's exercises
- String: a string of characters
  - How to use it in Ruby
  - Operations for strings
- Searching a keyword in a string
  - match
  - Submatch
  - Recursive definition
- Random numbers (another slide)
- Exercises

Searching a Substring

- Find a given keyword in a string
  - Ex. Analysis of DNA sequences

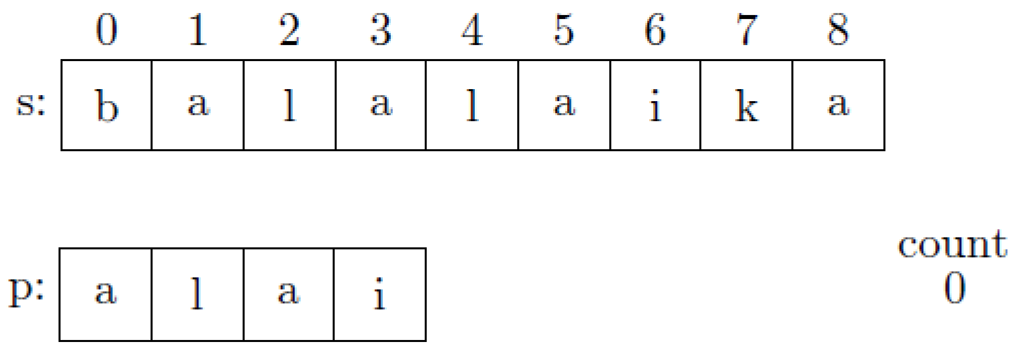


Q. Is there "alai", denoted by p, in the sequence?

Searching a Substring

- Find a given keyword in a string
  - Ex. Analysis of DNA sequences

count how many
characters are matched

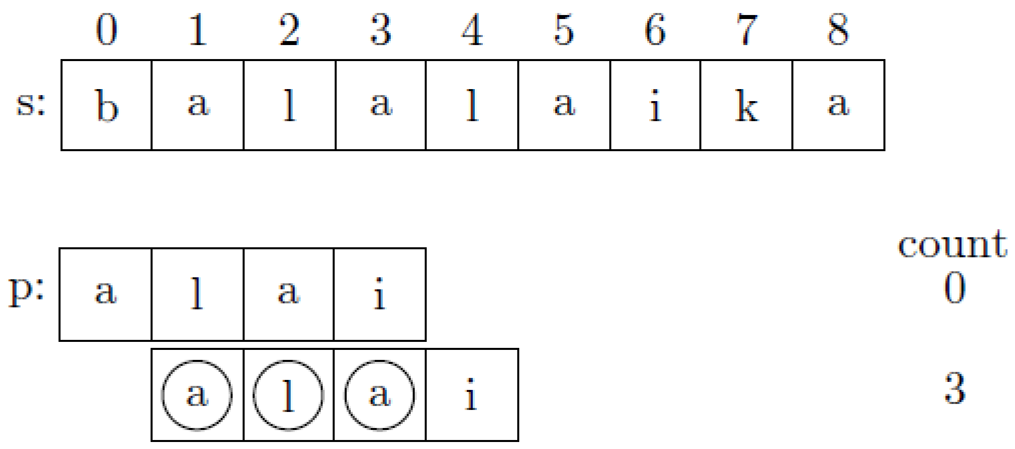


Searching a Substring

- Find a given keyword in a string
  - Ex. Analysis of DNA sequences

count how many characters are matched

shift by one

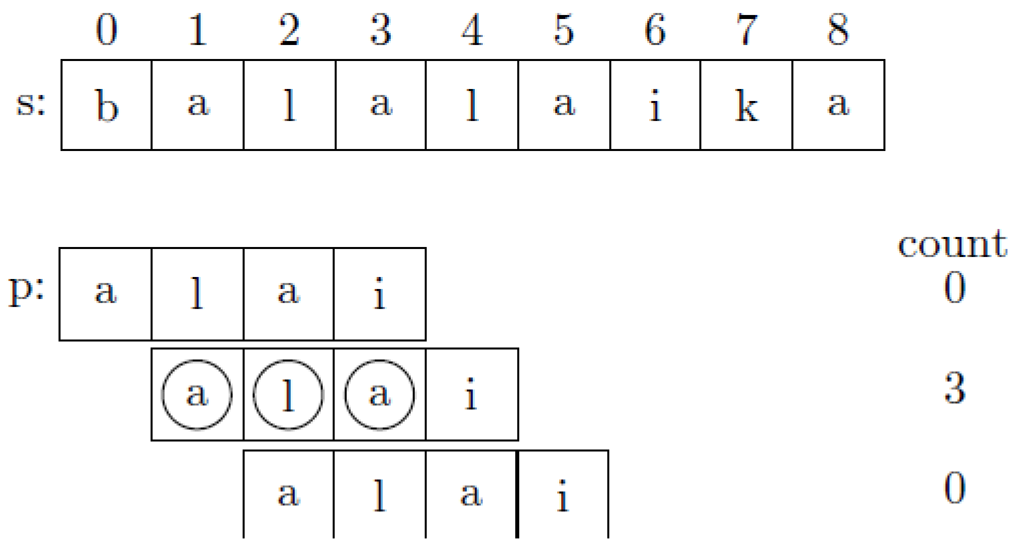


Searching a Substring

- Find a given keyword in a string
  - Ex. Analysis of DNA sequences

count how many
characters are matched

shift by one

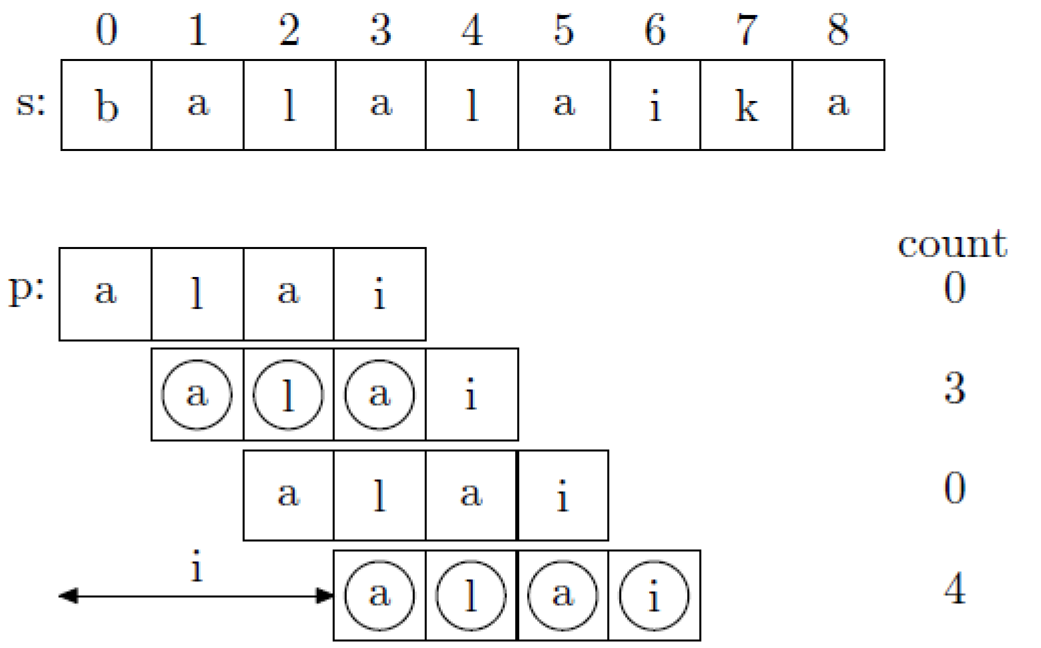


Searching a Substring

- Find a given keyword in a string
  - Ex. Analysis of DNA sequences

count how many characters are matched

shift by one



# matches

For each i in 0..8,
  (where i represents the place of p in s)
  compare s[i..(i+4)] and p by counting how many characters are matched from beginning

Searching Keywords

```
def match(s,p)
  i = 0
  w = p.length()
  while submatch(s,i,p,w) < w
    i = i + 1
  end
  i
end
```

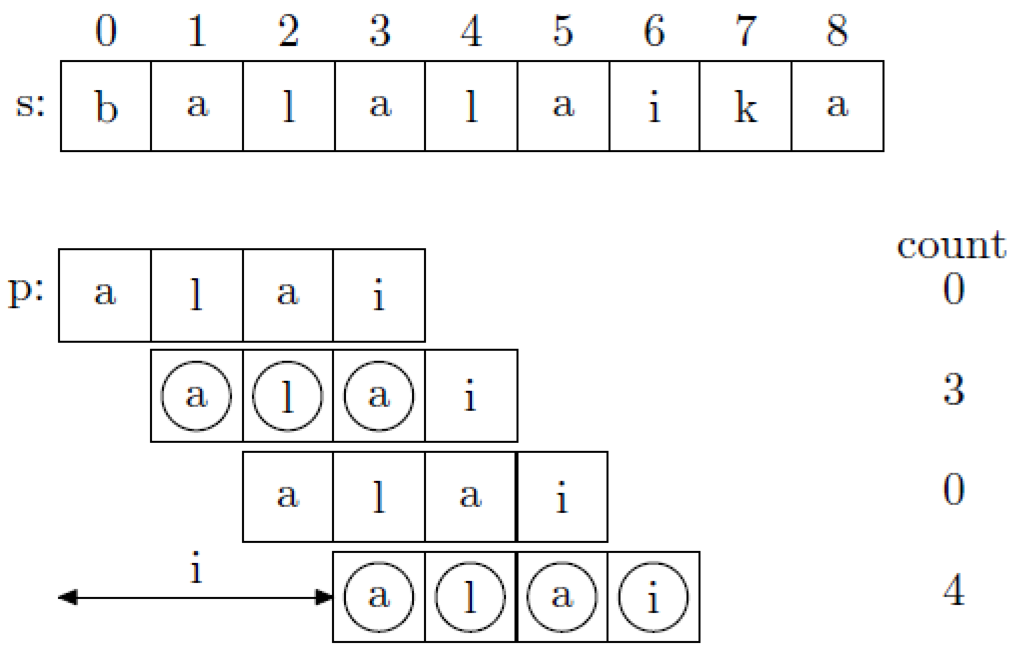
Function that counts how many characters are matched from the beginning

Decide if
# matched = length of p

Downloadable from ITC-LMS

Framework of Function submatch(s,i,p,w)

- Four parameters
  - s: a string of characters which we want to examine
  - i: the index of s that p begins
  - p: a (short) string
  - w: the length of p



Details will be described later

Demonstration

	0	1	2	3	4	5	6	7	8
s:	b	a	l	a	l	a	i	k	a

	0	1	2	3
p:	a	l	a	i

The length of p

submatch(s, 0, p, 4)

The index that p starts

```
def match(s,p)
  i = 0
  w = p.length()
  while submatch(s,i,p,w) < w
    i = i + 1
  end
  i
end
```

Demonstration

	0	1	2	3	4	5	6	7	8
s:	b	a	l	a	l	a	i	k	a

	0	1	2	3
p:	a	l	a	i

The length of p

submatch(s, 0, p, 4) => 0

The index that p starts

```
def match(s,p)
    i = 0
    w = p.length()
    while submatch(s,i,p,w) < w
        i = i + 1
    end
    i
end
```

Demonstration

	0	1	2	3	4	5	6	7	8
s:	b	a	l	a	l	a	i	k	a

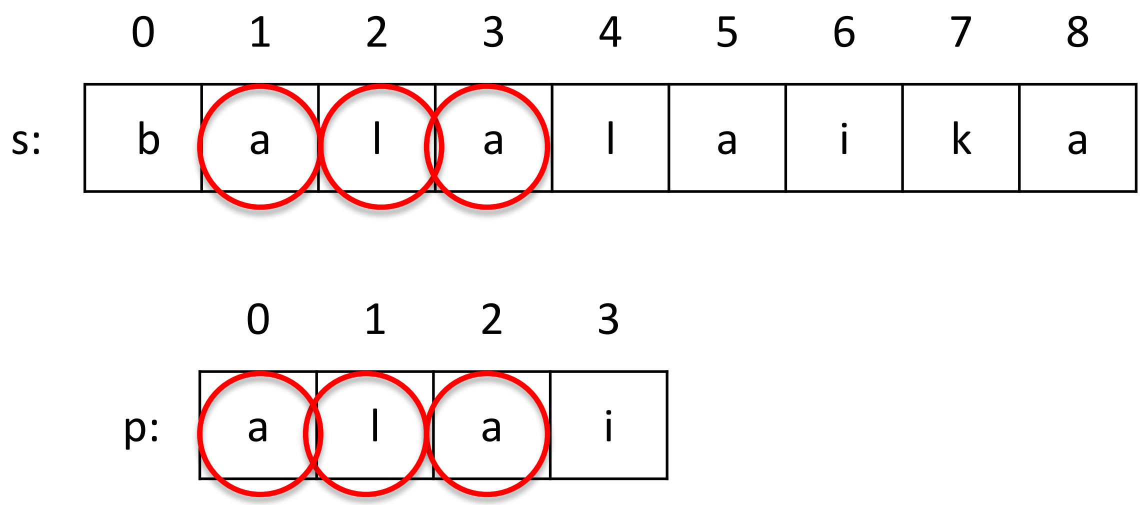
	0	1	2	3
p:	a	l	a	i

submatch(s, 1, p, 4)

p is shifted by one

```
def match(s,p)
    i = 0
    w = p.length()
    while submatch(s,i,p,w) < w
        i = i + 1
    end
    i
end
```

Demonstration



submatch(s, 1, p, 4) => 3

```
def match(s,p)
  i = 0
  w = p.length()
  while submatch(s,i,p,w) < w
    i = i + 1
  end
  i
end
```

Demonstration

	0	1	2	3	4	5	6	7	8
s:	b	a	l	a	l	a	i	k	a

	0	1	2	3
p:	a	l	a	i

```
def match(s,p)
   i = 0
   w = p.length()
   while submatch(s,i,p,w) < w
      i = i + 1
   end
   i
end
```

submatch(s, 2, p, 4)

p is shifted by one

Demonstration

	0	1	2	3	4	5	6	7	8
s:	b	a	l	a	l	a	i	k	a

	0	1	2	3
p:	a	l	a	i

submatch(s, 2, p, 4) => 0

```
def match(s,p)
   i = 0
   w = p.length()
   while submatch(s,i,p,w) < w
      i = i + 1
   end
   i
end
```

Demonstration

	0	1	2	3	4	5	6	7	8
s:	b	a	l	a	l	a	i	k	a

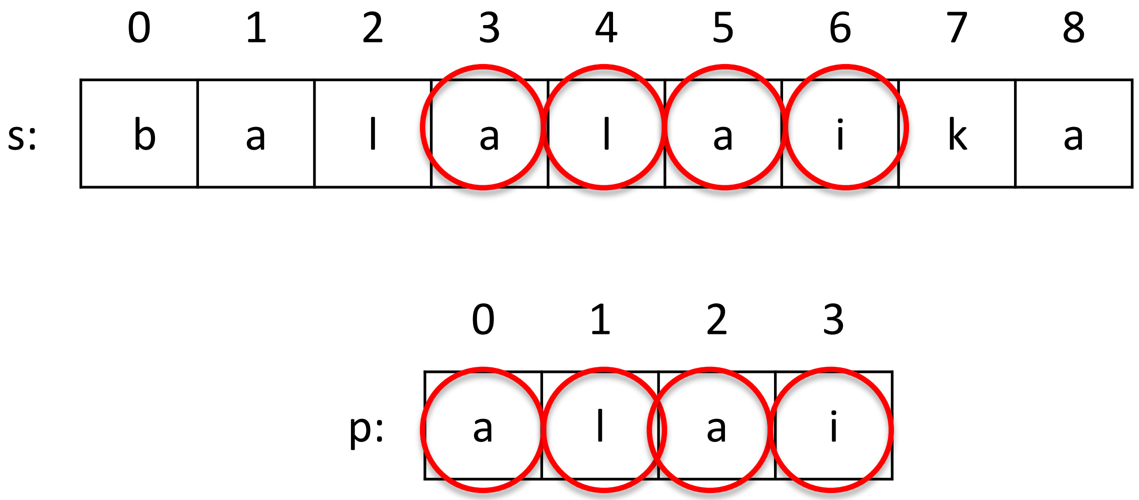
	0	1	2	3
p:	a	l	a	i

submatch(s, 3, p, 4)

p is shifted by one

```
def match(s,p)
  i = 0
  w = p.length()
  while submatch(s,i,p,w) < w
    i = i + 1
  end
  i
end
```

Demonstration



submatch(s, 3, p, 4) => 4

```
def match(s,p)
   i = 0
   w = p.length()
   while submatch(s,i,p,w) < w
      i = i + 1
   end
   i
end
```

Demonstration

	0	1	2	3	4	5	6	7	8
s:	b	a	l	a	l	a	i	k	a

	0	1	2	3
p:	a	l	a	i

```
def match(s,p)
   i = 0
   w = p.length()
   while submatch(s,i,p,w) < w
      i = i + 1
   end
   i
end
```

submatch(s, 3, p, 4) => 4

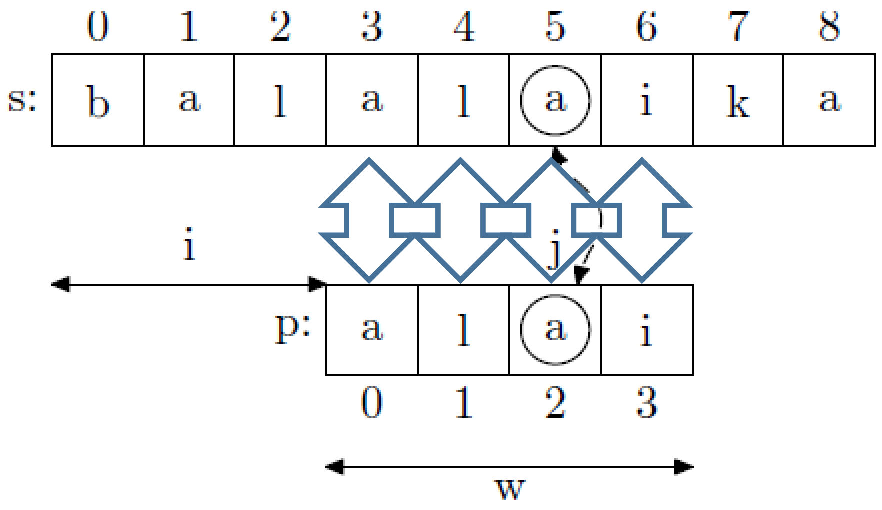
When i==3,
submatch(s, i, p, w) < w is false

Today’s Contents

- Review of Lec9 exercises
  - Complexity’s exercises
- String: a string of characters
  - How to use it in Ruby
  - Operations for strings
- Searching a keyword in a string
  - match
  - Submatch
  - Recursive definition
- Random numbers (another slide)
- Exercises

Procedure submatch(s,i,p,w)

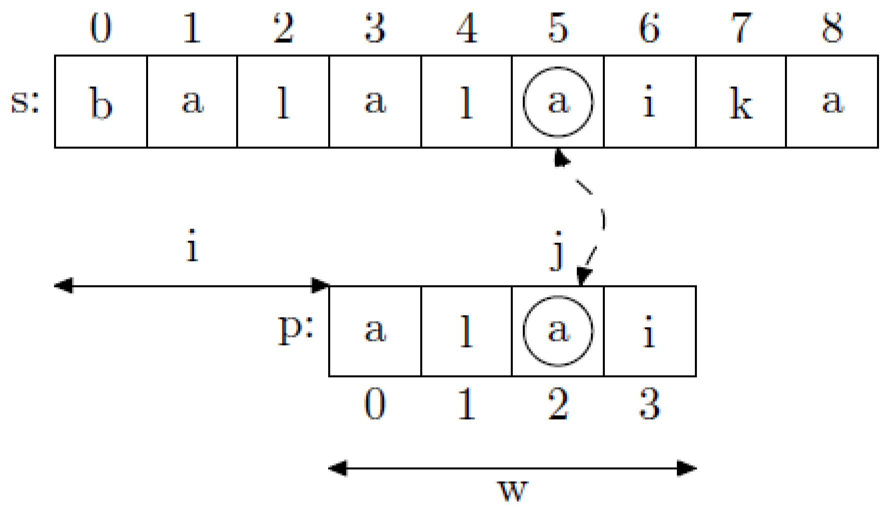
- Repeatedly compare corresponding elements in s and p at each place



Procedure submatch(s,i,p,w)

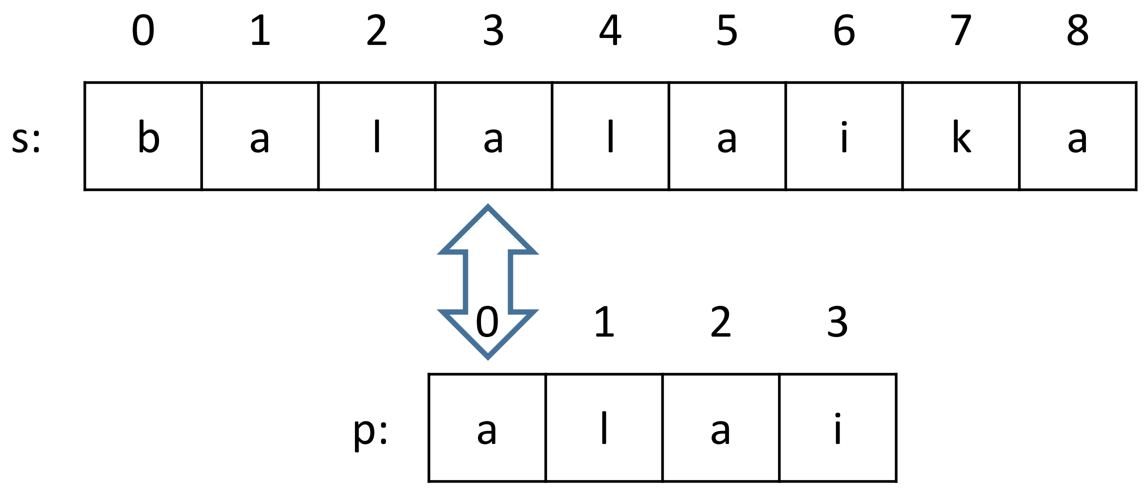
```
def submatch (s,i,p,w)
    j = 0  # Maintain #matched characters
    while j < w && s[(i+j)..(i+j)] == p[j..j]
        j = j + 1
    end
    j
end
```

Compare
the (i+j)th in s and the jth in p



while they coincide
we increment j by one

Procedure submatch(s,i,p,w)



submatch(s, 3, p, 4)

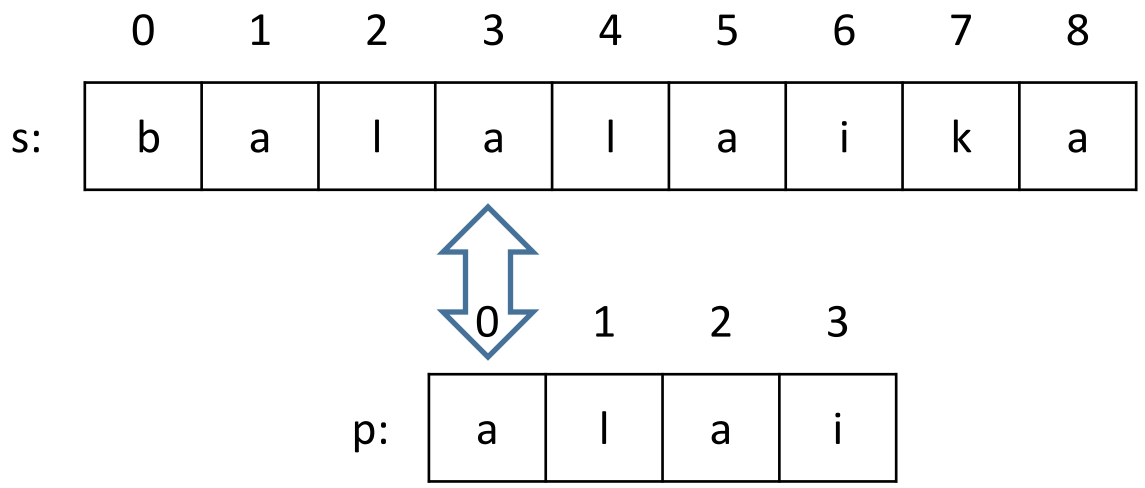
i==3 w==4

j==0

j < w && s[(i+j)..(i+j)] == p[j..j]

```
def submatch (s,i,p,w)
   j = 0
   while j < w && s[(i+j)..(i+j)] == p[j..j]
      j = j + 1
   end
   j
end
```

Procedure submatch(s,i,p,w)



submatch(s, 3, p, 4)

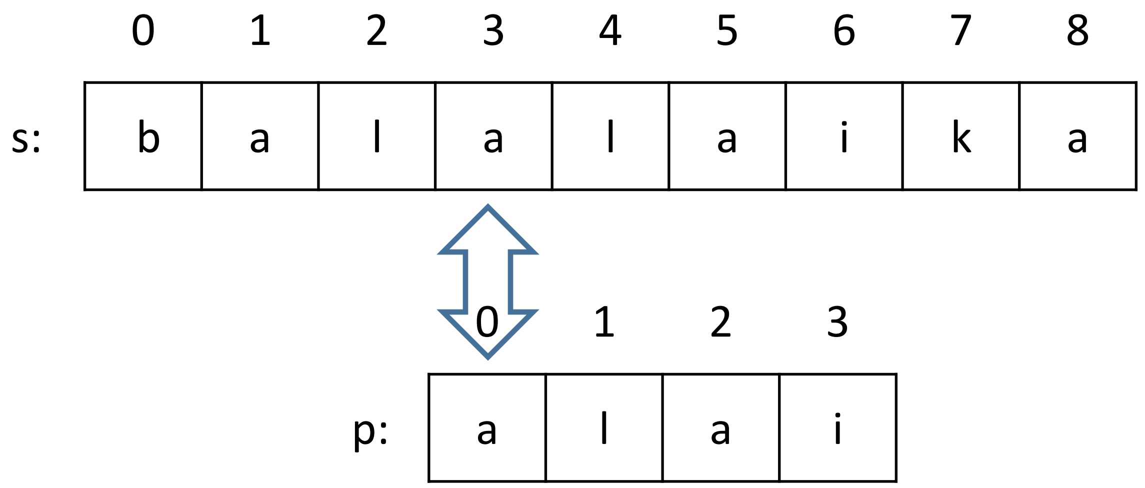
i==3 w==4

j==0

$j < w \;\&\&\; \underbrace{s[(i+j)..(i+j)]}_{\Rightarrow\ "a"} == \underbrace{p[j..j]}_{\Rightarrow\ "a"}$

```
def submatch (s,i,p,w)
   j = 0
   while j < w && s[(i+j)..(i+j)] == p[j..j]
      j = j + 1
   end
   j
end
```

Procedure submatch(s,i,p,w)



submatch(s, 3, p, 4)

i==3 w==4

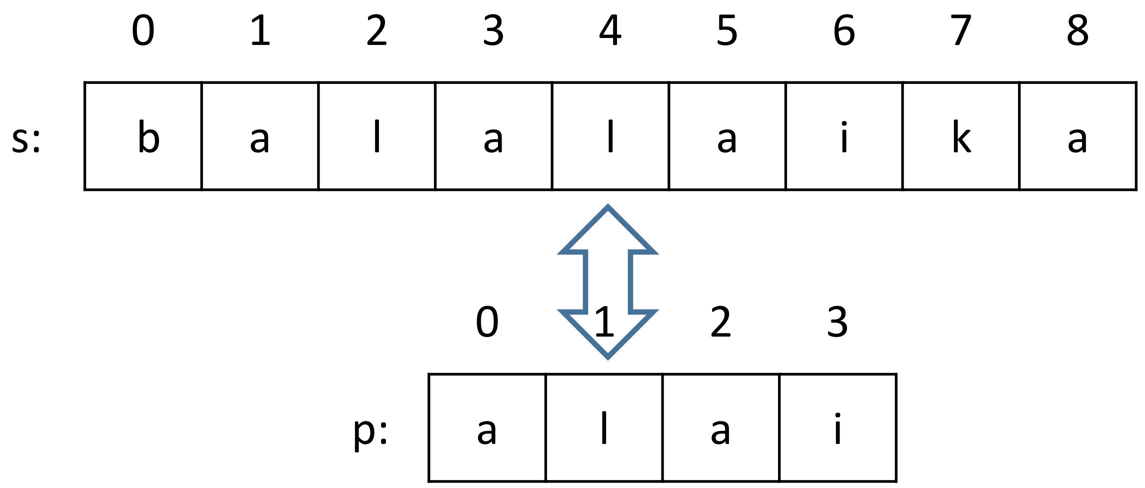
j==0

j < w && s[(i+j)..(i+j)] == p[j..j]

=> true

```
def submatch (s,i,p,w)
  j = 0
  while j < w && s[(i+j)..(i+j)] == p[j..j]
    j = j + 1
  end
  j
end
```

Procedure submatch(s,i,p,w)



submatch(s, 3, p, 4)

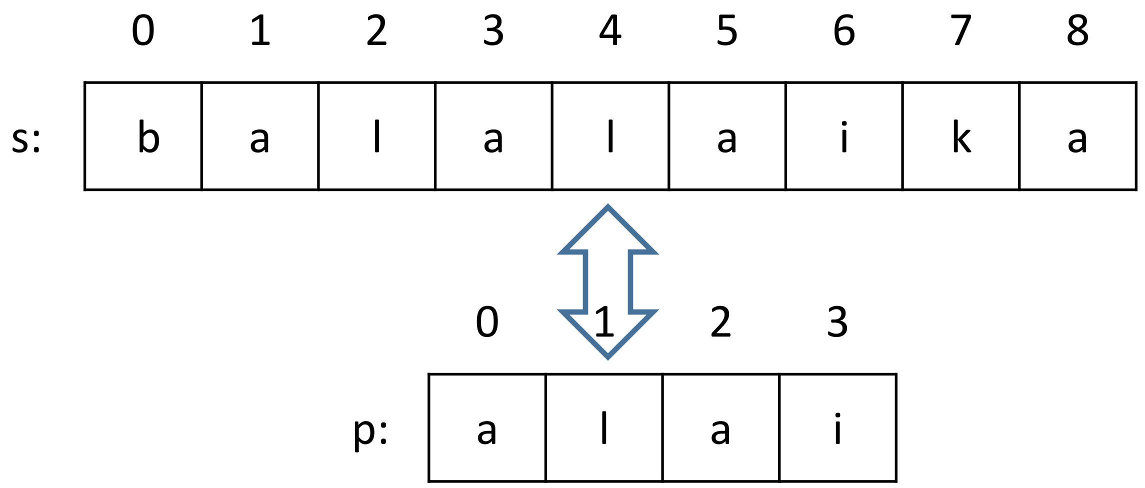
i==3 w==4

j==1

j < w && s[(i+j)..(i+j)] == p[j..j]

```
def submatch (s,i,p,w)
  j = 0
  while j < w && s[(i+j)..(i+j)] == p[j..j]
    j = j + 1
  end
  j
end
```

Procedure submatch(s,i,p,w)



submatch(s, 3, p, 4)

i==3 w==4

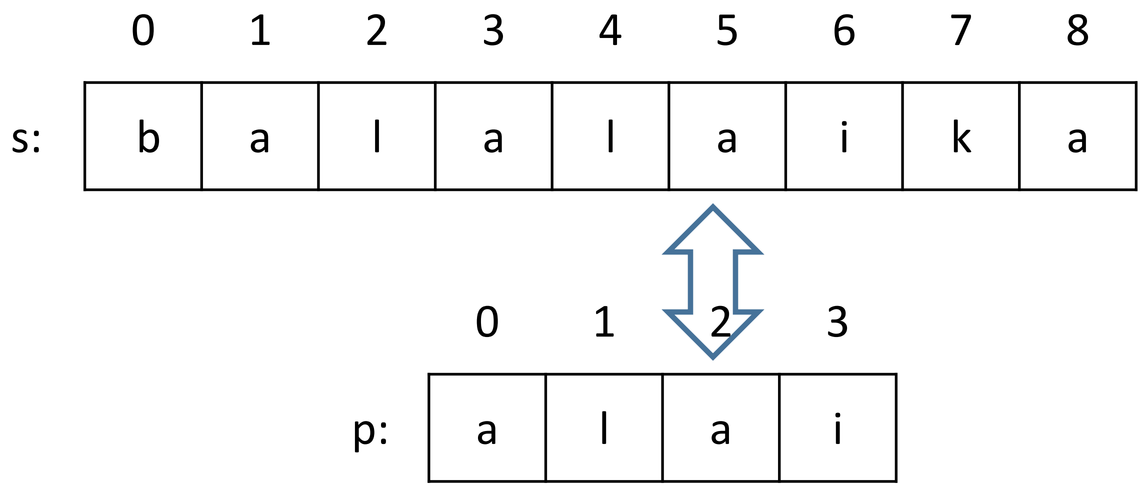
j==1

j < w && s[(i+j)..(i+j)] == p[j..j]

=> true

```
def submatch (s,i,p,w)
  j = 0
  while j < w && s[(i+j)..(i+j)] == p[j..j]
    j = j + 1
  end
  j
end
```

Procedure submatch(s,i,p,w)



submatch(s, 3, p, 4)

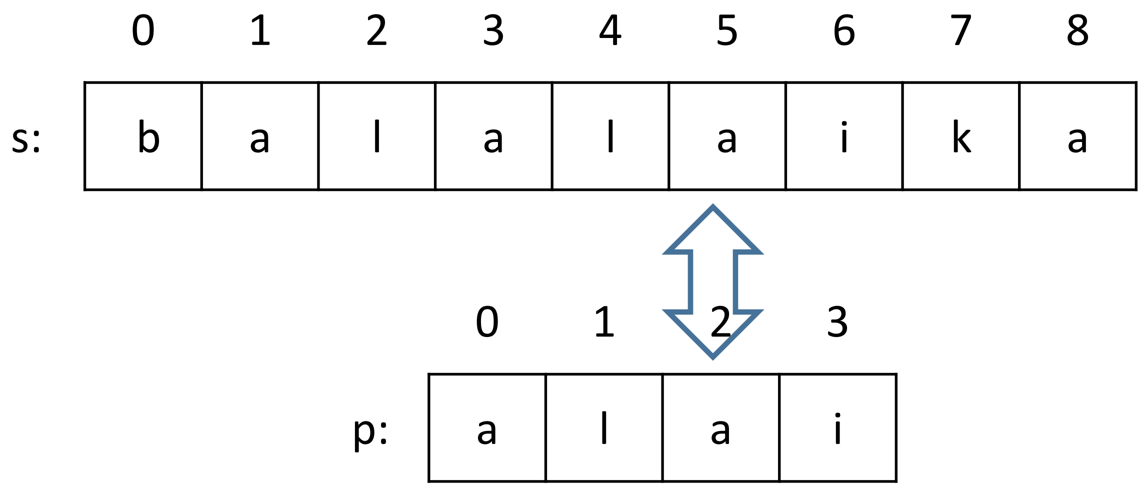
i==3 w==4

j==2

j < w && s[(i+j)..(i+j)] == p[j..j]

```
def submatch (s,i,p,w)
   j = 0
   while j < w && s[(i+j)..(i+j)] == p[j..j]
      j = j + 1
   end
   j
end
```

Procedure submatch(s,i,p,w)



submatch(s, 3, p, 4)

i==3 w==4

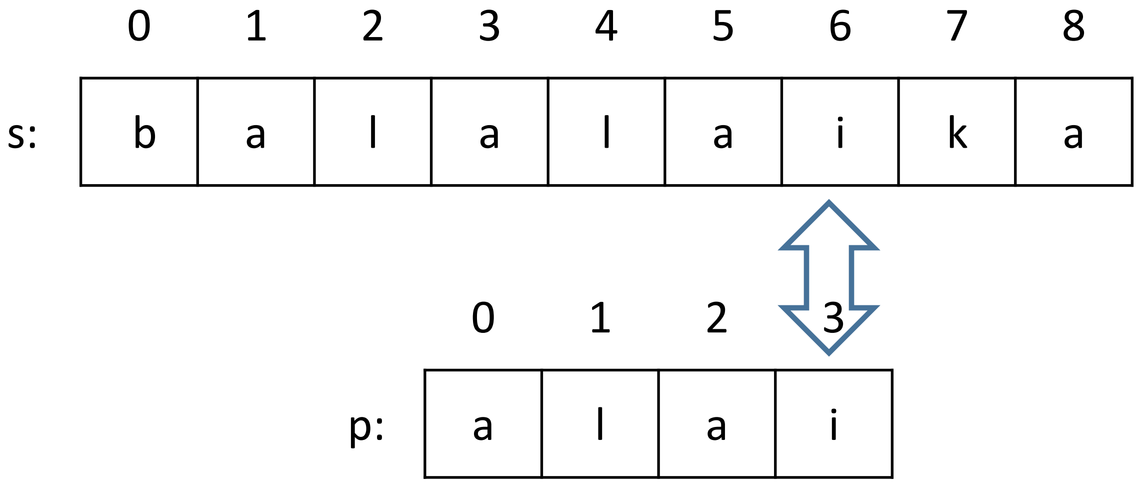
j==2

j < w && s[(i+j)..(i+j)] == p[j..j]

=> true

```
def submatch (s,i,p,w)
  j = 0
  while j < w && s[(i+j)..(i+j)] == p[j..j]
    j = j + 1
  end
  j
end
```

Procedure submatch(s,i,p,w)



submatch(s, 3, p, 4)

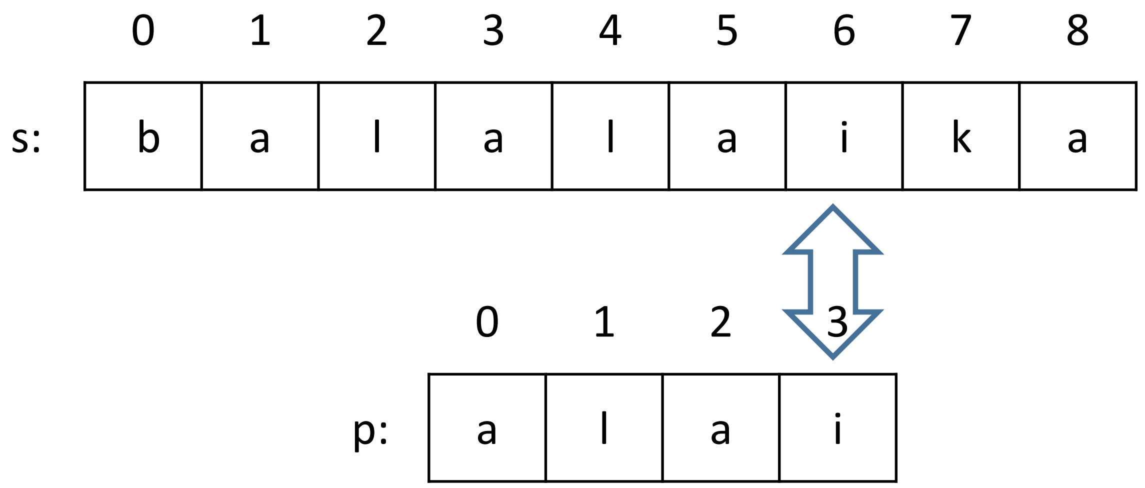
i==3 w==4

j==3

j < w && s[(i+j)..(i+j)] == p[j..j]

```
def submatch (s,i,p,w)
   j = 0
   while j < w && s[(i+j)..(i+j)] == p[j..j]
      j = j + 1
   end
   j
end
```

Procedure submatch(s,i,p,w)



submatch(s, 3, p, 4)

i==3 w==4

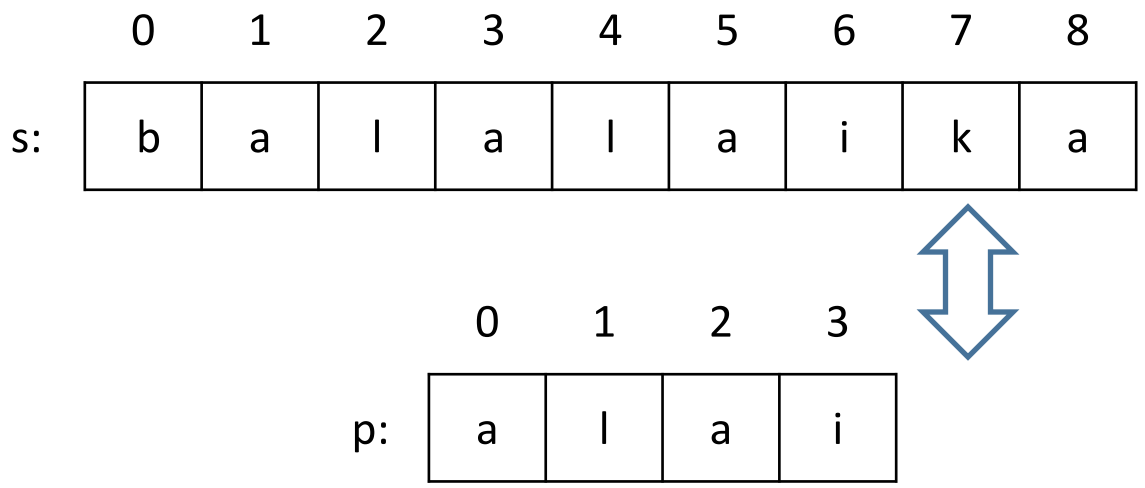
j==3

j < w && s[(i+j)..(i+j)] == p[j..j]

=> true

```
def submatch (s,i,p,w)
  j = 0
  while j < w && s[(i+j)..(i+j)] == p[j..j]
    j = j + 1
  end
  j
end
```

Procedure submatch(s,i,p,w)



submatch(s, 3, p, 4)

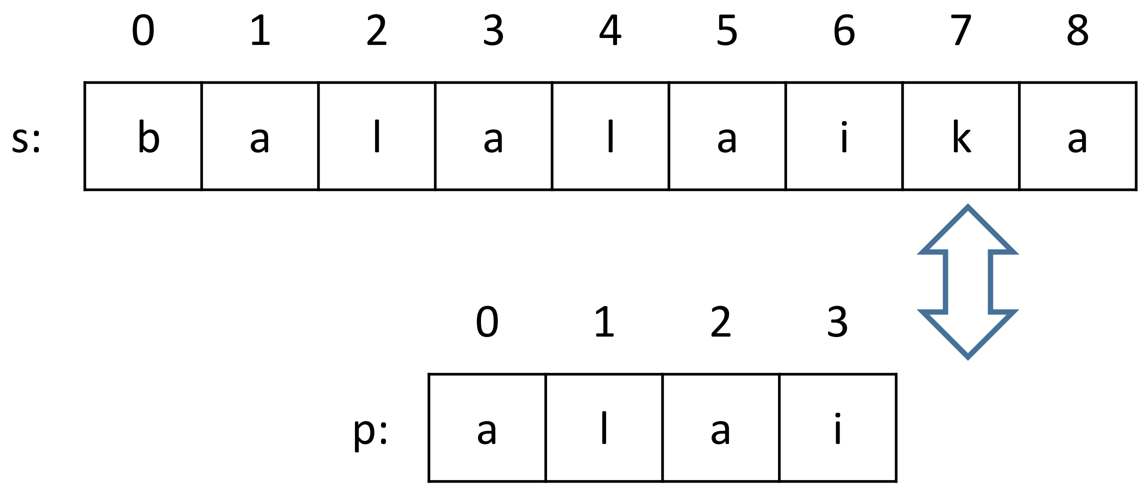
i==3 w==4

j==4

j < w && s[(i+j)..(i+j)] == p[j..j]

```
def submatch (s,i,p,w)
  j = 0
  while j < w && s[(i+j)..(i+j)] == p[j..j]
    j = j + 1
  end
  j
end
```

Procedure submatch(s,i,p,w)



submatch(s, 3, p, 4)

i==3 w==4

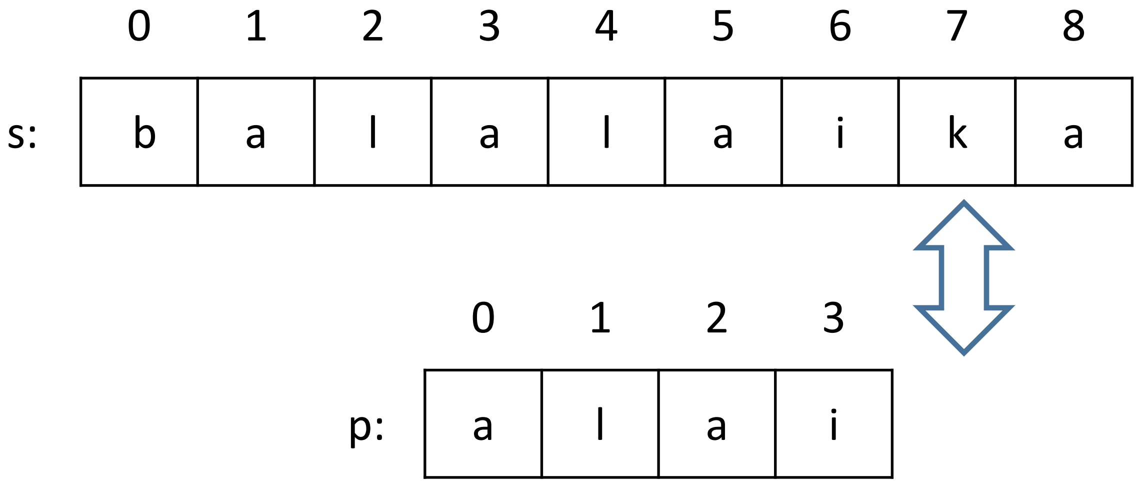
j==4

$\underline{j} < \underline{w}$ && s[(i+j)..(i+j)] == p[j..j]

=> 4   => 4   => false

```
def submatch (s,i,p,w)
  j = 0
  while j < w && s[(i+j)..(i+j)] == p[j..j]
    j = j + 1
  end
  j
end
```

Procedure submatch(s,i,p,w)



We do not check whether
s[(i+j)..(i+j)] == p[j..j]

submatch(s, 3,

i==3 w==4

j==4

j < w && s[(i+j)..(i+j)] == p[j..j]

=> false

```
n (s,i,p,w)

w && s[(i+j)..(i+j)] == p[j..j]
    j = j + 1
  end
  j
end
```

Today’s Contents

Searching Key Words

```
irb(main):007:0> match("balalaika", "alai")
=> 3
irb(main):008:0> match("hualalai", "alai")
=> 4
irb(main):009:0> match("balalaika", "aa")
```

Since “aa” is not included in “balalaika”,
it never terminates.
Press Control-C

Remarks

- Modifying the function match, we can make the function match_safe(s,p) that if it has no p, then return -1.

Modified Programs

```
def match_safe(s,p)
  i = 0
  w = p.length()
  while i + w <= s.length() && submatch(s,i,p,w) < w
    i = i + 1
  end
  if i + w > s.length()
    i = -1
  end
  i
end
```

Check if i+w is smaller than length of s

If “while” terminates by “i+w>s.length()” there is no p in s

Today's Contents

- Review of Lec9 exercises
  - Complexity's exercises
- String: a string of characters
  - How to use it in Ruby
  - Operations for strings
- Searching a keyword in a string
  - match
  - submatch
  - Remarks
  - Recursive definition
- Random numbers (another slide)
- Exercises

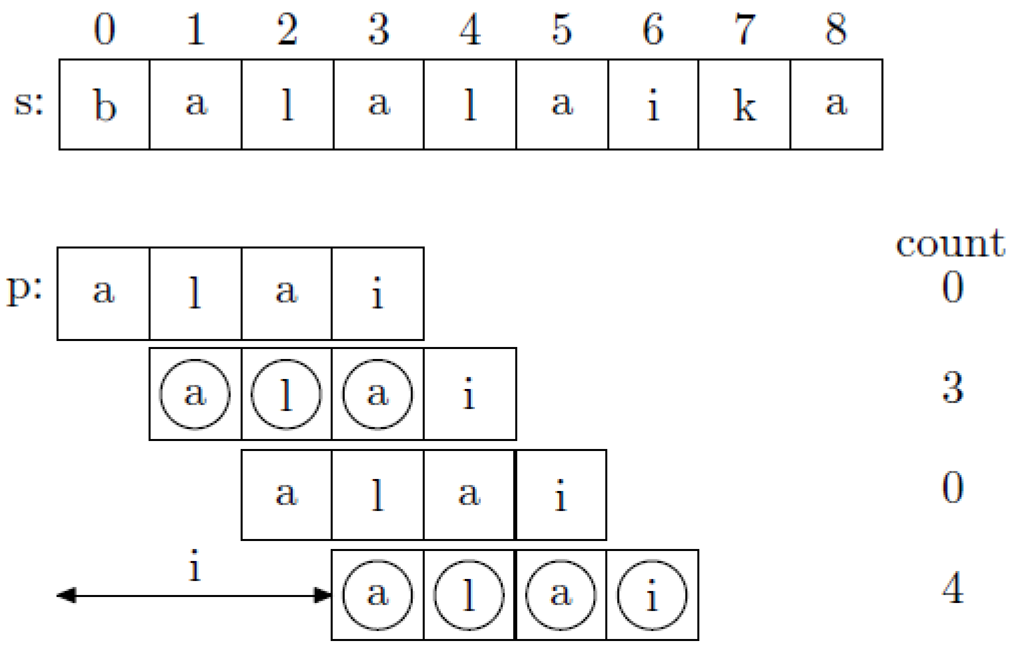
Using Recursion (with same complexity)

- Let n=s.length()-1: the last index of s

- match_r(s, p, i)
  - Return i if p coincides with s[i..(i+w-1)], and ow,
  - Return match_r (s, p, i+1)

Focus on s[(i+1)..n]

Solution: match_r(s,p,0)



Searching Keywords Recursively

```
def match_r(s,p,i)
    w = p.length()
    n = s.length()-1
    if submatch(s,i,p,w) == w
        i
    else
        match_r(s, p, i+1)
    end
end
```

Compare s[i..(i+w-1)] and p

Recursive call

Today's Contents

- Review of Lec9 exercises
  - Complexity's exercises
- String: a string of characters
  - How to use it in Ruby
  - Operations for strings
- Searching a keyword in a string
  - match
  - submatch
  - Remarks
  - Recursive definition
- Random numbers (another slide)
- Exercises

Today’s Contents

- Review of Lec9 exercises
  - Complexity’s exercises
- String: a string of characters
  - How to use it in Ruby
  - Operations for strings
- Searching a keyword in a string
  - match
  - submatch
  - Remarks
  - Recursive definition
- Random numbers (another slide)
- Exercises

Exercise 1:

- What is the computational complexity of the function match?
  - Let n be the length of s, and m be the length of p,
  - Represent the complexity using n and m
    - We may assume that p exists in s

- Hint: It suffices to consider the worst case:
  - Consider the following case & estimate #operations
    - $s=[\underbrace{aa\cdots aaaa}_{n-1}b]$
    - $p=[\underbrace{aa\cdots a}_{m-1}b]$

Exercise 2: Reverse a String

- Given a string, we want to reverse it
  - Ex. For s="abcdef", the output is "fedcba"

- Requirement
  - Use “while”
  - Do not use s.reverse() nor s.split("").reverse().join()
    - Functions already implemented in Ruby
  - We can use “+” for adding two strings

Exercise 2: Reverse a String

- Fill in the question marks

```
def reverse(s)
  result = ""  # empty string(length 0)
  i = ??
  while i >= 0 do
    ??
    i = i - 1
  end
  result  # return the reversed string
end
```

Try

```
irb(main):003:0> reverse("Ruby language")
```

Optional Exercises

- Rewrite the function “reverse” using recursion

Deadline of Today’s Exercises 1 & 2

- **By Jan. 6（Fri） 23:59**
  - Explain how you obtain solutions, not only solutions
- Through ITC-LMS
  - It is OK to submit a hand-written one if you want to do homework by hand
    - Recommend to scan it and send it by e-mail
    - You can hand in a hard one

Next Session: **Jan. 7(Sat)**

- Similarity detection of two strings
  - Dynamic programming:
    - technique for algorithm design