

関数から計算へ・文字列

進捗状況の確認

1. Sierpinskiのカーペットと三角形が再帰プログラムで書けた
2. Sierpinskiのカーペット(再帰)だけが書けた
3. Sierpinskiの三角形(再帰)だけが書けた
4. Sierpinskiの三角形がcombinationの配列で書けた
5. まだ書いていない

文字列

```
irb(main):003:0> s = "abra"
```

```
=> "abra"
```

```
irb(main):004:0> t = "cadabra"
```

```
=> "cadabra"
```

```
irb(main):005:0> u = s + t
```

```
=> "abracadabra"
```

```
irb(main):006:0> "123" + "456"
```

```
=> "123456"
```

```
irb(main):009:0> s.length()
```

```
=> 4
```

```
irb(main):010:0> (s + t).length()
```

```
=> 11
```

```
irb(main):011:0> s[0..0]      # s[0] でも良い
```

```
=> "a"
```

```
irb(main):012:0> s[1..2]
```

```
=> "br"
```

```
irb(main):013:0> t[1..(t.length()-1)]
```

```
=> "adabra"
```

文字列を作る: 2つの " 記号の間にある文字や記号は文字列を作る。

文字列の結合: $\boxed{\text{式}_1} + \boxed{\text{式}_2}$ という式は、 $\boxed{\text{式}_1}$, $\boxed{\text{式}_2}$ の値が文字列のとき、それらの文字列をつなげた文字列を作る。

文字列の長さ: $\boxed{\text{式}_1}.\text{length}()$ という式は、 $\boxed{\text{式}_1}$ が表わす文字列の長さを求める。

部分文字列: $\boxed{\text{式}_1}[\boxed{\text{式}_2}..\boxed{\text{式}_3}]$ という式は、 $\boxed{\text{式}_1}$ が表わす文字列の $\boxed{\text{式}_2}$ 番目から $\boxed{\text{式}_3}$ 番目までを取り出した文字列を作る。

言葉探し

```
def match(s,p)
```

```
    i=0
```

```
    w=p.length()
```

```
    while submatch(s,i,p,w) < w
```

```
        i = i + 1
```

```
    end
```

```
    i
```

```
end
```

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

p:	a	l	a	i					count
		a	l	a	i				0
			a	l	a	i			3
				a	l	a	i		0
					a	l	a	i	4

```
def submatch (s,i,p,w)
```

```
  j = 0
```

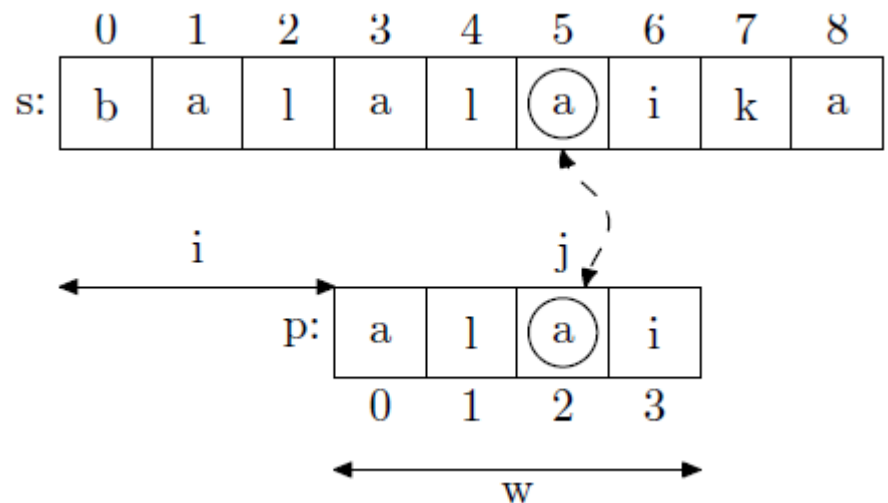
```
  while j < w && s[(i+j)..(i+j)] == p[j..j]
```

```
    j = j + 1
```

```
end
```

```
  j
```

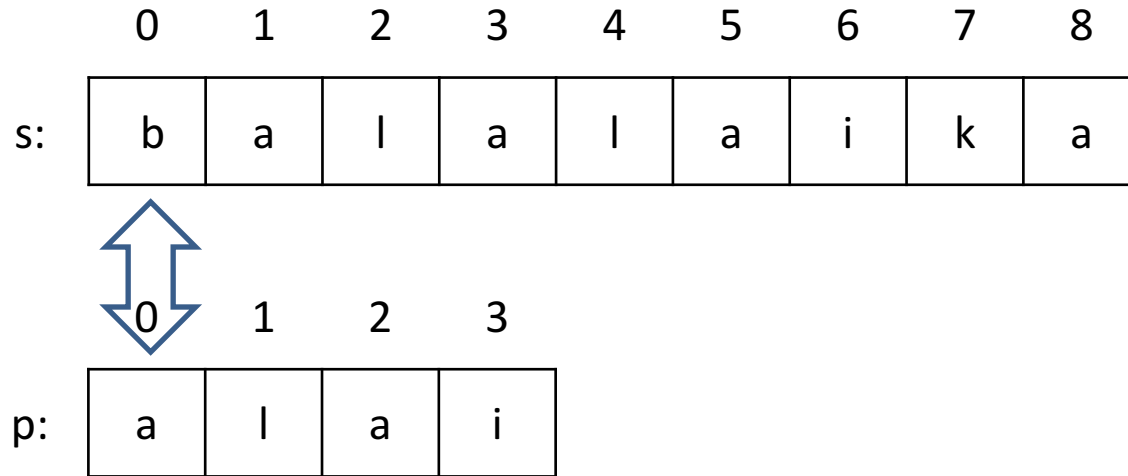
```
end
```



	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, 0, p, 4)

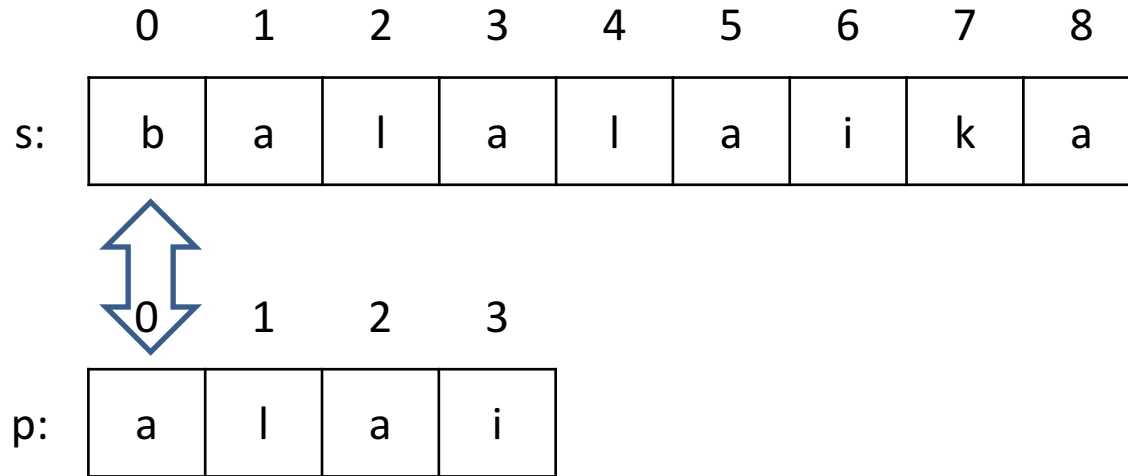


`submatch(s, 0, p, 4)`

`i==0 w==4`

`j==0`

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



submatch(s, 0, p, 4)

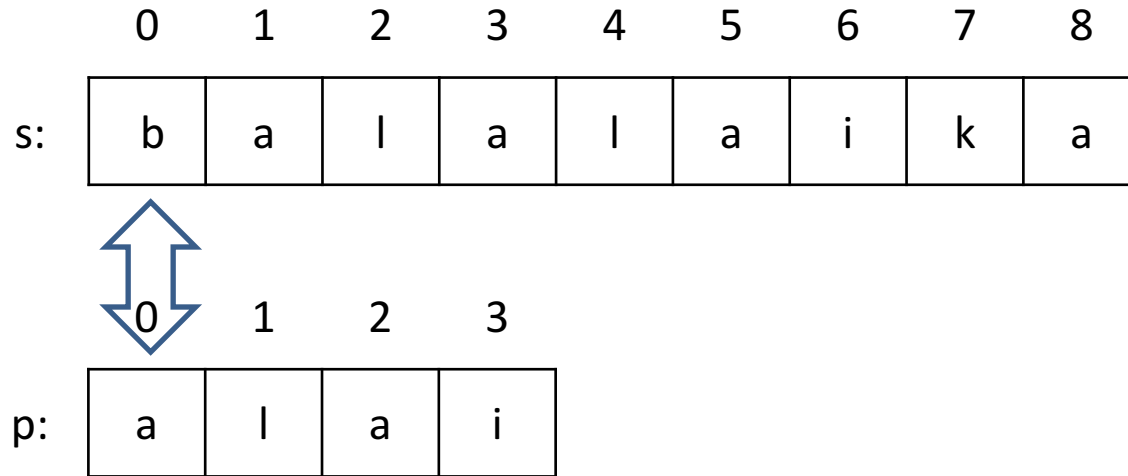
i==0 w==4

j==0

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

$\Rightarrow$ "b"

$\Rightarrow$ "a"



`submatch(s, 0, p, 4)`

`i==0 w==4`

`j==0`

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

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

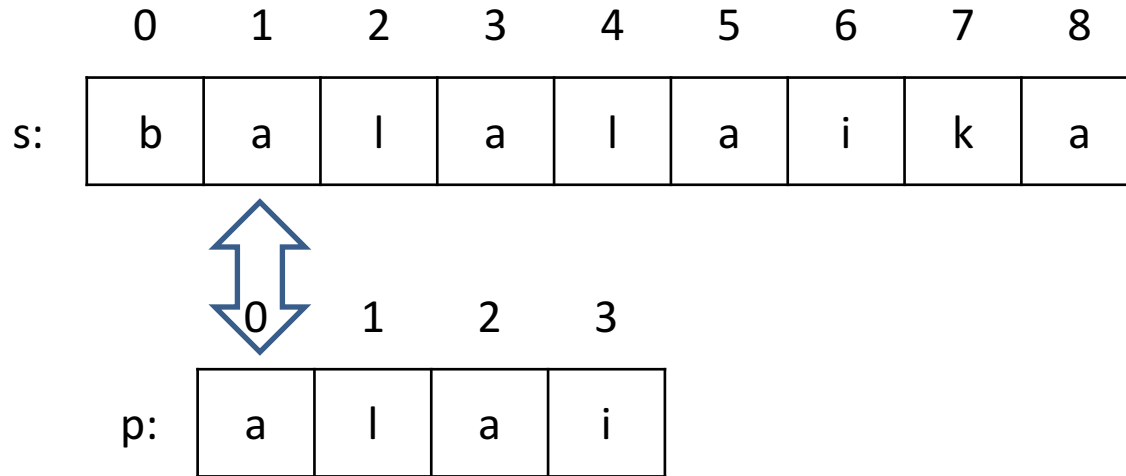
	0	1	2	3
p:	a	l	a	i

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

	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)

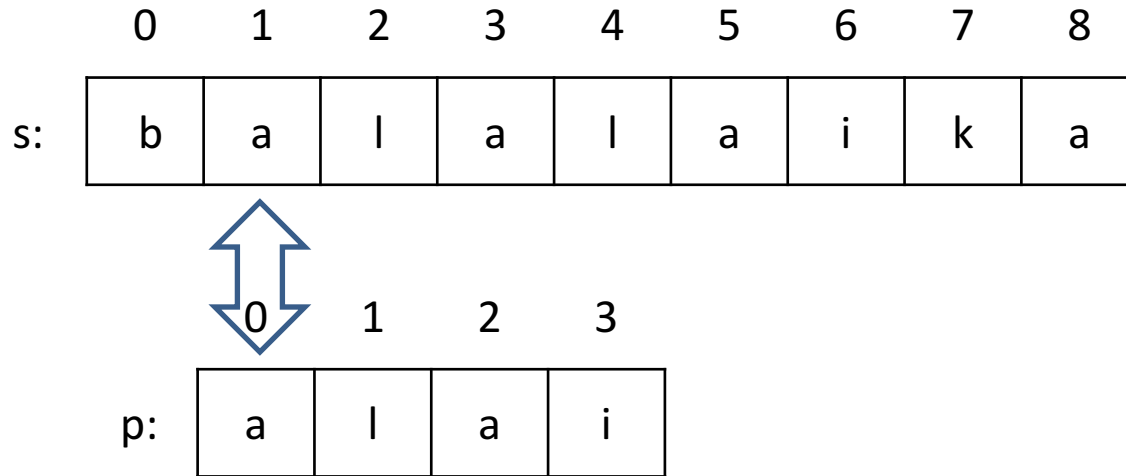


`submatch(s, 1, p, 4)`

`i==1 w==4`

`j==0`

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



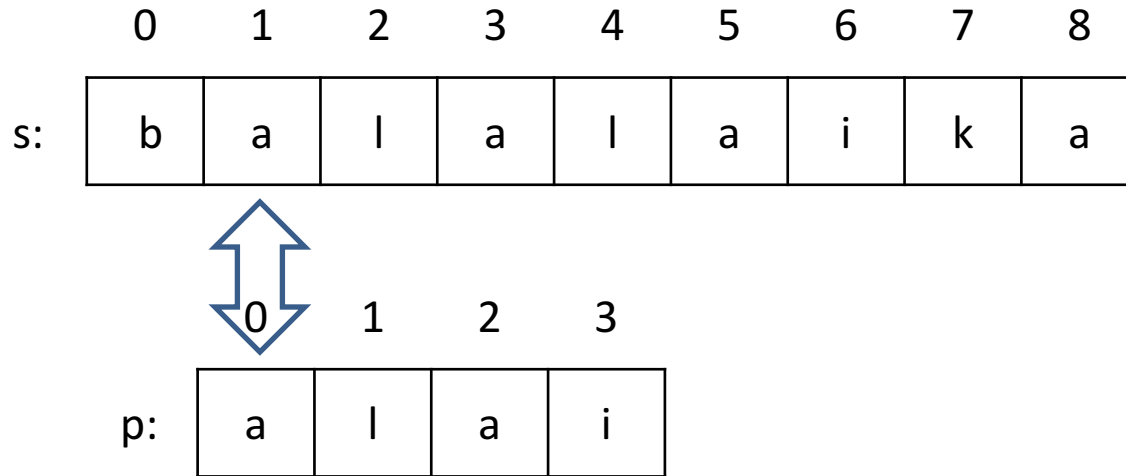
`submatch(s, 1, p, 4)`

`i==1 w==4`

`j==0`

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

`=> "a"` `=> "a"`

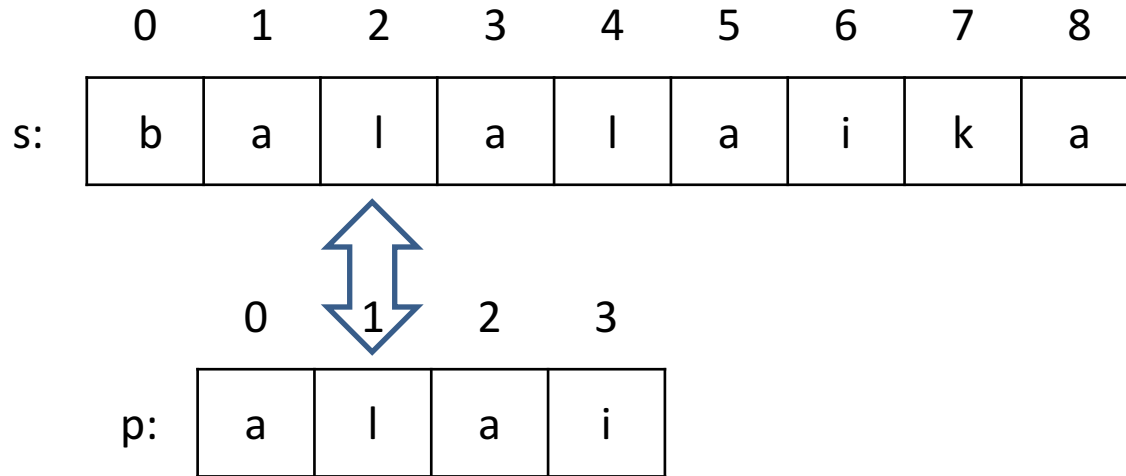


`submatch(s, 1, p, 4)`

`i==1 w==4`

`j==0`

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

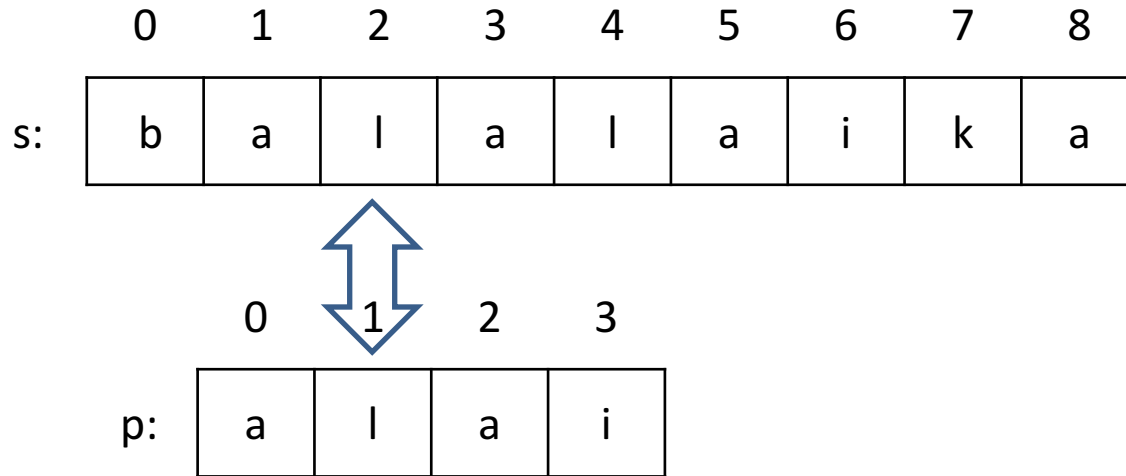


`submatch(s, 1, p, 4)`

`i==1 w==4`

`j==1`

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

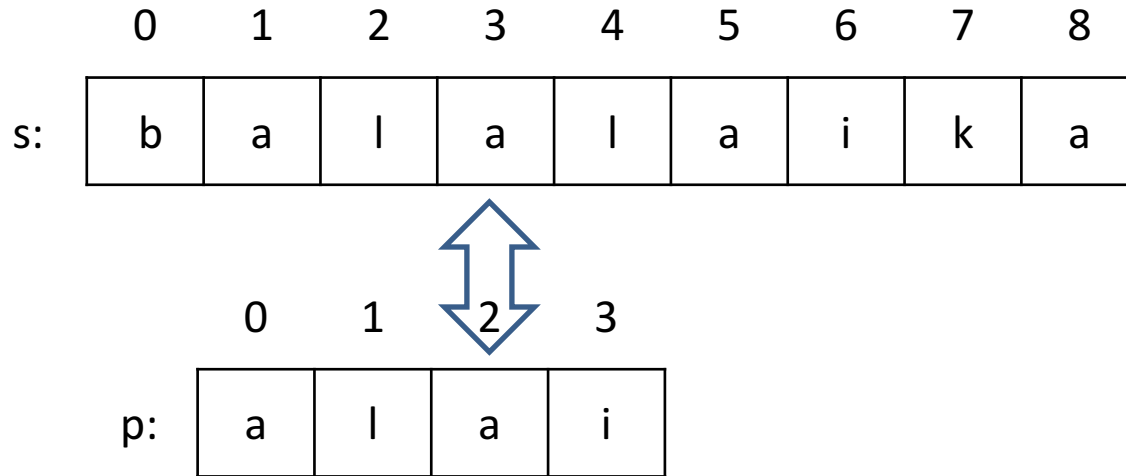


`submatch(s, 1, p, 4)`

`i==1 w==4`

`j==1`

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

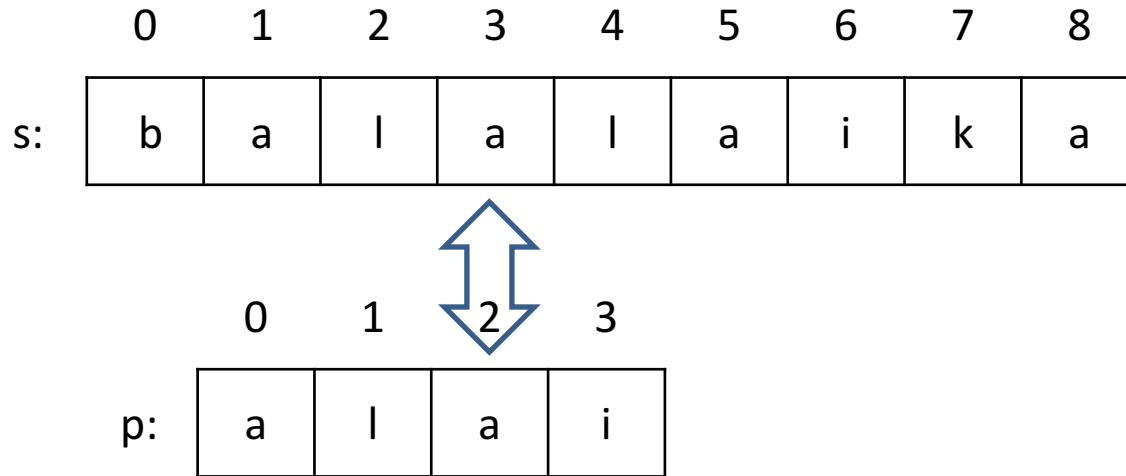


`submatch(s, 1, p, 4)`

`i==1 w==4`

`j==2`

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

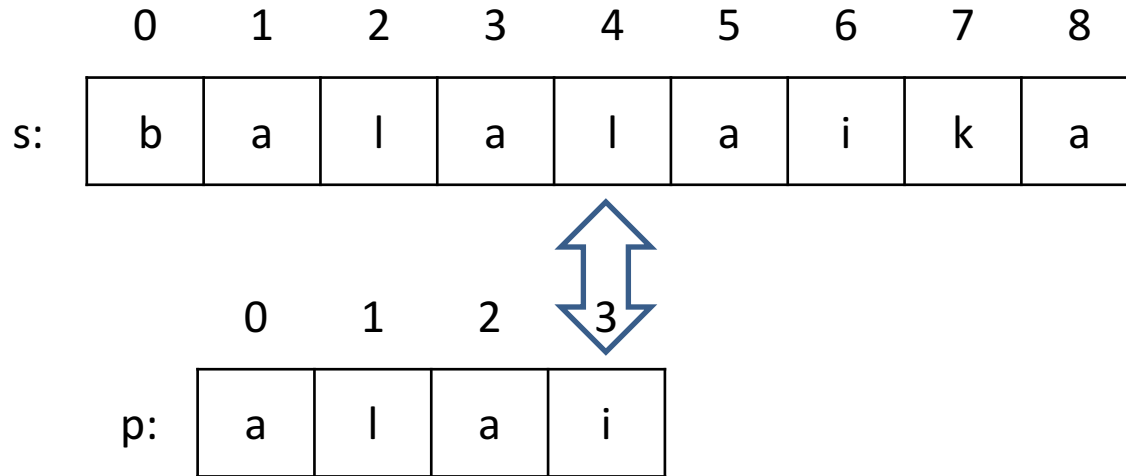


`submatch(s, 1, p, 4)`

`i==1 w==4`

`j==2`

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

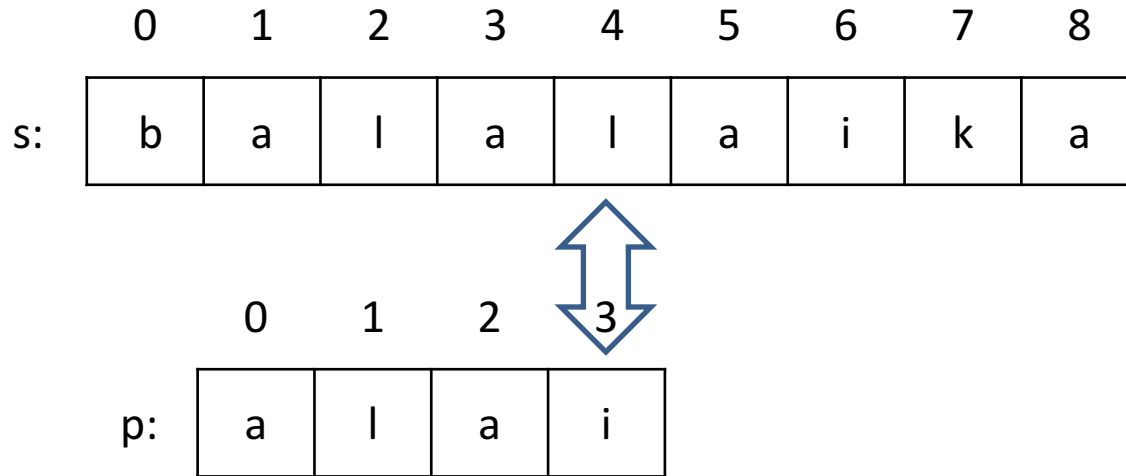


`submatch(s, 1, p, 4)`

`i==1 w==4`

`j==3`

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



`submatch(s, 1, p, 4)`

`i==1 w==4`

`j==3`

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

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

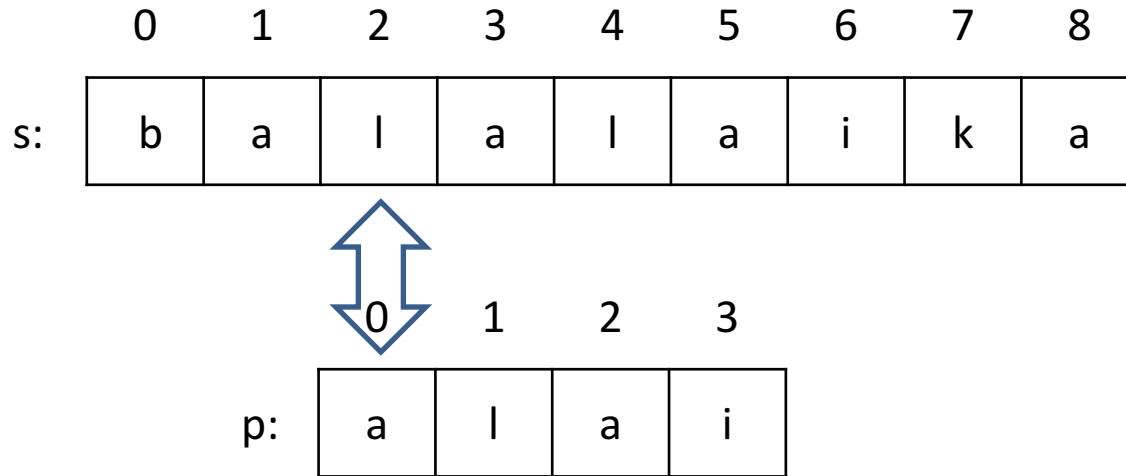
	0	1	2	3
p:	a	l	a	i

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

	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)

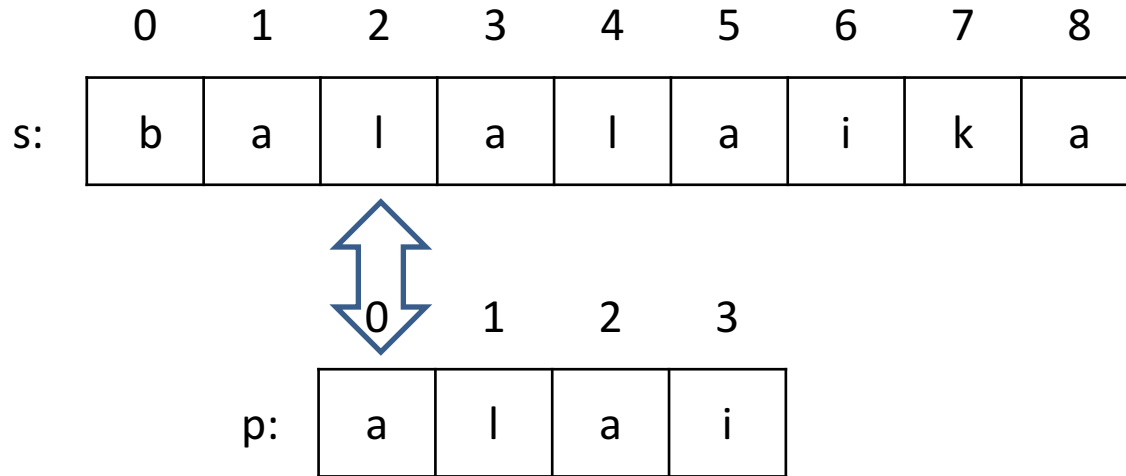


`submatch(s, 2, p, 4)`

`i==2 w==4`

`j==0`

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



`submatch(s, 2, p, 4)`

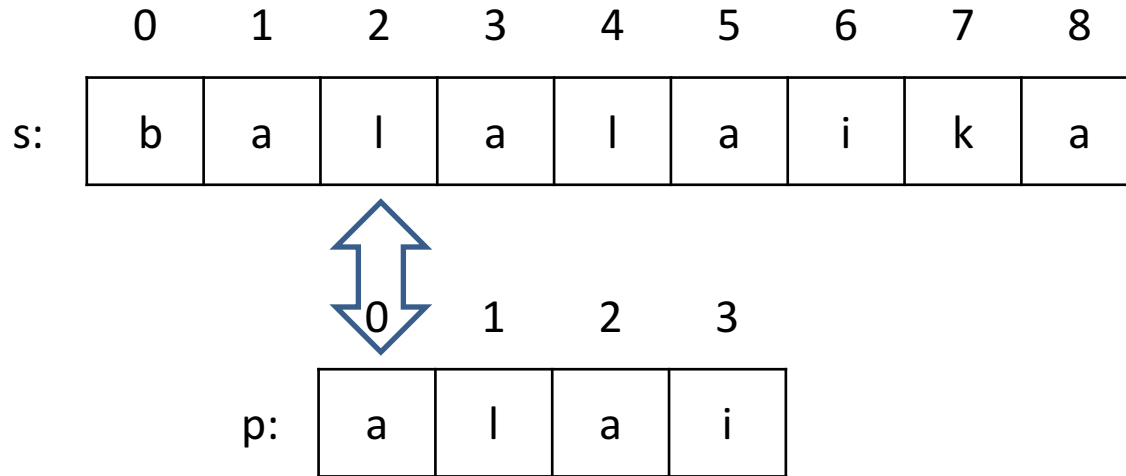
`i==2 w==4`

`j==0`

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

`=> "l"`

`=> "a"`



`submatch(s, 2, p, 4)`

`i==2 w==4`

`j==0`

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

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

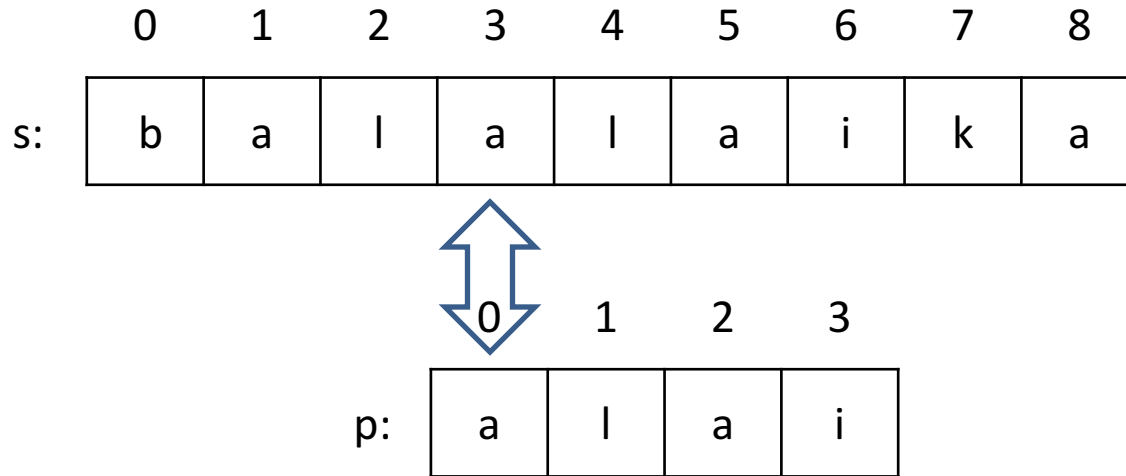
	0	1	2	3
p:	a	l	a	i

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

	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)

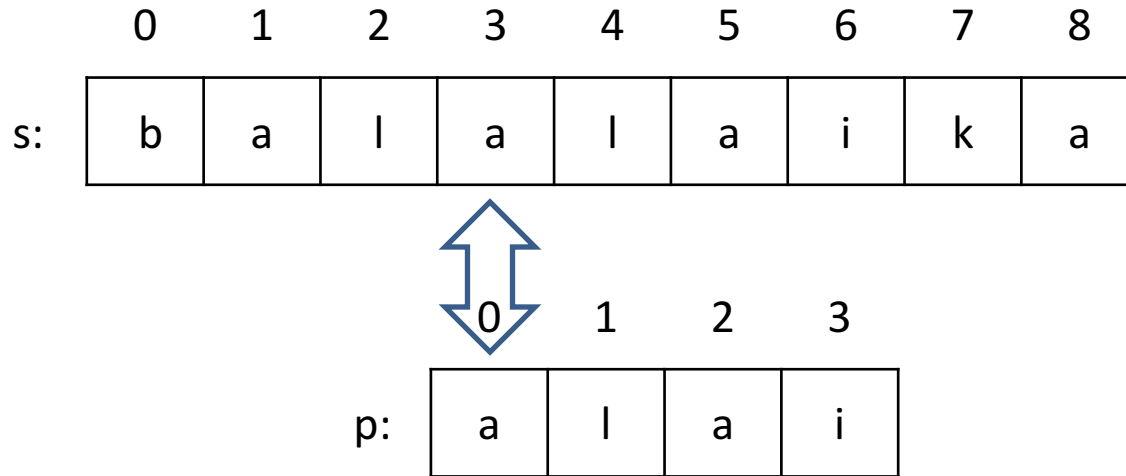


`submatch(s, 3, p, 4)`

`i==3 w==4`

`j==0`

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



`submatch(s, 3, p, 4)`

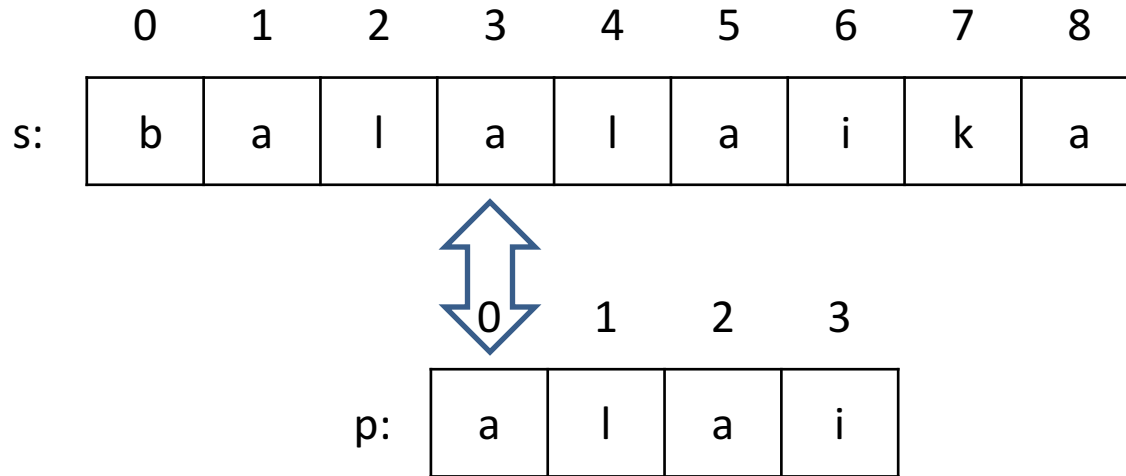
`i==3 w==4`

`j==0`

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

`=> "a"`

`=> "a"`

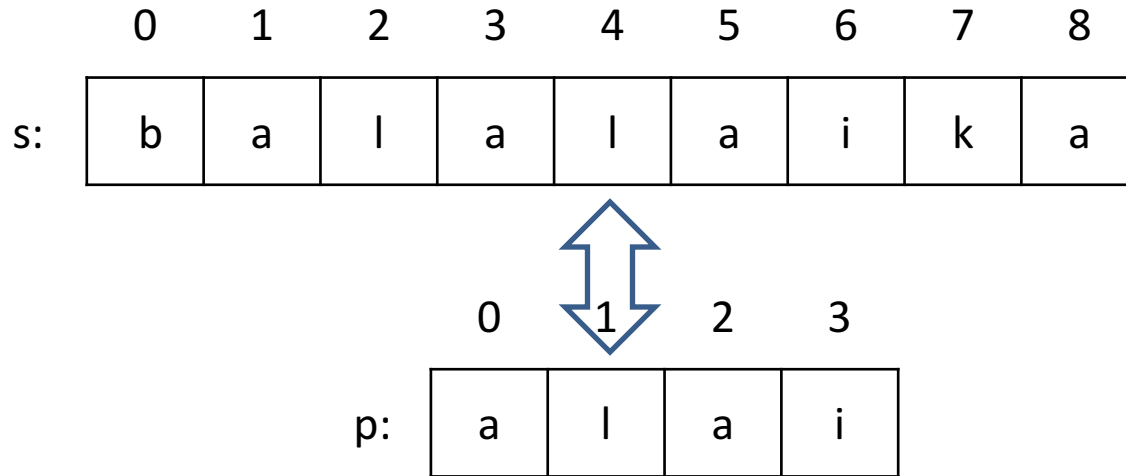


`submatch(s, 3, p, 4)`

`i==3 w==4`

`j==0`

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

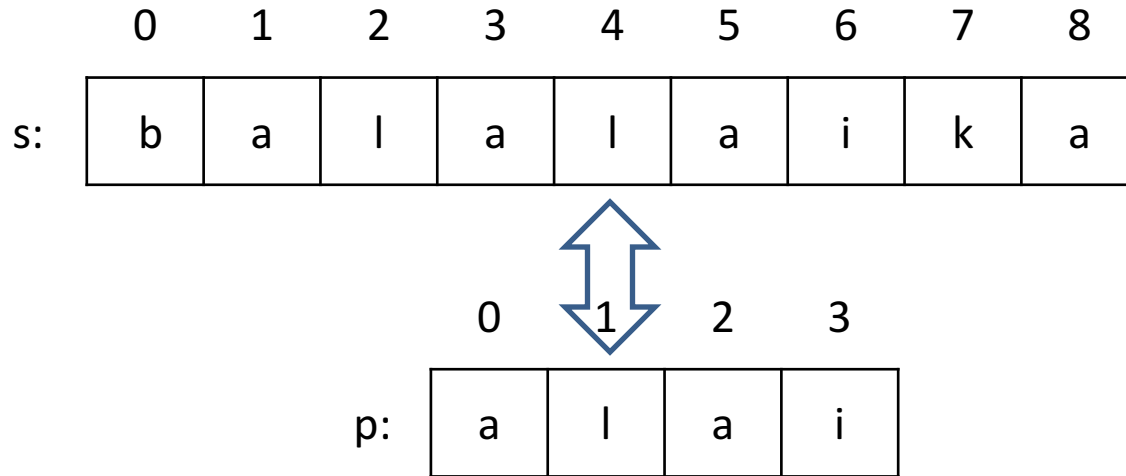


`submatch(s, 3, p, 4)`

`i==3 w==4`

`j==1`

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

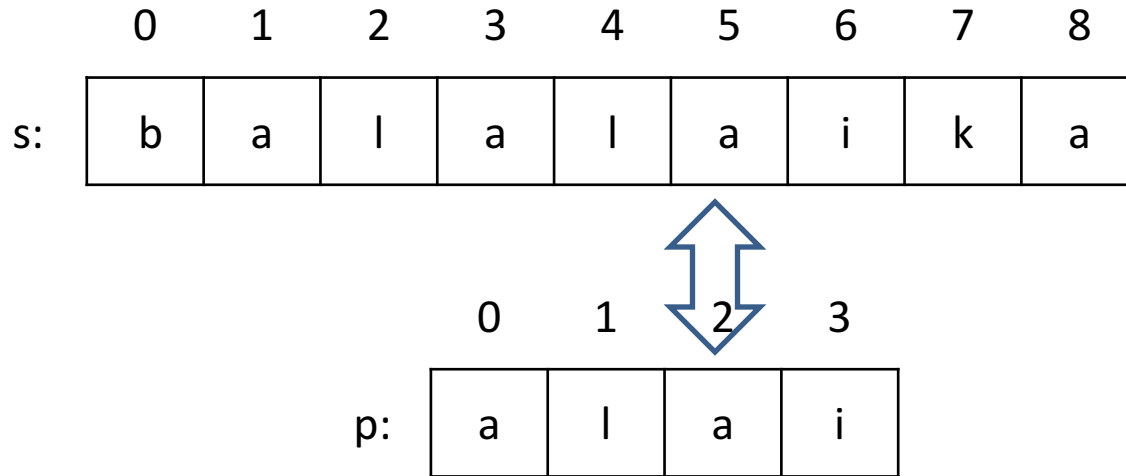


`submatch(s, 3, p, 4)`

`i==3 w==4`

`j==1`

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

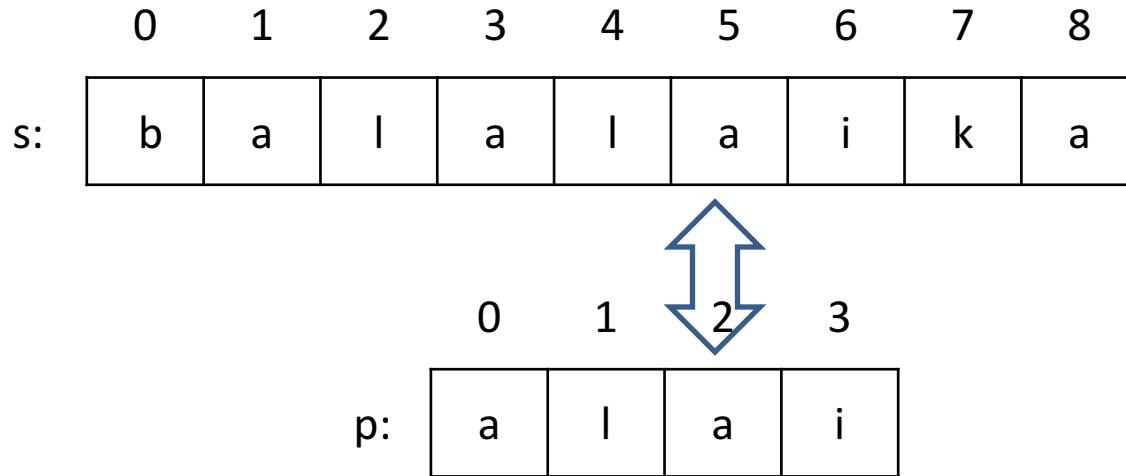


`submatch(s, 3, p, 4)`

`i==3 w==4`

`j==2`

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

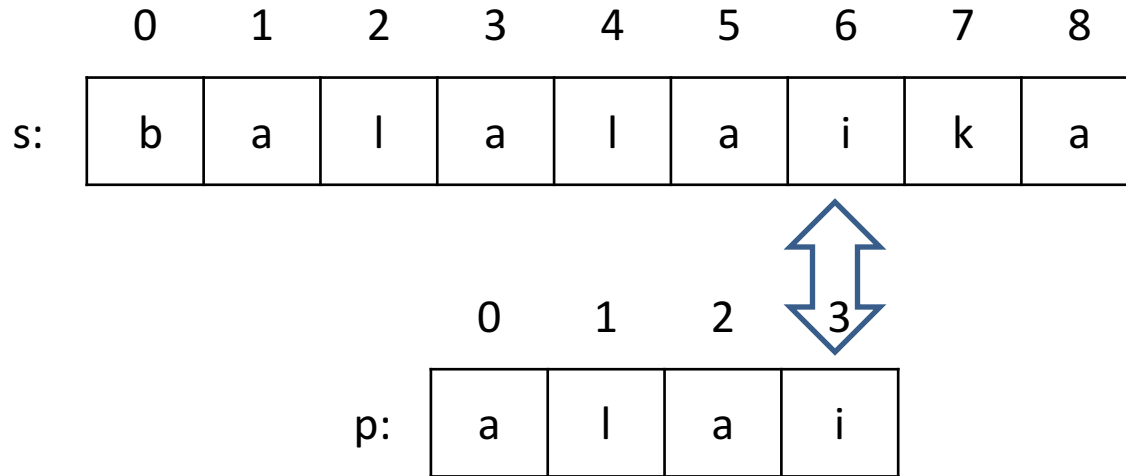


`submatch(s, 3, p, 4)`

`i==3 w==4`

`j==2`

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

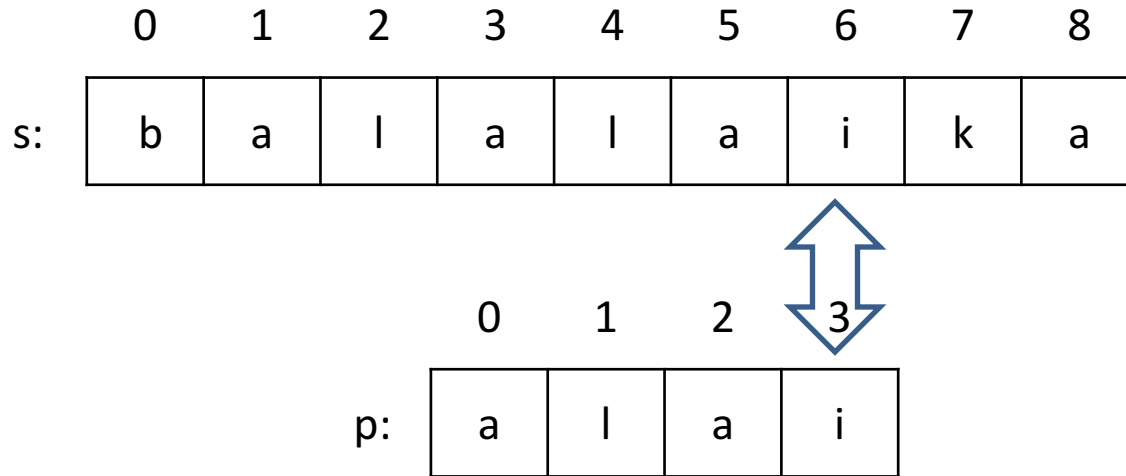


`submatch(s, 3, p, 4)`

`i==3 w==4`

`j==3`

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

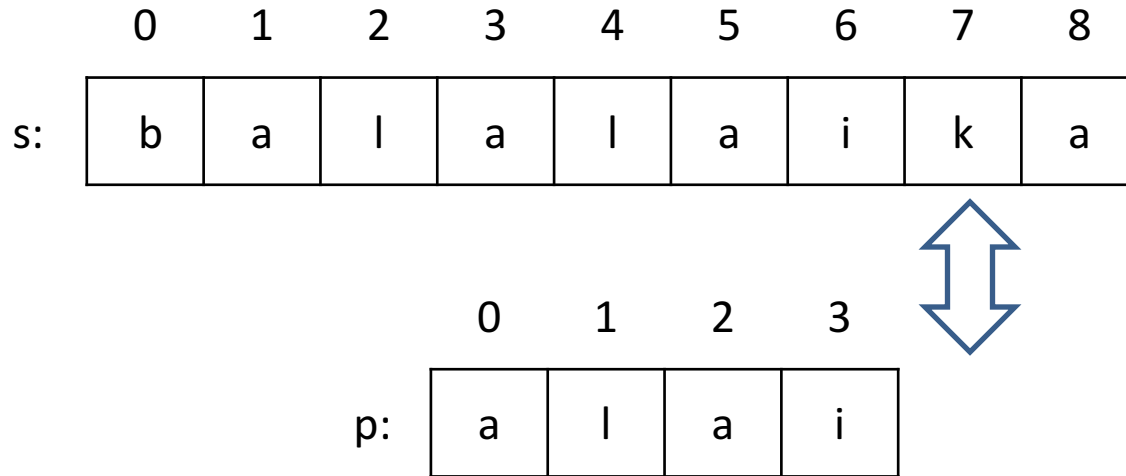


`submatch(s, 3, p, 4)`

`i==3 w==4`

`j==3`

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

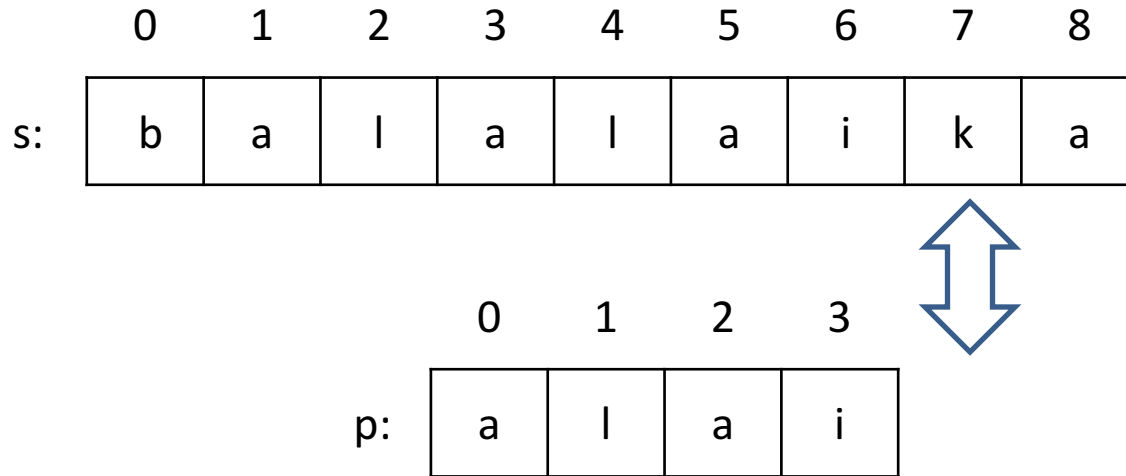


`submatch(s, 3, p, 4)`

`i==3 w==4`

`j==4`

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



`submatch(s, 3, p, 4)`

`i==3 w==4`

`j==4`

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

	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)

i==3 w==4

j==4

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

$j < w$ が false なので
 $s[(i+j)..(i+j)] == p[j..j]$ は
 検査されない

	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) => 4

	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

$\text{submatch}(s, 3, p, 4) \Rightarrow 4$

$i=3$ のときに、 $\text{submatch}(s, i, p, w) < w$ が false になる

言葉探し

```
def match(s,p)
```

```
    i=0
```

```
    w=p.length()
```

```
    while submatch(s,i,p,w) < w
```

```
        i = i + 1
```

```
    end
```

```
    i
```

```
end
```

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

p:	a	l	a	i					count
		a	l	a	i				0
			a	l	a	i			3
				a	l	a	i		0
					a	l	a	i	4

言葉探し

```
irb(main):007:0> match("balalaika", "alai")
```

```
=> 3
```

```
irb(main):008:0> match("hualalai", "alai")
```

```
=> 4
```

```
irb(main):009:0> match("balalaika", "aa")
```

止まらなくなるので Control-C

match.rbを授業のページからダウンロードする

練習

- `match` の定義では `s` 中に `p` が必ず現われることを仮定していた。`p` が現われない場合に `-1` と答える `match_safe(s,p)` を定義せよ。

(ヒント) `match` の `while` の条件が問題。

現われない場合に、`-1` にする方法も必要。

`match_safe(s,p)` ができた人は、教科書65ページの練習 4.5 に挑戦してみてください。配布プログラムは授業のWebページからダウンロードできます。

進捗状況の確認

1. `match_safe(s,p)`ができた(できた時点で投票してください)

`match_safe(s,p)`ができた人は、教科書65ページの練習 4.5 に挑戦してみてください。
配布プログラムは授業のWebページからダウンロードできます。

言葉探し

```
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
```

文字列の反転 – 反復版

```
def reverse_loop(s)
  result = ""
  for i in 0..s.length()-1
    result = s[i..i] + result
  end
  result
end
```

```
irb(main):002:0> reverse_loop("yama")
=> "amay"
```

文字列の反転 – 再帰版

```
def reverse1(s)
  if s.length() == 1
    s
  else
    reverse1(s[1..(s.length()-1)]) + s[0..0]
  end
end
```

```
irb(main):002:0> reverse1("yama")
=> "amay"
```

文字列の反転 – 再帰版

```
def reverse2(s)
  if s.length() == 1
    s
  else
    s[(s.length()-1)..(s.length()-1)] +
    reverse2(s[0..(s.length()-2)])
  end
end
```

```
irb(main):002:0> reverse2("yama")
=> "amay"
```

わからない？

1. わかったので大丈夫
2. reverse_loop
3. reverse1 (再帰)
4. reverse2 (再帰)
5. そのほか 具体的に書く

文字列への変換

```
irb(main):010:0> 123+123
```

```
=> 246
```

```
irb(main):011:0> "123"+"123"
```

```
=> "123123"
```

```
irb(main):012:0> "123"+123
```

```
TypeError: no implicit conversion of Fixnum into String
```

```
from (irb):12:in `+'
```

```
from (irb):12
```

```
from :0
```

文字列への変換

```
irb(main):013:0> "123"+123.to_s()
```

```
=> "123123"
```

```
irb(main):014:0> i = 10
```

```
=> 10
```

```
irb(main):015:0> i.to_s()
```

```
=> "10"
```

```
irb(main):016:0> (i+1).to_s()
```

```
=> "11"
```

2進数文字列への変換 – 基本

- 2で割った余りを逆に並べる

25 $\rightarrow$ “11001”

12 ... 1

6 ... 0

3 ... 0

1 ... 1

0 ... 1

2進数への変換 – 反復版

```
def binary_loop(n)
  s = ""
  while n >= 1
    s = s + (n%2).to_s()
    n = n / 2
  end
  s
end
```

irb(main):007:0> **binary_loop(25)**

=> "10011"

2進数への変換 – 反復版

```
def binary_good(n)
  s = ""
  while n >= 1
    s = (n%2).to_s() + s
    n = n / 2
  end
  s
end
```

irb(main):008:0> **binary_good(25)**

=> "11001"

2進数への変換 – 再帰版

```
def binary(n)
  if n < 2
    n.to_s()
  else
    binary(n/2) + (n%2).to_s()
  end
end
```

irb(main):009:0> **binary(25)**

=> "11001"

わからない？

1. わかったので、大丈夫
2. `binary_loop`
3. `binary_good` (前に付け足していく)
4. `binary` (再帰: 先に前を計算して後に付ける)
5. そのほか 具体的に書く

ハノイの塔



ハノイの塔



ハノイの塔



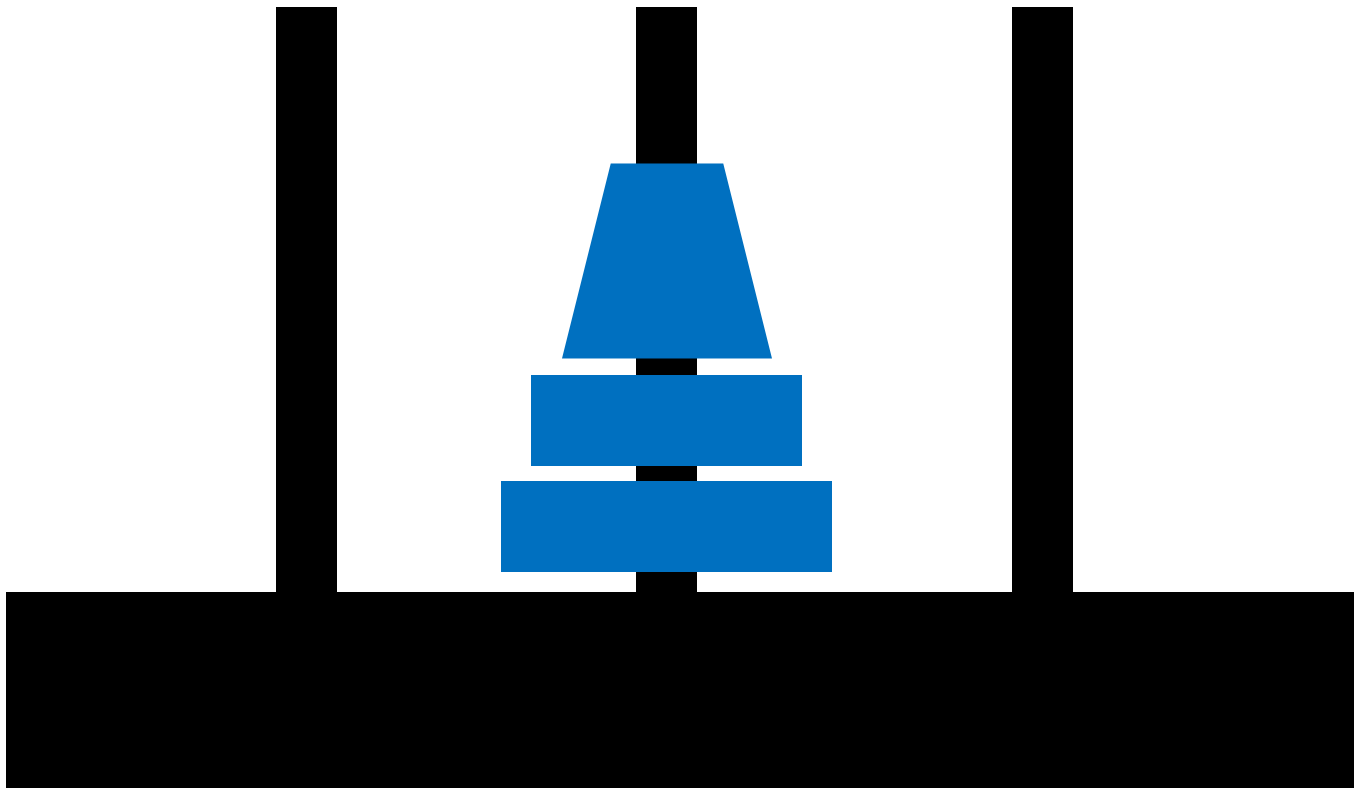
ハノイの塔



ハノイの塔



ハノイの塔



ハノイの塔



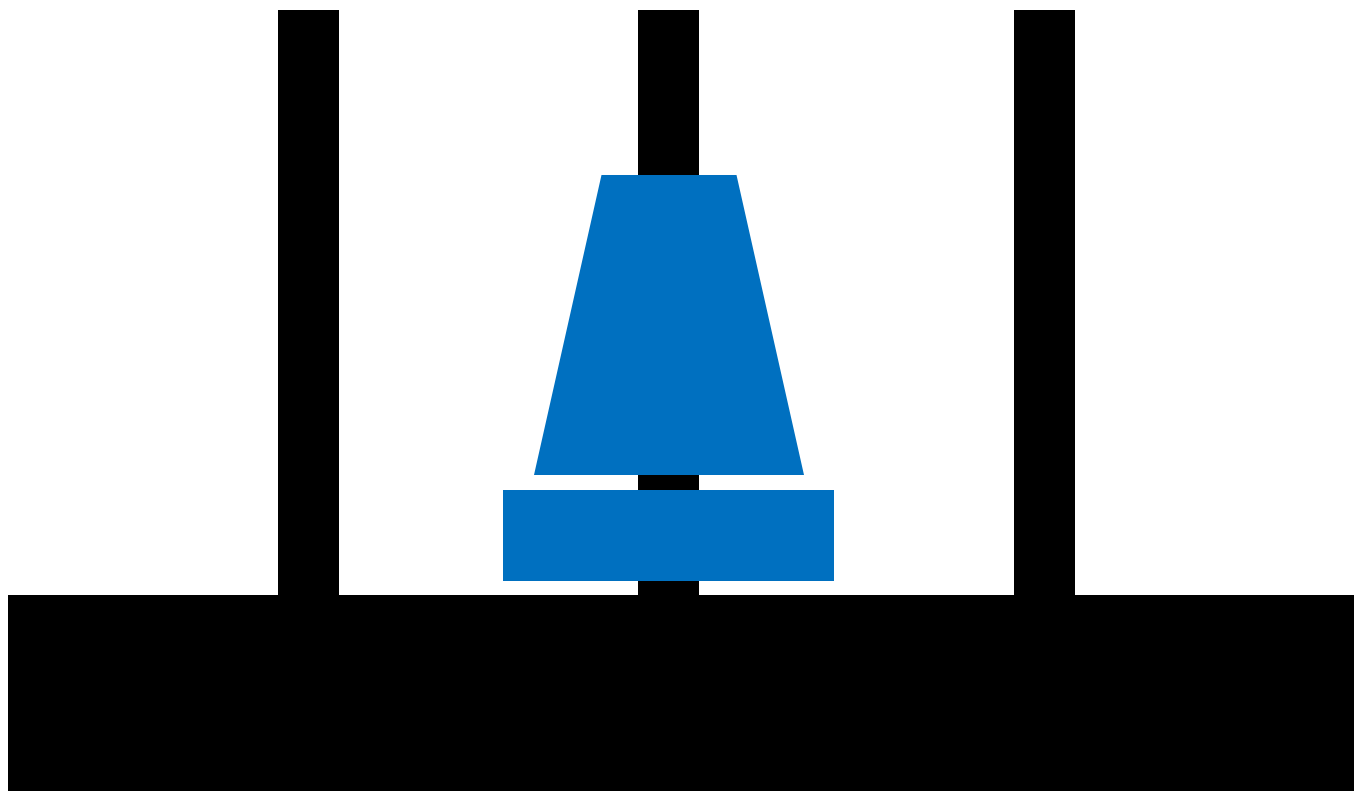
ハノイの塔



ハノイの塔



ハノイの塔



```
def hanoi_times(n)
  if n==0
    0
  else
    hanoi_times(___) + 1 + hanoi_times(___)
  end
end
```

```
def hanoi(n, from, to, buf)
  if n==0
    ""
  else
    hanoi(__, __, __, __) +
    "from " + from + " to " + to + "; " +
    hanoi(__, __, __, __)
  end
end
```

```
irb(main):001:0> load("./hanoi.rb")
```

```
=> true
```

```
irb(main):002:0> hanoi(4, "A", "B", "C")
```

```
⇒ "from A to C; from A to B; from C to B; from A  
  to C; from B to A; from B to C; from A to C;  
  from A to B; from C to B; from C to A; from B  
  to A; from C to B; from A to C; from A to B;  
  from C to B; "
```

```
irb(main):003:0> hanoi_times(4)
```

```
=> 15
```

練習

- ハノイの塔での円盤の移動回数や移動方法を求めるプログラムを作成する。

(ヒント)問題を小さくすることを考える。 n 枚移動の問題を $n-1$ 枚の移動に分解する。

レポート

- 課題

以下の関数を定義した hanoi.rb を提出せよ

- ハノイの塔で円盤 n 枚の場合の移動回数を返す再帰的な関数 `hanoi_times(n)`
- ハノイの塔で, `from` から `to` へ円盤 n 枚を移動する過程を書いた文字列を返す再帰的な関数 `hanoi(n, from, to, buf)`

- 提出先

ITC-LMS

- 〆切

11月13日 授業開始前

ハノイの塔

