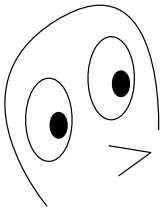
String Search

Martin Kay

Stanford University
and University of the Saarland

Look back

FSA's

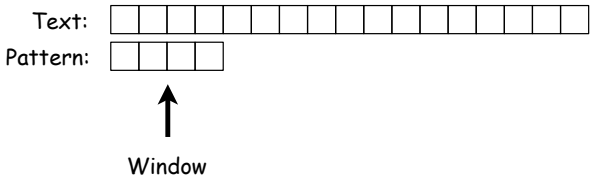


Look forward

Suffix trees

The String-searching Problem

- **Decide whether a text (of t characters) contains a given substring, or pattern (of p characters)**
- **Identify all instances of the pattern in a text.**



Naive Search in Prolog

```
naive_search(Pattern, TextSuffix, 1) :-
  append(Pattern, _, TextSuffix).
naive_search(Pattern, [_ | TextSuffix], N) :-
  naive_search(Pattern, TextSuffix, N0),
  N is N0+1.
```

append?

Naive Search in Ruby

```
def search(pattern, text)
  plen = pattern.length
  tlen = text.length
  (tlen-plen+1).times do |i|
    yield i if text[i, plen]==pattern
  end
end
```

opaque!

Naive Search in Ruby

```
def search1(pattern, text)
  i=-1
  while i
    yield i if i=text.index(pattern, i+1)
  end
end
```

opaque!

Naive Search in Ruby

```
def naive_match(pattern, text, trace=false)
  plen = pattern.length
  tlen = text.length
    for i in 0..tlen-plen+1
      for j in 0..plen-1
        break unless text[i+j]==pattern[j]
        if j==plen-1
         yield i
        end
     end
  end
end
```

Hilaire Belloc: Tarantella

Do you remember an Inn,
Miranda?
Do you remember an Inn?
And the tedding and the spreading
Of the straw for a bedding,
And the fleas that tease in the High Pyrenees,
And the wine that tasted of tar?
And the cheers and the jeers of the young muleteers
(Under the vine of the dark veranda)?
Do you remember an Inn, Miranda?
Do you remember an Inn?
And the cheers and the jeers of the young muleteers
Who hadn't got a penny,
And who weren't paying any,
And the hammer at the doors and the din?
And the hip! hop! hap!
Of the clap
Of the hands to the swirl and the twirl
Of the girl gone chancing,
Glancing,
Dancing,
Backing and advancing,
Snapping of the clapper to the spin
Out and in-
And the ting, tong, tang of the guitar!
Do you remember an Inn,
Miranda?
Do you remember an Inn?

Never more;
Miranda,
Never more.
Only the high peaks hoar;
And Aragon a torrent at the door.
No sound
in the walls of the halls where falls
The tread
Of the feet of the dead to the ground,
No sound:
But the boom
Of the far waterfall like doom.

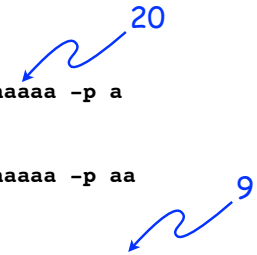
Naive Search

```
% naive_search.rb -f belloc -p z
1025 comparisons, 0 matches

% naive_search.rb -f belloc -p the
1129 comparisons,36 matches

% naive_search.rb -f belloc -p 'Do you remember an Inn'
1131 comparisons,6 matches

% naive_search.rb -f belloc -p 'Do you remember an Inn?'
1136 comparisons,3 matches
```


Naive Search

```
% naive_search.rb -s abababacabadabacabadabacababa -p
'abacabadabacaba'
    2 abacabadabacabadabacababa
   10 abacabadabacababa
50 comparisons, 2 matches
```


Worst Case




```
% naive_search.rb -s aaaaaaaaaaaaaaaaaaaa -p a
20 comparisons, 20 matches

% naive_search.rb -s aaaaaaaaaaaaaaaaaaaa -p aa
38 comparisons ,19 matches

% naive_search.rb -s aaaaaaaaaaaaaaaaaaaa -p aaaaaaaaa
108 comparisons, 12 matches

% naive_search.rb -s aaaaaaaaaaaaaaaaaaaa -p aaaaaaaaaa
110 comparisons, 11 matches

% naive_search.rb -s aaaaaaaaaaaaaaaaaaaa -p aaaaaaaaaaa
110 comparisons, 10 matches

% naive_search.rb -s aaaaaaaaaaaaaaaaaaaa -p
aaaaaaaāaaaa
108 comparisons, 9 matches
```

Naive Search

```
% naive_search.rb -s abababacabadabacabadabacababa -p
'abacabadabacaba'
    2 abacabadabacabadabacababa
   10 abacabadabacababa
50 comparisons, 2 matches
```


On-line text scan

- On-line: paced by some external process, in this case the availability of text characters.
- Advance the current process as much as possible before the next character becomes available.

On-line Search

```
def on_line_search(pattern, text, trace)
  plen = pattern.length
  tlen = text.length
  locs=[0]
  for i in 0..tlen
    new_locs=[0]
    for j in locs
      if i+plen-j <= tlen
        if text[i]==pattern[j]
          if j==plen-1
            yield i-plen+1
          else
            new_locs << j+1
          end
        end
      end
    end
    locs=new_locs
  end
end
```



```
abababacabadabacabadabacababa
abacabadabacaba
 abacabadabacaba
```

```
ababacabadabacabadabacababa
abacabadabacaba
 ✗bacabadabacaba
```



```
ababacabadabacabadabacababa
  abacabadabacaba
    acabadabacaba
```



```
ababacabadabacabadabacababa
aba✗abadabacaba
  abacabadabacaba
   ✗bacabadabacaba
```



```
ababacabadabacabadabacababa
  abacabadabacaba
    a✗acabadabacaba
     ✗bacabadabacaba
```

```
ababacabadabacabadabacababa
  abacabadabacaba            (7)
      abacabadabacaba        (3)
        abacabadabacaba      (1)
```


Naive Search both ways

```
% naive_search.rb -s abababacabadabacabadabacababa -p
'abacabadabacaba'
    2 abacabadabacabadabacababa
   10 abacabadabacababa
50 comparisons, 2 matches

% on_line_search.rb -s abababacabadabacabadabacababa -p
'abacabadabacaba'
    2 abacabadabacabadabacababa
   10 abacabadabacababa
50 comparisons, 2 matches
```



```
% monotonic_search.rb -s abababacabadabacabadabacababa -p
   'abacabadabacaba'
[0]
[0, 1]
[0, 2]
[0, 1, 3]
[0, 2]
[0, 1, 3]
[0, 4]
[0, 1, 5]
[0, 2, 6]
[0, 1, 3, 7]
[0, 8]
[0, 1, 9]
[0, 2, 10]
[0, 1, 3, 11]
```

```
[0, 4, 12]
[0, 1, 5, 13]
[0, 2, 6, 14]
   2 abacabadabacabadabacababa
[0, 1, 3, 7]
[0, 8]
[0, 1, 9]
[0, 2, 10]
[0, 1, 3, 11]
[0, 4, 12]
[0, 1, 5, 13]
[0, 2, 6, 14]
  10 abacabadabacababa
[0, 1, 3, 7]
[0, 2]
81 comparisons, 2 matches
```

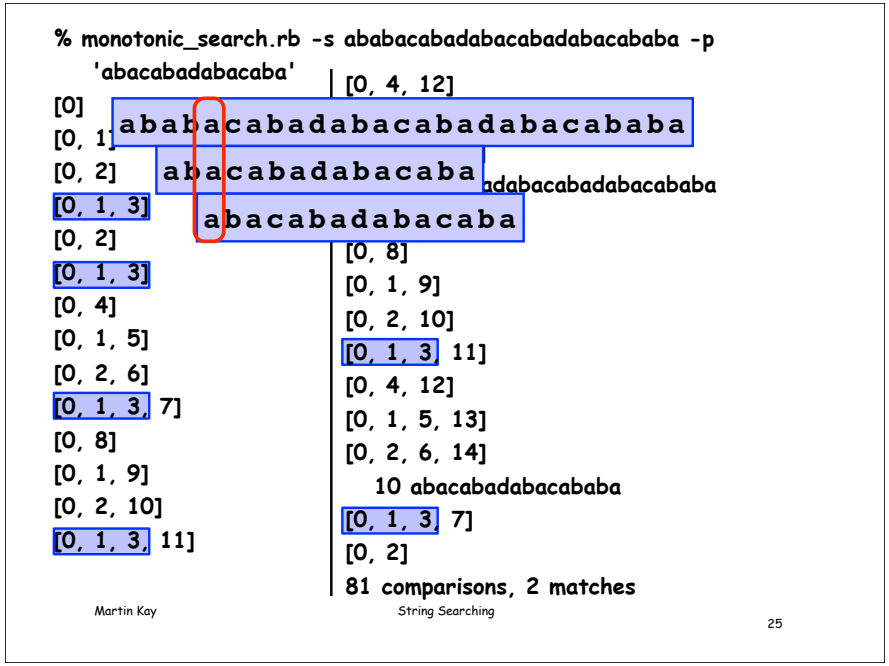
Numbers are positions in the pattern

A number in a given position always has the same sequence numbers to its left


```
% monotonic_search.rb -s abababacabadabacabadabacababa -p
   'abacabadabacaba'
[0]
[0, 1]
[0, 2]
[0, 1, 3]
[0, 2]
[0, 1, 3]
[0, 4]
[0, 1, 5]
[0, 2, 6]
[0, 1, 3, 7]
[0, 8]
[0, 1, 9]
[0, 2, 10]
[0, 1, 3, 11]
```

```
ababacabadabacabadabacababa
  abacabadabacaba
    abacabadabacaba
```

```
[0, 4, 12]
[0, 8]
[0, 1, 9]
[0, 2, 10]
[0, 1, 3, 11]
[0, 4, 12]
[0, 1, 5, 13]
[0, 2, 6, 14]
  10 abacabadabacababa
[0, 1, 3, 7]
[0, 2]
81 comparisons, 2 matches
```


% monotonic_search.rb -s abababacabadabacabadabacababa -p 'abacabadabacaba'
[0]
[0, 1]
[0, 2]
[0, 1, 3]
[0, 2]
[0, 1, 3]
[0, 4]
[0, 1, 5]
[0, 2, 6]
[0, 1, 3, 7]
[0, 8]
[0, 1, 9]
[0, 2, 10]
[0, 1, 3, 11]
ababacabadabacabadabacababa
abacabadabacaba
abacabadabacaba
[0, 4, 12]
[0, 8]
[0, 1, 9]
[0, 2, 10]
[0, 1, 3, 11]
[0, 4, 12]
[0, 1, 5, 13]
[0, 2, 6, 14]
10 abacabadabacababa
[0, 1, 3, 7]
[0, 2]
81 comparisons, 2 matches
Martin Kay
String Searching
25

% monotonic_search.rb -s abababacabadabacabadabacababa -p 'abacabadabacaba'
[0]
[0, 1]
[0, 2]
[0, 1, 3]
[0, 2]
[0, 1, 3]
[0, 4]
[0, 1, 5]
[0, 2, 6]
[0, 1, 3, 7]
[0, 8]
[0, 1, 9]
[0, 2, 10]
[0, 1, 3, 11]
ababacabadabacabadabacababa
abacabadabacaba
abacabadabacaba
[0, 4, 12]
[0, 2, 10]
[0, 1, 3, 11]
[0, 4, 12]
[0, 1, 5, 13]
[0, 2, 6, 14]
10 abacabadabacababa
[0, 1, 3, 7]
[0, 2]
81 comparisons, 2 matches
Martin Kay
String Searching
26

% monotonic_search.rb -s abababacabadabacabadabacababa -p 'abacabadabacaba'
[0]
[0, 1]
[0, 2]
[0, 1, 3]
[0, 2]
[0, 1, 3]
[0, 4]
[0, 1, 5]
[0, 2, 6]
[0, 1, 3, 7]
[0, 8]
[0, 1, 9]
[0, 2, 10]
[0, 1, 3, 11]
ababacabadabacabadabacababa
abacabadabacaba
abacabadabacaba
[0, 2, 10]
[0, 1, 3, 11]
[0, 4, 12]
[0, 1, 5, 13]
[0, 2, 6, 14]
10 abacabadabacababa
[0, 1, 3, 7]
[0, 2]
81 comparisons, 2 matches
Martin Kay
String Searching
27

% monotonic_search.rb -s abababacabadabacabadabacababa -p 'abacabadabacaba'
[0]
[0, 1]
[0, 2]
[0, 1, 3]
[0, 2]
[0, 1, 3]
[0, 4]
[0, 1, 5]
[0, 2, 6]
[0, 1, 3, 7]
[0, 8]
[0, 1, 9]
[0, 2, 10]
[0, 1, 3, 11]
ababacabadabacabadabacababa
abacabadabacaba
abacabadabacaba
[0, 2, 10]
[0, 1, 3, 11]
[0, 4, 12]
[0, 1, 5, 13]
[0, 2, 6, 14]
10 abacabadabacababa
[0, 1, 3, 7]
[0, 2]
81 comparisons, 2 matches
Martin Kay
String Searching
28

```
% monotonic_search.rb -s ababacabadabacabadabacababa -p
    'abacabadabacaba'
[0]                                 [0, 4, 12]
[0, 1]                              [0, 1, 5, 13]
[0, 2]                              [0, 2, 6, 14]
[0, 1, 3]                             2 abacabadabacabadabacababa
[0, 2]                              [0, 1, 3, 7]
[0, 1, 3]                           [0, 8]
[0, 4]                              [0, 1, 9]
[0, 1, 5]                           [0, 2, 10]
[0, 2, 6]                           [0, 1, 3, 11]
[0, 1, 3, 7]                        [0, 4, 12]
[0, 8]                              [0, 1, 5, 13]
[0, 1, 9]                           [0, 2, 6, 14]
[0, 2, 10]                            10 abacabadabacababa
[0, 1, 3, 11]                       [0, 1, 3, 7]
                                    [0, 2]
                                    81 comparisons, 2 matches
```



```
% monotonic_search.rb -s ababacabadabacabadabacababa -p
    'abacabadabacaba'
[0]                                 [0, 4, 12]
[0, 1]                              [0, 1, 5, 13]
[0, 2]                              [0, 2, 6, 14]
[0, 1, 3]                             2 abacabadabacabadabacababa
[0, 2                               [0, 1, 3, 7]
[0, 1, 3                            ababacabadabacabadabacababa
[0, 4]                               abacabadabacaba
[0, 1, 5]                                abacabadabacaba
[0, 2, 6]                           [0, 4 abacabadabacaba
[0, 1, 3, 7]                        [0, 1, 5, 13]
[0, 8]                              [0, 2, 6, 14]
[0, 1, 9]                             10 abacabadabacababa
[0, 2, 10]                          [0, 1, 3, 7]
[0, 1, 3, 11]                       [0, 2]
                                    81 comparisons, 2 matches
```



```
% monotonic_search.rb -s ababacabadabacabadabacababa -p
    'abacabadabacaba'
[0]                                 [0, 4, 12]
[0, 1]                              [0, 1, 5, 13]
[0, 2]                              [0, 2, 6, 14]
[0, 1, 3]                             2 abacabadabacabadabacababa
[0, 2                               [0, 1, 3, 7]
[0, 1, 3                            ababacabadabacabadabacababa
[0, 4]                               abacabadabacaba
[0, 1, 5]                                abacabadabacaba
[0, 2, 6]                           [0, 4 abacabadabacaba
[0, 1, 3, 7]                        [0, 1, 5, 13]
[0, 8]                              [0, 2, 6, 14]
[0, 1, 9]                             10 abacabadabacababa
[0, 2, 10]                          [0, 1, 3, 7]
[0, 1, 3, 11]                       [0, 2]
                                    81 comparisons, 2 matches
```



```
% monotonic_search.rb -s ababacabadabacabadabacababa -p
    'abacabadabacaba'
[0]                                 [0, 4, 12]
[0, 1]                              [0, 1, 5, 13]
[0, 2]                              [0, 2, 6, 14]
[0, 1, 3]                             2 abacabadabacabadabacababa
[0, 2                               [0, 1, 3, 7]
[0, 1, 3                            ababacabadabacabadabacababa
[0, 4]                               abacabadabacaba
[0, 1, 5]                                abacabadabacaba
[0, 2, 6]                           [0, 4 abacabadabacaba
[0, 1, 3, 7]                        [0, 1, 5, 13]
[0, 8]                              [0, 2, 6, 14]
[0, 1, 9]                             10 abacabadabacababa
[0, 2, 10]                          [0, 1, 3, 7]
[0, 1, 3, 11]                       [0, 2]
                                    81 comparisons, 2 matches
```

```
% monotonic_search.rb -s abababacabadabacabadabacababa -p
    'abacabadabacaba'
[0]
[0, 1]
[0, 2]
[0, 1, 3]
[0, 2]
[0, 1, 3]
[0, 4]
[0, 1, 5]
[0, 2, 6]
[0, 1, 3, 7]
[0, 8]
[0, 1, 9]
[0, 2, 10]
[0, 1, 3, 11]
[0, 4, 12]
[0, 1, 5, 13]
[0, 2, 6, 14]
   2 abacabadabacabadabacababa
[0, 1, 3, 7]
[0, 8]
[0, 1, 9]
[0, 2, 10]
[0, 1, 3, 11]
[0, 4, 12]
[0, 1, 5, 13]
[0, 2, 6, 14]
  10 abacabadabacababa
[0, 1, 3, 7]
[0, 2]
81 comparisons, 2 matches
```

If this matches

then so have these!


```
% monotonic_search.rb -s abababacabadabacabadabacababa -p
    'abacabadabacaba'
[0]
[0, 1]
[0, 2]
[0, 1, 3]
[0, 2]
[0, 1, 3]
[0, 4]
[0, 1, 5]
[0, 2, 6]
[0, 1, 3, 7]
[0, 8]
[0, 1, 9]
[0, 2, 10]
[0, 1, 3, 11]
[0, 4, 12]
[0, 1, 5, 13]
[0, 2, 6, 14]
   2 abacabadabacabadabacababa
[0, 1, 3, 7]
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	−1	0	0	1	0	1	2	3	0	1	2	3	4	5	6	7

```
[0, 2, 10]
[0, 1, 3, 11]
[0, 4, 12]
[0, 1, 5, 13]
[0, 2, 6, 14]
  10 abacabadabacababa
[0, 1, 3, 7]
[0, 2]
81 comparisons, 2 matches
```



```
% monotonic_search.rb -s abababacabadabacabadabacababa -p
    'abacabadabacaba'
[0]
[0, 1]
[0, 2]
[0, 1, 3]
[0, 2]
[0, 1, 3]
[0, 4]
[0, 1, 5]
[0, 2, 6]
[0, 1, 3, 7]
[0, 8]
[0, 1, 9]
[0, 2, 10]
[0, 1, 3, 11]
[0, 4, 12]
[0, 1, 5, 13]
[0, 2, 6, 14]
   2 abacabadabacabadabacababa
[0, 1, 3, 7]
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	−1	0	0	1	0	1	2	3	0	1	2	3	4	5	6	7

```
[0, 2, 10]
[0, 1, 3, 11]
[0, 4, 12]
[0, 1, 5, 13]
[0, 2, 6, 14]
  10 abacabadabacababa
[0, 1, 3, 7]
[0, 2]
81 comparisons, 2 matches
```



```
% monotonic_search.rb -s abababacabadabacabadabacababa -p
    'abacabadabacaba'
[0]
[0, 1]
[0, 2]
[0, 1, 3]
[0, 2]
[0, 1, 3]
[0, 4]
[0, 1, 5]
[0, 2, 6]
[0, 1, 3, 7]
[0, 8]
[0, 1, 9]
[0, 2, 10]
[0, 1, 3, 11]
[0, 4, 12]
[0, 1, 5, 13]
[0, 2, 6, 14]
   2 abacabadabacabadabacababa
[0, 1, 3, 7]
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	−1	0	0	1	0	1	2	3	0	1	2	3	4	5	6	7

```
[0, 2, 10]
[0, 1, 3, 11]
[0, 4, 12]
[0, 1, 5, 13]
[0, 2, 6, 14]
  10 abacabadabacababa
[0, 1, 3, 7]
[0, 2]
81 comparisons, 2 matches
```

```
% monotonic_search.rb -s ababacabadabacabadabacababa -p
    'abacabadabacaba'
[0]
[0, 1]
[0, 2]
[0, 1, 3]
[0, 2]
[0, 1, 3]
[0, 4]
[0, 1, 5]
[0, 2, 6]
[0, 1, 3, 7]
[0, 8]
[0, 1, 9]
[0, 2, 10]
[0, 1, 3, 11]
[0, 4, 12]
[0, 1, 5, 13]
[0, 2, 6, 14]
   2 abacabadabacabadabacababa
[0, 1, 3, 7]
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
−1	0	0	1	0	1	2	3	0	1	2	3	4	5	6	7

```
[0, 2, 10]
[0, 1, 3, 11]
[0, 4, 12]
[0, 1, 5, 13]
[0, 2, 6, 14]
   10 abacabadabacababa
[0, 1, 3, 7]
[0, 2]
81 comparisons, 2 matches
```



```
% monotonic_search.rb -s ababacabadabacabadabacababa -p
    'abacabadabacaba'
[0]
[0, 1]
[0, 2]
[0, 1, 3]
[0, 2]
[0, 1, 3]
[0, 4]
[0, 1, 5]
[0, 2, 6]
[0, 1, 3, 7]
[0, 8]
[0, 1, 9]
[0, 2, 10]
[0, 1, 3, 11]
[0, 4, 12]
[0, 1, 5, 13]
[0, 2, 6, 14]
   2 abacabadabacabadabacababa
[0, 1, 3, 7]
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
−1	0	0	1	0	1	2	3	0	1	2	3	4	5	6	7

```
[0, 2, 10]
[0, 1, 3, 11]
[0, 4, 12]
[0, 1, 5, 13]
[0, 2, 6, 14]
   10 abacabadabacababa
[0, 1, 3, 7]
[0, 2]
81 comparisons, 2 matches
```



```
% monotonic_search.rb -s ababacabadabacabadabacababa -p
    'abacabadabacaba'
[0]
[0, 1]
[0, 2]
[0, 1, 3]
[0, 2]
[0, 1, 3]
[0, 4]
[0, 1, 5]
[0, 2, 6]
[0, 1, 3, 7]
[0, 8]
[0, 1, 9]
[0, 2, 10]
[0, 1, 3, 11]
[0, 4, 12]
[0, 1, 5, 13]
[0, 2, 6, 14]
   2 abacabadabacabadabacababa
[0, 1, 3, 7]
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
−1	0	0	1	0	1	2	3	0	1	2	3	4	5	6	7

```
[0, 2, 10]
[0, 1, 3, 11]
[0, 4, 12]
[0, 1, 5, 13]
[0, 2, 6, 14]
   10 abacabadabacababa
[0, 1, 3, 7]
[0, 2]
81 comparisons, 2 matches
```

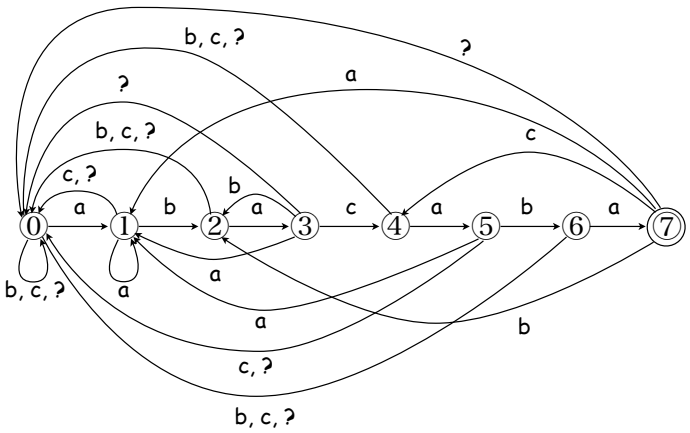


```
% monotonic_search.rb -s ababacabadabacabadabacababa -p
    'abacabadabacaba'
[0]
[0, 1]
[0, 2]
[0, 1, 3]
[0, 2]
[0, 1, 3]
[0, 4]
[0, 1, 5]
[0, 2, 6]
[0, 1, 3, 7]
[0, 8]
[0, 1, 9]
[0, 2, 10]
[0, 1, 3, 11]
[0, 4, 12]
[0, 1, 5, 13]
[0, 2, 6, 14]
   2 abacabadabacabadabacababa
[0, 1, 3, 7]
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
−1	0	0	1	0	1	2	3	0	1	2	3	4	5	6	7

```
[0, 2, 10]
[0, 1, 3, 11]
[0, 4, 12]
[0, 1, 5, 13]
[0, 2, 6, 14]
   10 abacabadabacababa
[0, 1, 3, 7]
[0, 2]
81 comparisons, 2 matches
```

Σ = {a, b, c}




```
% search -a fsm -p abacaba -s abacaba
 0. Transitions = {"a"=>1, "b"=>0, "c"=>0, "?"=>0}
 1. Transitions = {"b"=>2, "a"=>1, "c"=>0, "?"=>0}
 2. Transitions = {"a"=>3, "b"=>0, "c"=>0, "?"=>0}
 3. Transitions = {"c"=>4, "a"=>1, "b"=>2, "?"=>0}
 4. Transitions = {"a"=>5, "b"=>0, "c"=>0, "?"=>0}
 5. Transitions = {"b"=>6, "a"=>1, "c"=>0, "?"=>0}
 6. Transitions = {"a"=>7, "b"=>0, "c"=>0, "?"=>0}
 7. Transitions = {"a"=>1, "b"=>2, "c"=>4, "?"=>0}
abacaba
abacaba
text length = 7, pattern length = 7
comparisons = 7,        matches = 1
```



```
% search -a fsm -p abacaba -s abababacababa
abababacababa
    abacaba
text length = 13, pattern length = 7
comparisons = 13,        matches = 1

% search -a fsm -p abacaba -s xxxxxxxxxxxxabacaba
xxxxxxxxxxxxabacaba
           abacaba
text length = 19, pattern length = 7
comparisons = 19,        matches = 1
```


States

```
class State

attr_accessor :transitions

def initialize
  @transitions = {}
end

def add_transition(char, destination)
  @transitions[char] = destination
end


end
```

Building

```
vector = failure_array(pattern)
pattern.each_char{|c| alphabet[c] = true}
alphabet[ANY] = true
start = s = State.new
for i in (0..plen)
  states[i] = (s = State.new)
  s.transitions[pattern[i]] = states.length if i<plen
  alphabet.each_key do |k|
    if i==0
      states[0].transitions[k] = 0 unless states[0].transitions[k]
    else
      j = i
      until j==-1
        if s1 = states[j].transitions[k]
          s.transitions[k] = s1
          break
        end
        j = vector[j]
      end
    end
  end
end
```

Make a start state

The main transition

Must have a transition for every character

All transitions from the start state return there.

Search for shorter prefixes.

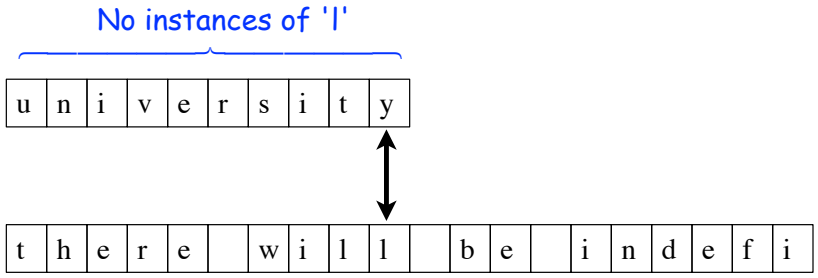
Searching

```
state = 0
for i in (0..text.length)
  c = text[i]
  c = ANY unless alphabet.include?(c)
  state = states[state].transitions[c]
  if state == plen
    matches += 1
    yield i-plen+1
  end
end
```

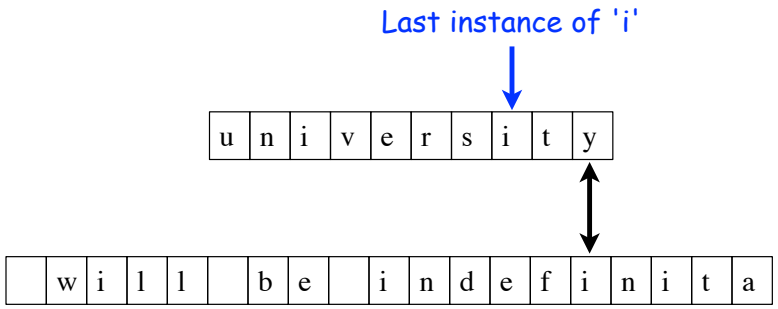

Boyer-Moore

- Compare window to text <u>from right to left</u>
- When a match fails, move the window <u>as far to the right as possible</u> consistent with not missing any matches.

Boyer-Moore-Horspool

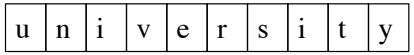


Boyer-Moore-Horspool



Boyer-Moore-Horspool

Skip	
a	10
b	10
...	
e	5
i	2
n	8
r	4
s	3
t	1
u	9
v	6
y	0



The skip Table

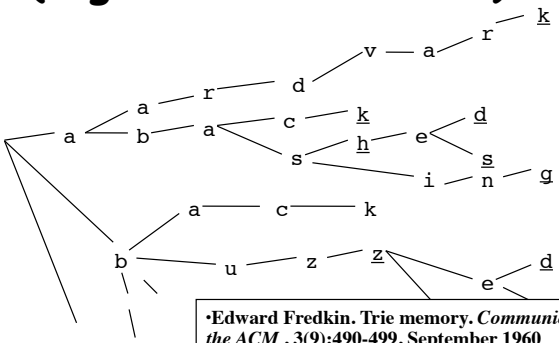
a b a c a b a

```
  0 : 7     1 : 7     2 : 7     3 : 7     4 : 7     5 : 7     6 : 7     7 : 7
  8 : 7     9 : 7    10 : 7    11 : 7    12 : 7    13 : 7    14 : 7    15 : 7
 16 : 7    17 : 7    18 : 7    19 : 7    20 : 7    21 : 7    22 : 7    23 : 7
 24 : 7    25 : 7    26 : 7    27 : 7    28 : 7    29 : 7    30 : 7    31 : 7
 32 : 7    33 : 7    34 : 7    35 : 7    36 : 7    37 : 7    38 : 7    39 : 7
 40 : 7    41 : 7    42 : 7    43 : 7    44 : 7    45 : 7    46 : 7    47 : 7
 48 : 7    49 : 7    50 : 7    51 : 7    52 : 7    53 : 7    54 : 7    55 : 7
 56 : 7    57 : 7    58 : 7    59 : 7    60 : 7    61 : 7    62 : 7    63 : 7
 64 : 7  65 A: 7    66 B: 7    67 C: 7    68 D: 7    69 E: 7    70 F: 7    71 G: 7
 72 H: 7    73 I: 7    74 J: 7    75 K: 7    76 L: 7    77 M: 7    78 N: 7    79 O: 7
 80 P: 7    81 Q: 7    82 R: 7    83 S: 7    84 T: 7    85 U: 7    86 V: 7    87 W: 7
 88 X: 7    89 Y: 7    90 Z: 7    91 : 7     92 : 7     93 : 7     94 : 7     95 : 7
 96 : 7    97 a: 2    98 b: 1    99 c: 3   100 d: 7   101 e: 7   102 f: 7   103 g: 7
104 h: 7   105 i: 7   106 j: 7   107 k: 7   108 l: 7   109 m: 7   110 n: 7   111 o: 7
112 p: 7   113 q: 7   114 r: 7   115 s: 7   116 t: 7   117 u: 7   118 v: 7   119 w: 7
120 x: 7   121 y: 7   122 z: 7   123 : 7    124 : 7    125 : 7    126 : 7    127 : 7
```


How to find the longest prefix of the pattern efficiently

Answer: Prefix Trees
(= backwards suffix trees)

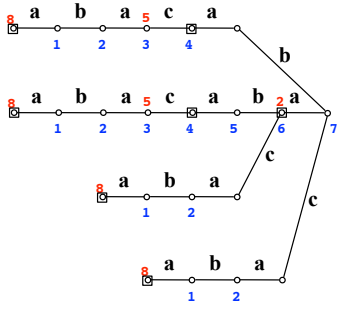
"Tries"
(Digital Search Trees)

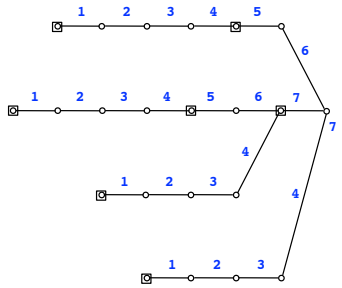


•Edward Fredkin. Trie memory. *Communications of the ACM* , 3(9):490-499, September 1960

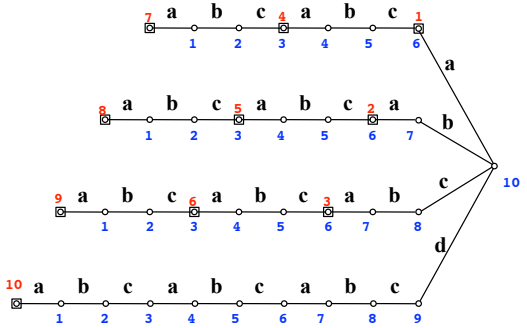
A suffix tree is a trie in which the words are the suffixes of some given string

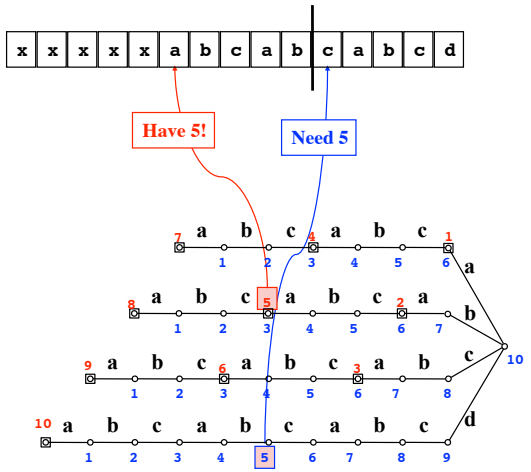
A prefix tree is (let us say) a trie in which the words are the prefixes for some given string, read from right to left.

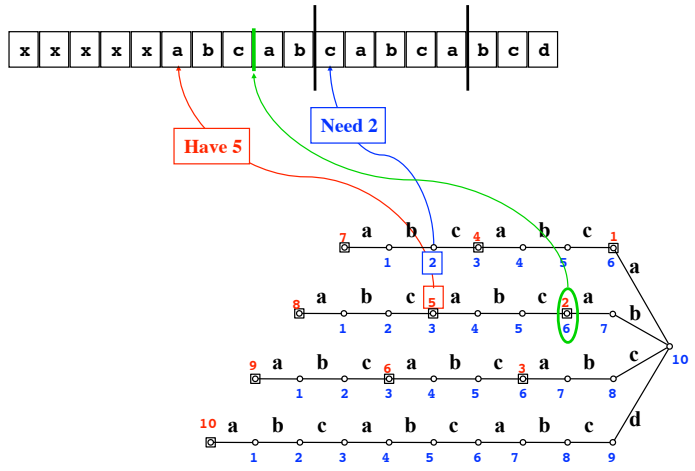


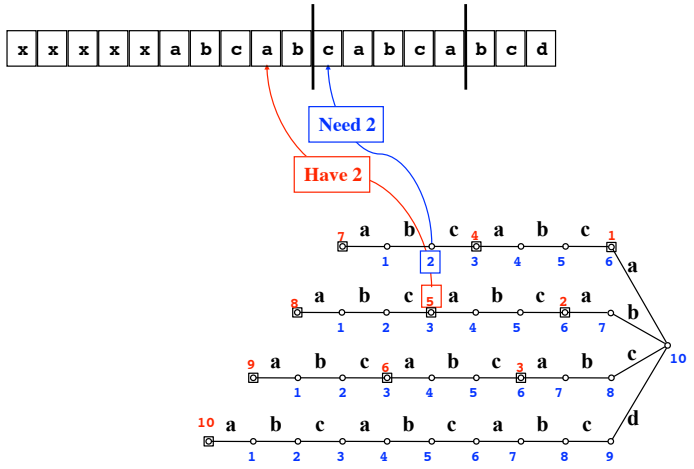


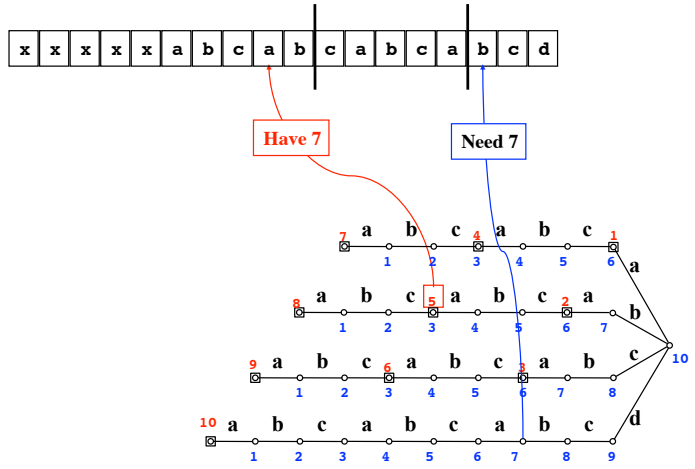
10 9 8 7 6 5 4 3 2 1
6 5 4 3 2 1

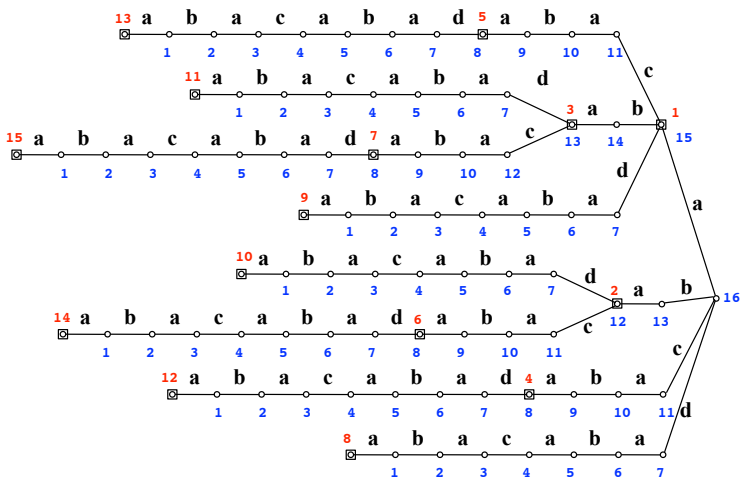



x x x x x a b c a b c a b c d
Have 5!
Need 5
a b c a b c
a b c a b c a
a b c a b c a b
a b c a b c a b c
10


x x x x x a b c a b c a b c a b c d
Have 5
Need 2
a b c a b c
a b c a b c a
a b c a b c a b
a b c a b c a b c
10


x x x x x a b c a b c a b c a b c d
Need 2
Have 2
a b c a b c
a b c a b c a
a b c a b c a b
a b c a b c a b c
10


x x x x x a b c a b c a b c a b c d
Have 7
Need 7
a b c a b c
a b c a b c a
a b c a b c a b
a b c a b c a b c
10



A Really Silly Idea

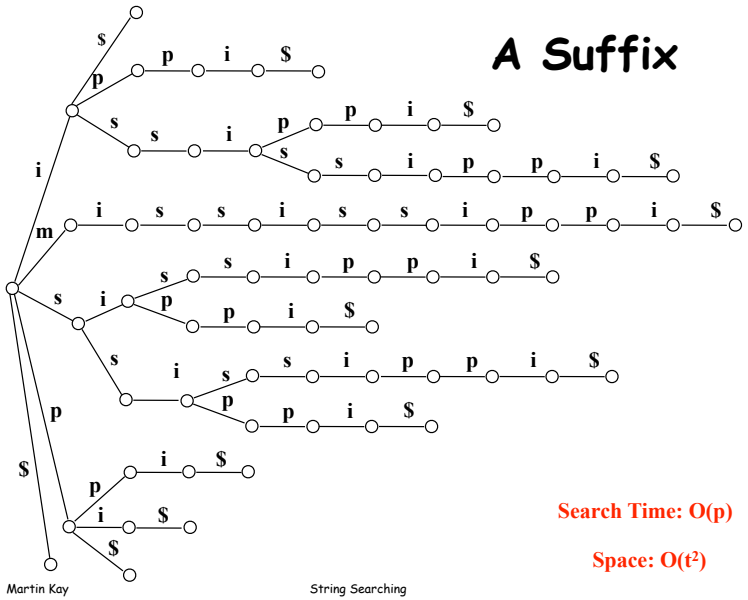
Index by all $\binom{n+1}{2}$ substrings of the text.

Observe: Every substring of a string is a prefix of some suffix of the string. So use a digital search tree to index on the suffixes of the text.

A Corpus

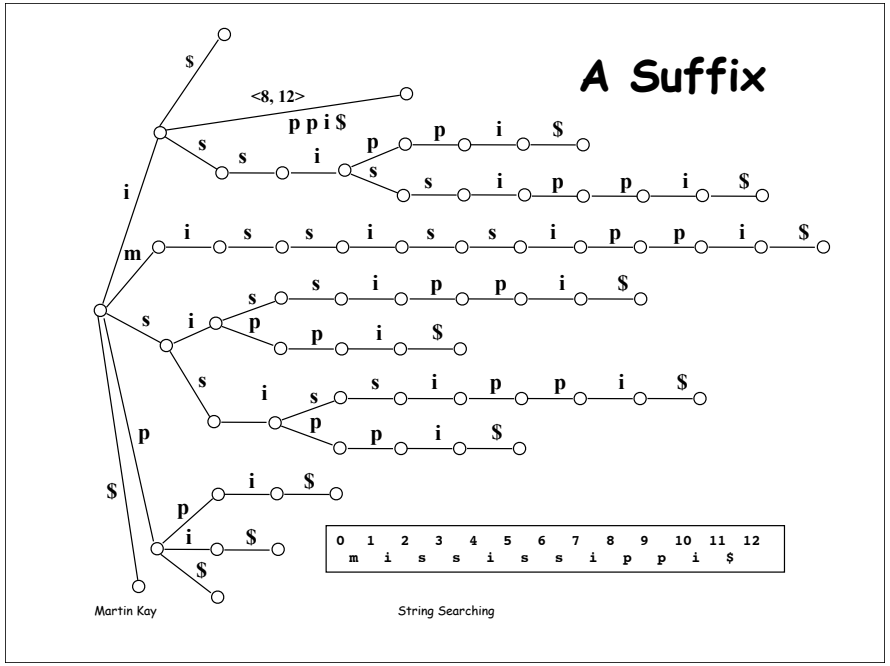
mississippi$

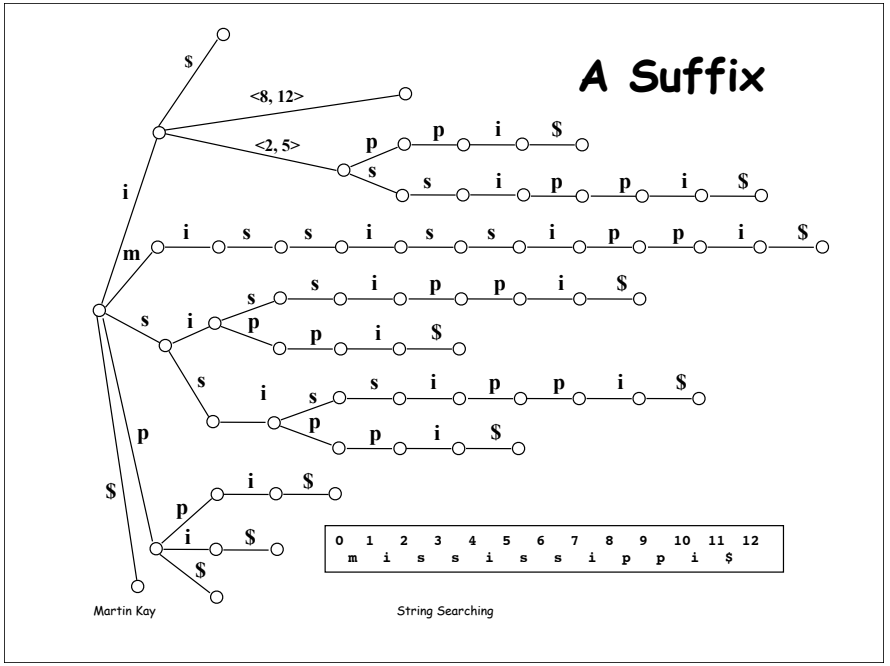
A Suffix

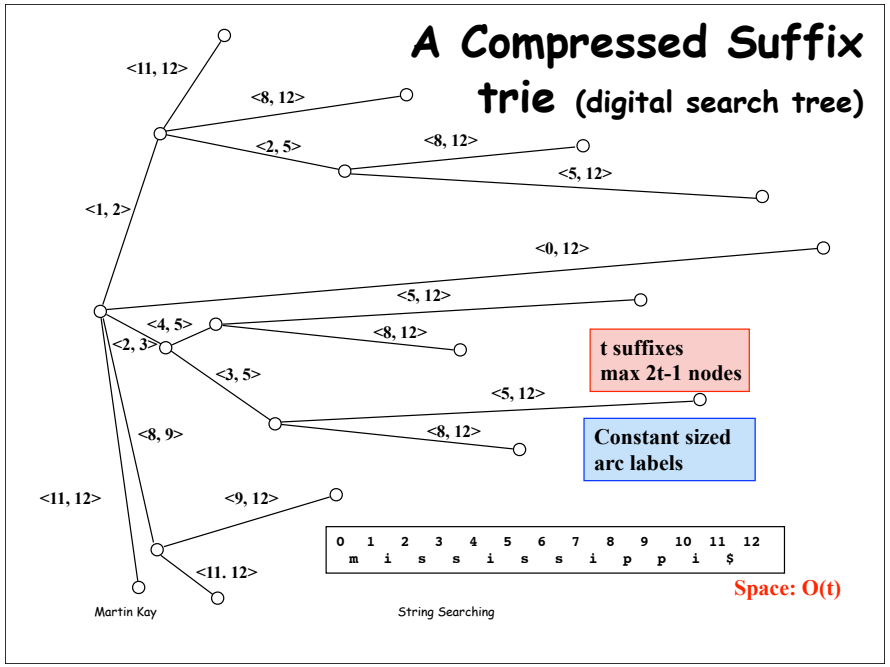


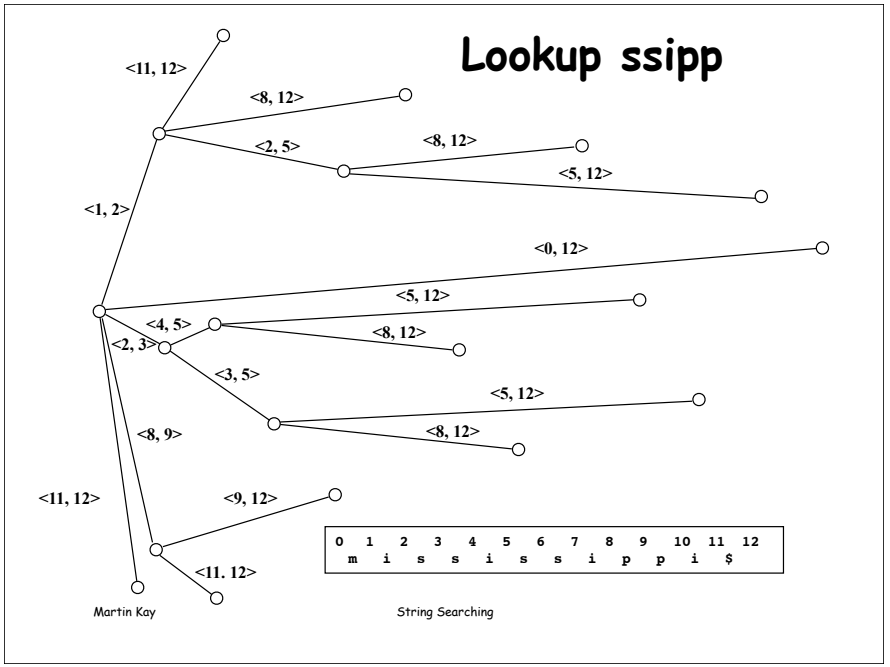
Search Time: O(p)

Space: $O(t^2)$

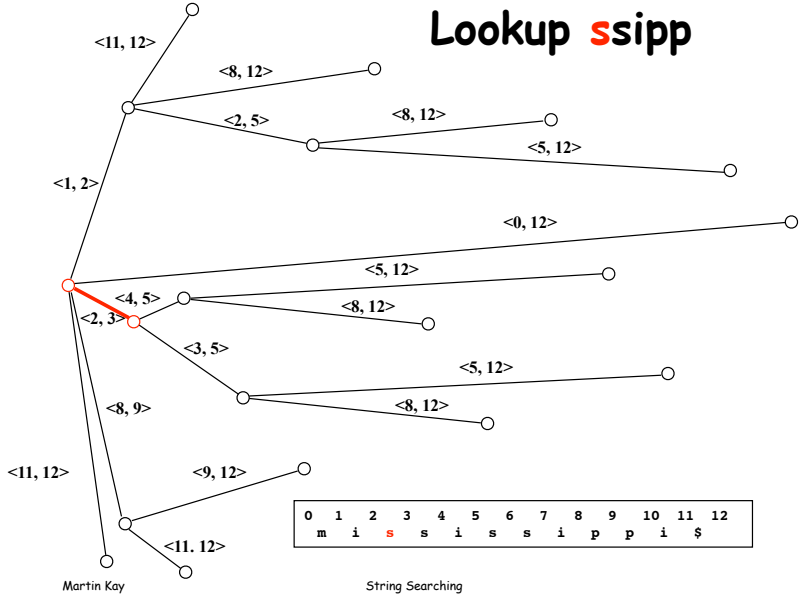

A Suffix
$
<8, 12>
p p i $
s s i p p i $
s s i p p i $
i
m i s s i s s i p p i $
s s i p p i $
s i p p i $
s i s s i p p i $
p p i $
p
$ i $
p i $
$
0 1 2 3 4 5 6 7 8 9 10 11 12
m i s s i s s i p p i $
Martin Kay
String Searching


A Suffix
$
<8, 12>
<2, 5>
p p i $
s s i p p i $
i
m i s s i s s i p p i $
s s i p p i $
s i p p i $
s i s s i p p i $
p p i $
p
$ i $
p i $
$
0 1 2 3 4 5 6 7 8 9 10 11 12
m i s s i s s i p p i $
Martin Kay
String Searching

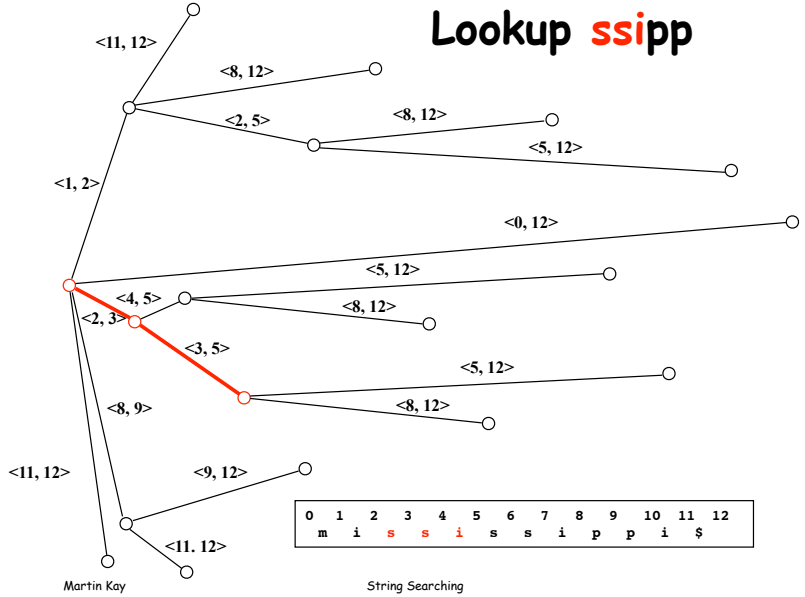

A Compressed Suffix
trie (digital search tree)
<11, 12>
<8, 12>
<2, 5>
<8, 12>
<5, 12>
<1, 2>
<0, 12>
<5, 12>
<4, 5>
<8, 12>
<2, 3>
<3, 5>
<5, 12>
<8, 12>
<8, 9>
<11, 12>
<9, 12>
<11. 12>
t suffixes
max 2t-1 nodes
Constant sized
arc labels
0 1 2 3 4 5 6 7 8 9 10 11 12
m i s s i s s i p p i $
Space: O(t)
Martin Kay
String Searching


Lookup ssipp
<11, 12>
<8, 12>
<2, 5>
<8, 12>
<5, 12>
<1, 2>
<0, 12>
<5, 12>
<4, 5>
<8, 12>
<2, 3>
<3, 5>
<5, 12>
<8, 12>
<8, 9>
<11, 12>
<9, 12>
<11. 12>
0 1 2 3 4 5 6 7 8 9 10 11 12
m i s s i s s i p p i $
Martin Kay
String Searching

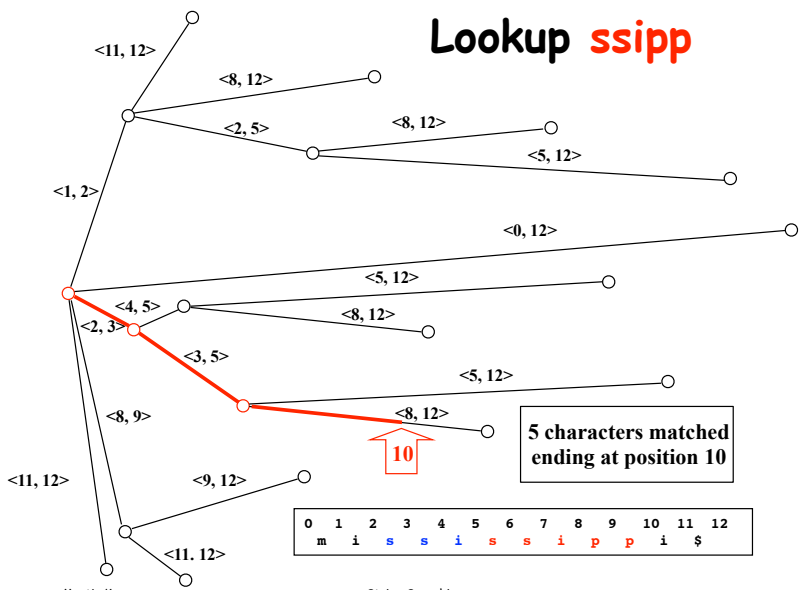
Lookup ssipp



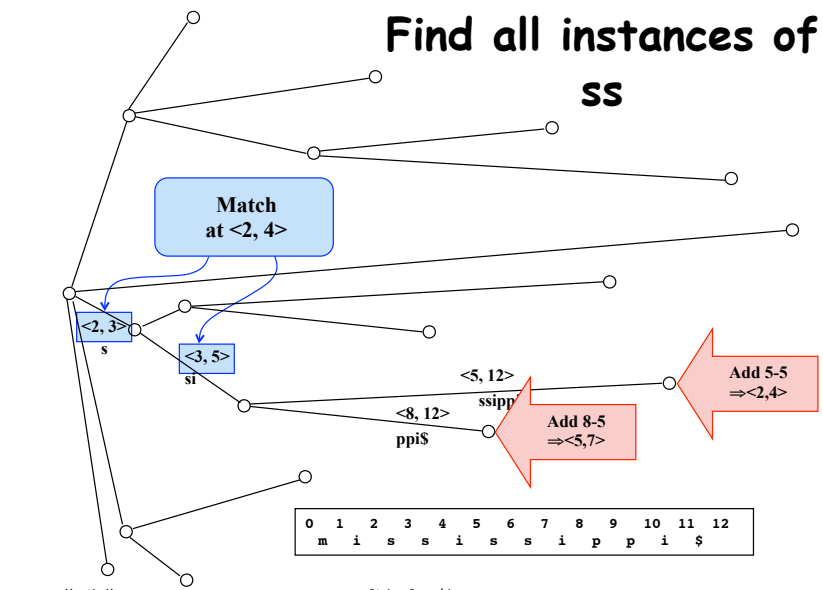
Lookup ssipp

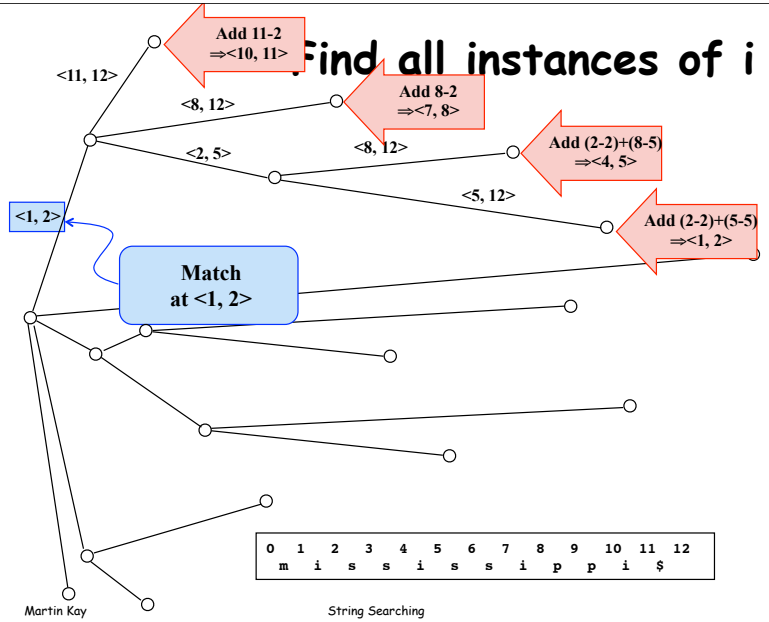


Lookup ssipp



Find all instances of ss





Observe:

A (sub)tree with n terminals contains a total of at most 2n-1 nodes. Therefore finding all occurrences, once the first has been found requires, at most, that 2 nodes be visited for each hit.