Strings Overview

Mark Matthews PhD

Summary

- Text Pattern Matching
- Pattern Matching Applications
- Java and Strings
- Naive / brute force approach
- Knuth-Morris-Pratt algorithm

String processing

String. Sequence of characters.

Important fundamental abstraction.

- Genomic sequences.
- Information processing.
- Communication systems (e.g., email).
- Programming systems (e.g., Java programs).
- …

" The digital information that underlies biochemistry, cell biology, and development can be represented by a simple string of G's, A's, T's and C's. This string is the root data structure of an organism's biology. " — *M. V. Olson*

Data structures for modelling life itself!

Substring search

Goal. Find pattern of length M in a text of length N.

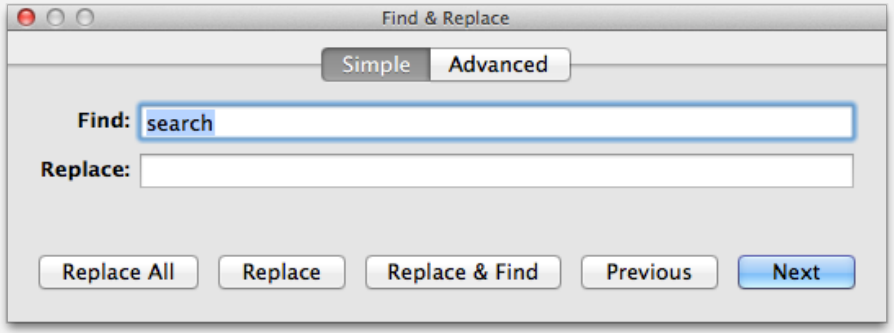
Substring search applications

Goal. Find pattern of length M in a text of length N.

typically N >> M

```
pattern ⟶ N E E D L E

text    ⟶ I N A H A Y S T A C K N E E D L E I N A
                                   match
```



Substring search applications

Goal. Find pattern of length M in a text of length N.

Computer forensics. Search memory or disk for signatures,
e.g., all URLs or RSA keys that the user has entered.

http://citp.princeton.edu/memory

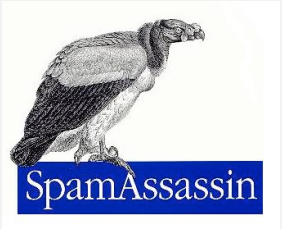
Substring search applications

Goal. Find pattern of length M in a text of length N.

typically N >> M

Identify patterns indicative of spam.

- PROFITS
- L0SE WE1GHT
- herbal Viagra
- There is no catch.
- This is a one-time mailing.
- This message is sent in compliance with spam regulations.



Substring search applications

Electronic surveillance.

Need to monitor all internet traffic. (security)

No way! (privacy)

Well, we're mainly interested in "ATTACK AT DAWN"

OK. Build a machine that just looks for that.

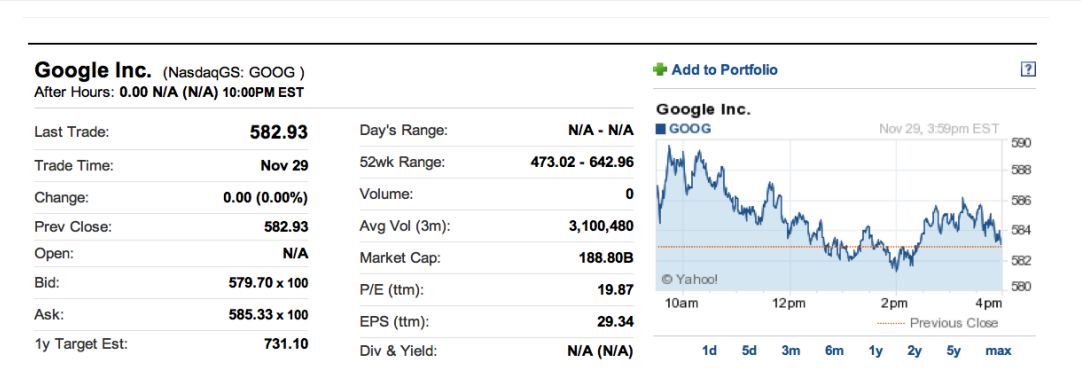
"ATTACK AT DAWN" substring search machine

found

Substring search applications

Screen scraping. Extract relevant data from web page.

Ex. Find string delimited by and after first occurrence of pattern Last Trade:.




```
...
<tr>
<td class= "yfnc_tablehead1"
width= "48%">
Last Trade:
</td>
<td class= "yfnc_tabledata1">
<big><b>452.92</b></big>
</td></tr>
<td class= "yfnc_tablehead1"
width= "48%">
Trade Time:
</td>
<td class= "yfnc_tabledata1">
...
```

Screen scraping: Java implementation

Java library. The indexOf() method in Java's string library returns the index of the first occurrence of a given string, starting at a given offset.

```
public class StockQuote
{
   public static void main(String[] args)
   {
      String name = "http://finance.yahoo.com/q?s=";
      In in = new In(name + args[0]);
      String text = in.readAll();
      int start    = text.indexOf("Last Trade:", 0);
      int from     = text.indexOf("<b>",  start);
      int to       = text.indexOf("</b>", from);
      String price = text.substring(from + 3, to);
      StdOut.println(price);
   }
}
```

```
% java StockQuote goog
582.93

% java StockQuote msft
24.84
```

Strings in Java

The char data type

C char data type. Typically an 8-bit integer.

- Supports 7-bit ASCII.
- Can represent at most 256 characters.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Hexadecimal to ASCII conversion table

some Unicode characters

Java char data type. A 16-bit unsigned integer.

- Supports original 16-bit Unicode.
- Supports 21-bit Unicode 3.0 (awkwardly).

Java's String Class

Pattern matching

text.contains(pattern) -> return boolean

text.indexOf(pattern) -> returns index or -1

Sub-task

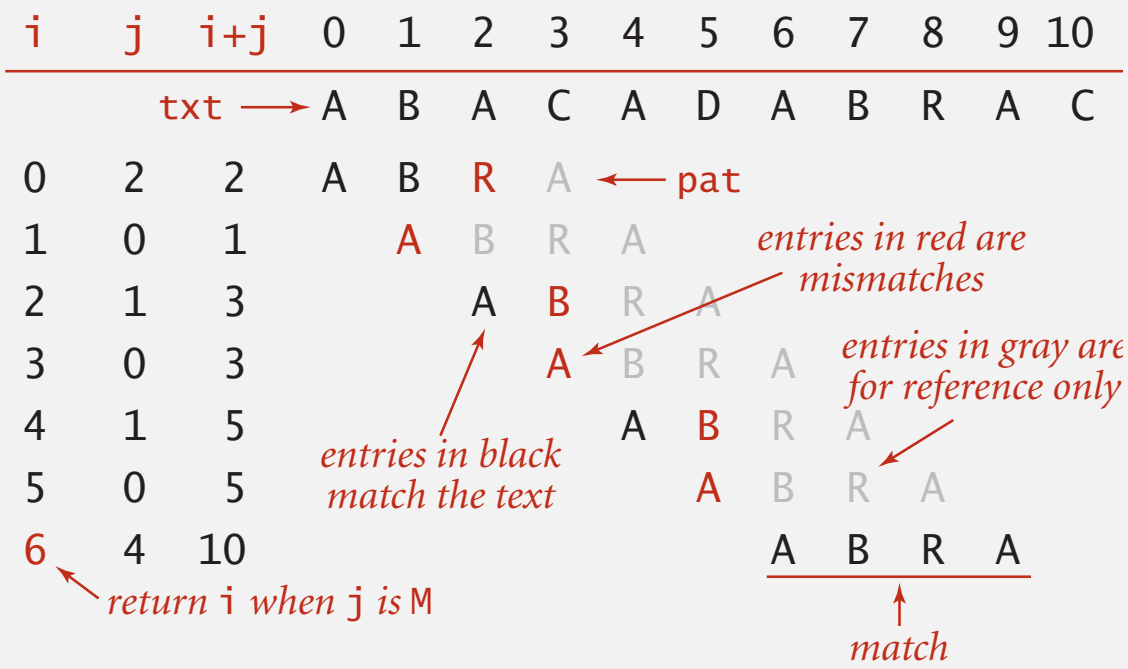
text.replace(pattern, substitute)

text.split(pattern)

Pattern Matching with Brute Force

Brute-force substring search

Check for pattern starting at each text position.



Brute-force substring search: Java implementation

Check for pattern starting at each text position.

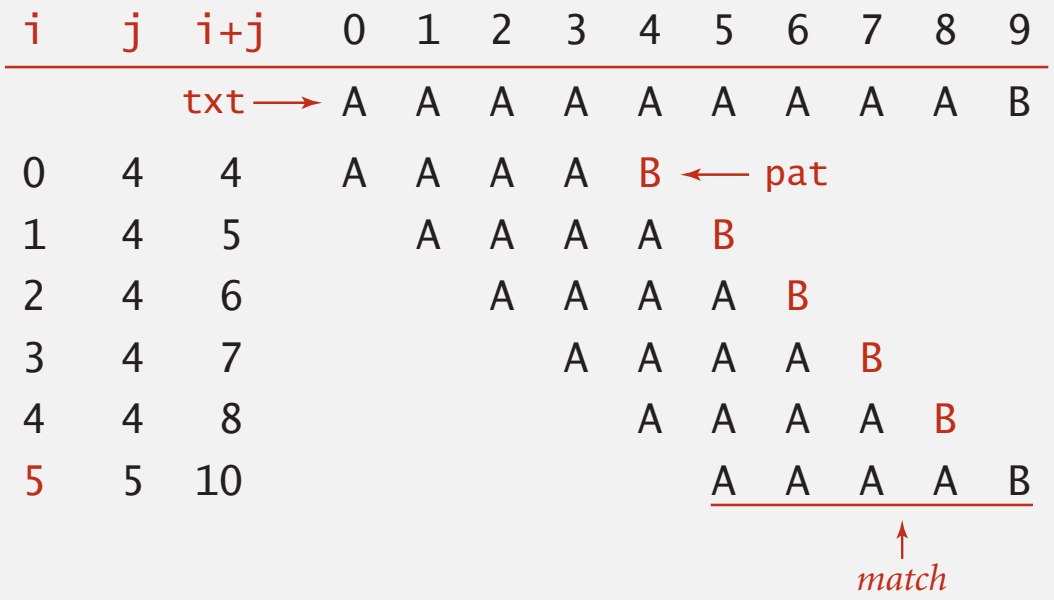

```
public static int search(String pat, String txt)
{
   int M = pat.length();
   int N = txt.length();
   for (int i = 0; i <= N - M; i++)
   {
      int j;
      for (j = 0; j < M; j++)
         if (txt.charAt(i+j) != pat.charAt(j))
            break;
      if (j == M) return i;
   }
   return N;
}
```

← index in text where pattern starts

← not found

Brute-force substring search: worst case

Brute-force algorithm can be slow if text and pattern are repetitive.

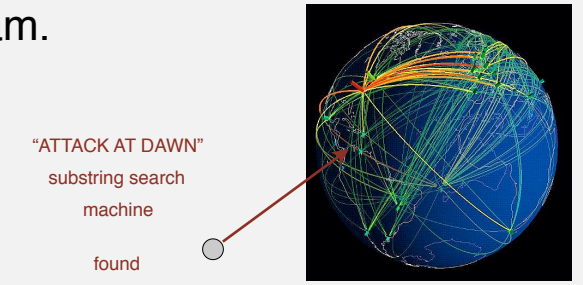


Worst case. $\sim M N$ char compares.

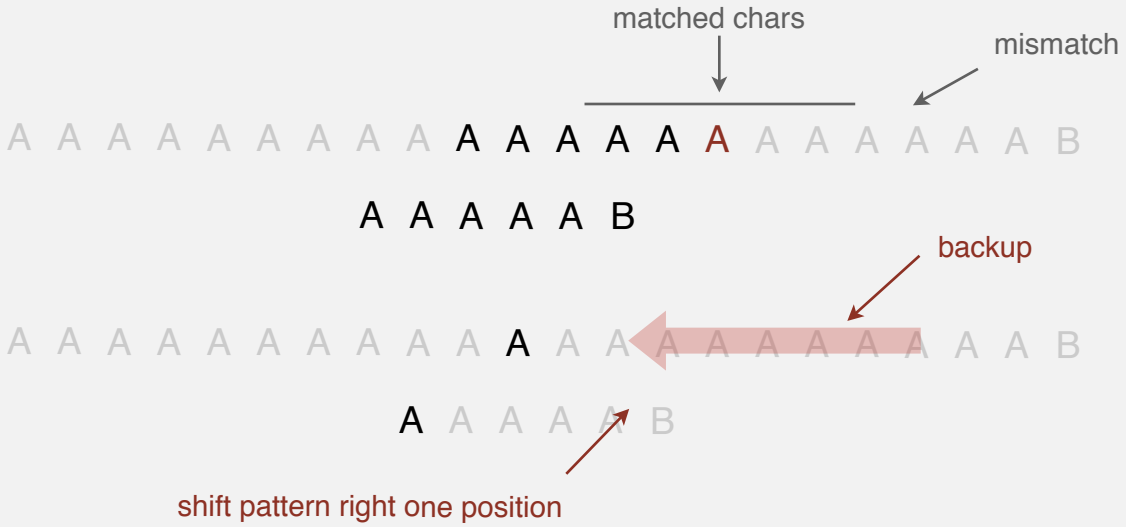
Backup

In many applications, we want to avoid backup in text stream.

- Treat input as stream of data.
- Abstract model: standard input.



Brute-force algorithm needs backup for every mismatch.



Approach 1. Maintain buffer of last M characters.

Approach 2. Better algorithms to avoid backup.

Algorithmic challenges in substring search

Brute-force is not always good enough.

Theoretical challenge. Linear-time guarantee. ← fundamental algorithmic problem

Practical challenge. Avoid backup in text stream. ← often no room or time to save text

Now is the time for all people to come to the aid of their party. Now is the time for all good people to come to the aid of their party. Now is the time for many good people to come to the aid of their party. Now is the time for all good people to come to the aid of their party. Now is the time for a lot of good people to come to the aid of their party. Now is the time for all of the good people to come to the aid of their party. Now is the time for all good people to come to the aid of their party. Now is the time for each good person to come to the aid of their party. Now is the time for all good people to come to the aid of their party. Now is the time for all good Republicans to come to the aid of their party. Now is the time for all good people to come to the aid of their party. Now is the time for many or all good people to come to the aid of their party. Now is the time for all good people to come to the aid of their party. Now is the time for all good Democrats to come to the aid of their party. Now is the time for all people to come to the aid of their party. Now is the time for all good people to come to the aid of their party. Now is the time for many good people to come to the aid of their party. Now is the time for all good people to come to the aid of their party. Now is the time for a lot of good people to come to the aid of their party. Now is the time for all of the good people to come to the aid of their party. Now is the time for all good people to come to the aid of their attack at dawn party. Now is the time for each person to come to the aid of their party. Now is the time for all good people to come to the aid of their party. Now is the time for all good Republicans to come to the aid of their party. Now is the time for all good people to come to the aid of their party. Now is the time for many or all good people to come to the aid of their party. Now is the time for all good people to come to the aid of their party. Now is the time for all good Democrats to come to the aid of their party.

Aside: Brute Force algorithm strategy

Brute force is a powerful design pattern
Typically used when we have some function we want to optimise

Approach:
- enumerate all possible configurations or options
- pick the best of all the options

Advantages:
- typically easy to implement

Disadvantages:
- performance is proportional to number of potential solutions

How to Set Up Your New Car

It’s full of electronics, so think of it as another device that you can make to be more you.

Tell me about the last time you bought a new car and sat down to read the entire owner’s manual. Go ahead — I’ll wait.

Reading an owner’s manual cover to cover would be no small commitment. The book for the 2020 BMW X5, for instance, is 404 pages. The guides for the new Cadillac XT6 and Volvo XC90 are only slightly slimmer, at 384 and 208 pages, respectively. A Mercedes-Benz GLC 300 coupe’s plush manual lays out instructions over 607 pages, including a 39-page index.

Instead of nudging you into reading the whole thing or trying to remember the dealer’s new car walk-through of features, let’s instead do the CliffsNotes version and concentrate on the personalization settings. Much like choosing the display in a new TV and the ringtones and notifications you prefer in a new cellphone, automakers allow you to personalize a car too. And you’ll enjoy the ride a lot more if you do.

Put the paper owner’s manual back and open the electronic version that’s available in many cars. The driver’s seat is a comfortable spot, and you have a cup holder for your coffee, so let’s settle in and set up your new baby. And if you’ve owned the car for several years and haven’t done this, you are in for a nice surprise.

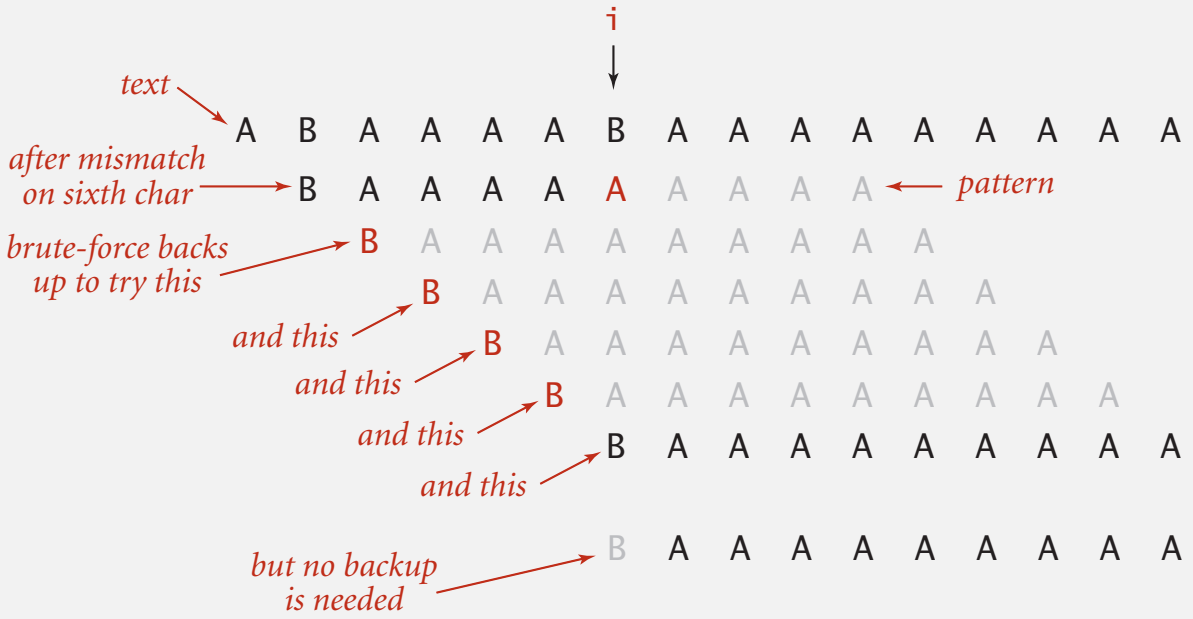
Knuth-Morris-Pratt Algorithm

Knuth-Morris-Pratt substring search

Intuition. Suppose we are searching in text for pattern BAAAAAAAAA.

- Suppose we match 5 chars in pattern, with mismatch on 6^{th} char.
- We know previous 6 chars in text are BAAAAB.
- Don't need to back up text pointer!

assuming { A, B } alphabet



Knuth-Morris-Pratt algorithm. Clever method to always avoid backup. (!)

Achieves running time of O (n + m) -> O(n)

Knuth-Morris-Pratt: brief history

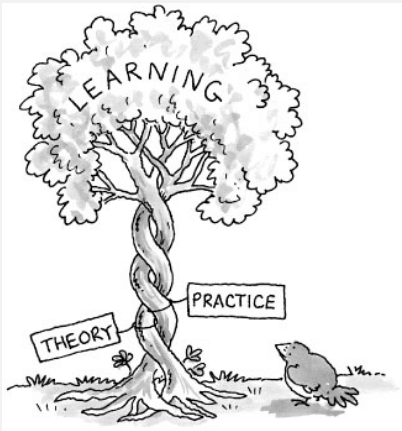
- Independently discovered by two theoreticians and a hacker.
  - Knuth: inspired by esoteric theorem, discovered linear algorithm
  - Pratt: made running time independent of alphabet size
  - Morris: built a text editor for the CDC 6400 computer
- Theory meets practice.

SIAM J. COMPUT.
Vol. 6, No. 2, June 1977

FAST PATTERN MATCHING IN STRINGS*

DONALD E. KNUTH†, JAMES H. MORRIS, JR.‡ AND VAUGHAN R. PRATT¶

Abstract. An algorithm is presented which finds all occurrences of one given string within another, in running time proportional to the sum of the lengths of the strings. The constant of proportionality is low enough to make this algorithm of practical use, and the procedure can also be extended to deal with some more general pattern-matching problems. A theoretical application of the algorithm shows that the set of concatenations of even palindromes, i.e., the language $\{\alpha\alpha^R\}^*$, can be recognized in linear time. Other algorithms which run even faster on the average are also considered.




Don Knuth


Jim Morris


Vaughan Pratt

Donald Knuth: Story behind KMP algorithm

https://www.youtube.com/watch?v=jAoTQRlhzQ8

Knuth-Morris-Pratt substring search: Java implementation

Key differences from brute-force implementation.

- Need to precompute dfa[][] from pattern.
- Text pointer i never decrements.

```
public int search(String txt)
{
   int i, j, N = txt.length();
   for (i = 0, j = 0; i < N && j < M; i++)
      j = dfa[txt.charAt(i)][j];
   if (j == M) return i - M;
   else        return N;
}
```

no backup

Running time.

- Simulate DFA on text: at most N character accesses

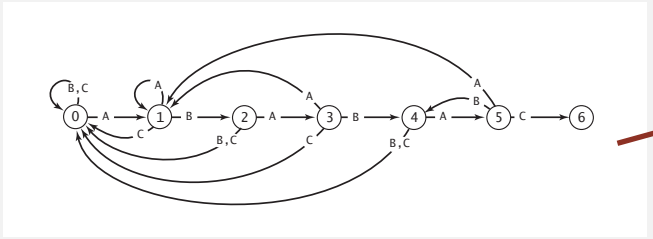
Knuth-Morris-Pratt substring search: Java implementation

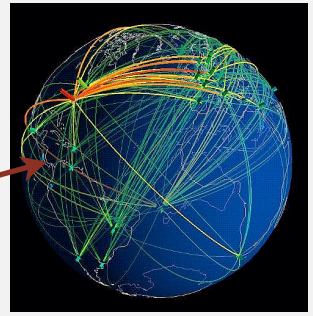
Key differences from brute-force implementation.

- Need to precompute dfa[][] from pattern.
- Text pointer i never decrements.
- Could use input stream.

```
public int search(In in)
{
   int i, j;
   for (i = 0, j = 0; !in.isEmpty() && j < M; i++)
      j = dfa[in.readChar()][j];
   if (j == M) return i - M;
   else        return NOT_FOUND;
}
```

no backup





Deterministic finite state automaton (DFA)

DFA is abstract string-searching machine.

- Finite number of states (including start and halt).
- Exactly one transition for each char in alphabet.
- Accept if sequence of transitions leads to halt state.

internal representation

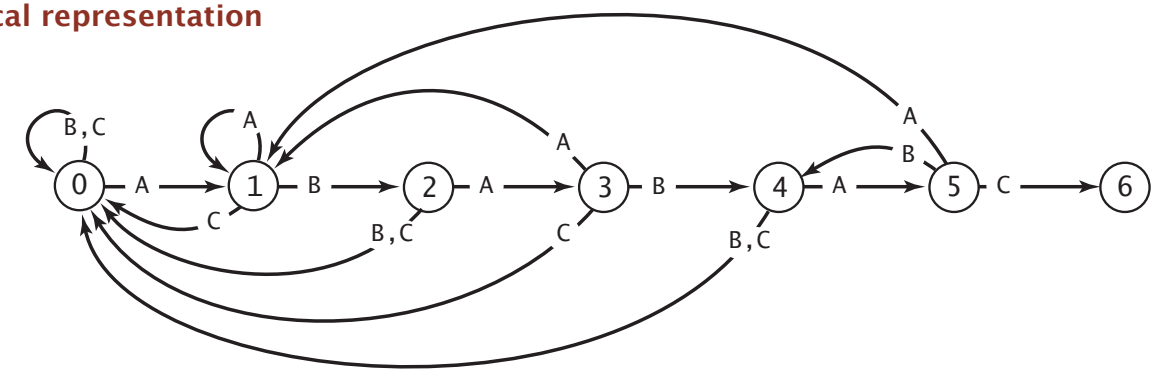
	j	0	1	2	3	4	5
pat.charAt(j)		A	B	A	B	A	C
dfa[][j]	A	1	1	3	1	5	1
	B	0	2	0	4	0	4
	C	0	0	0	0	0	6

If in state j reading char c:

if j is 6 halt and accept

else move to state dfa[c][j]

graphical representation



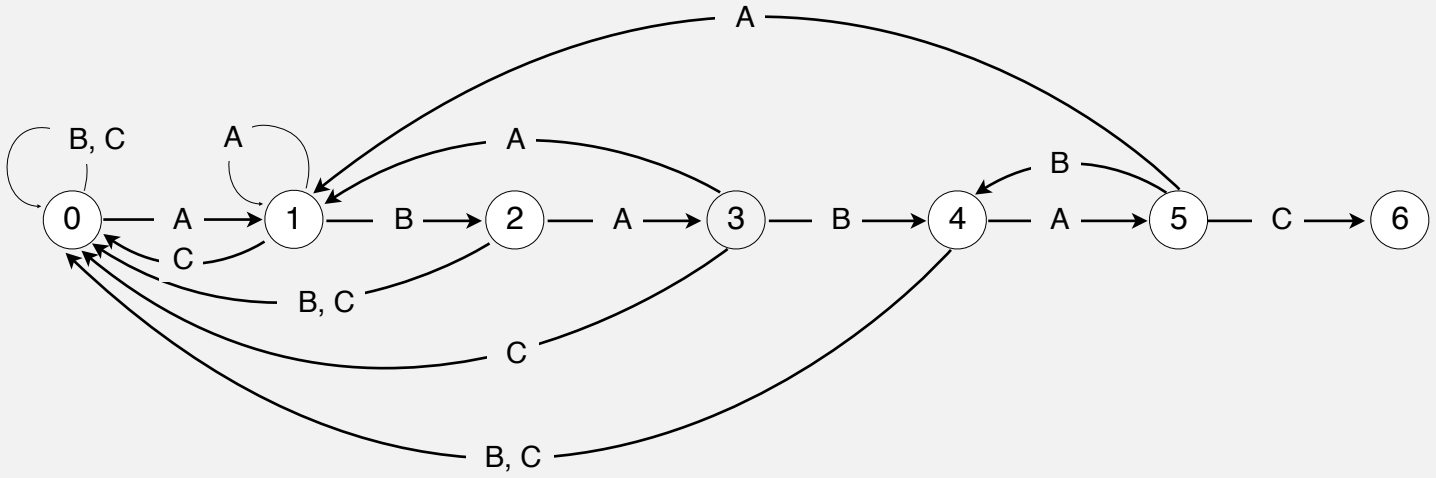
Knuth-Morris-Pratt demo: DFA simulation

A A B A C A A B A B A C A A

pat.charAt(j)

dfa[][j]

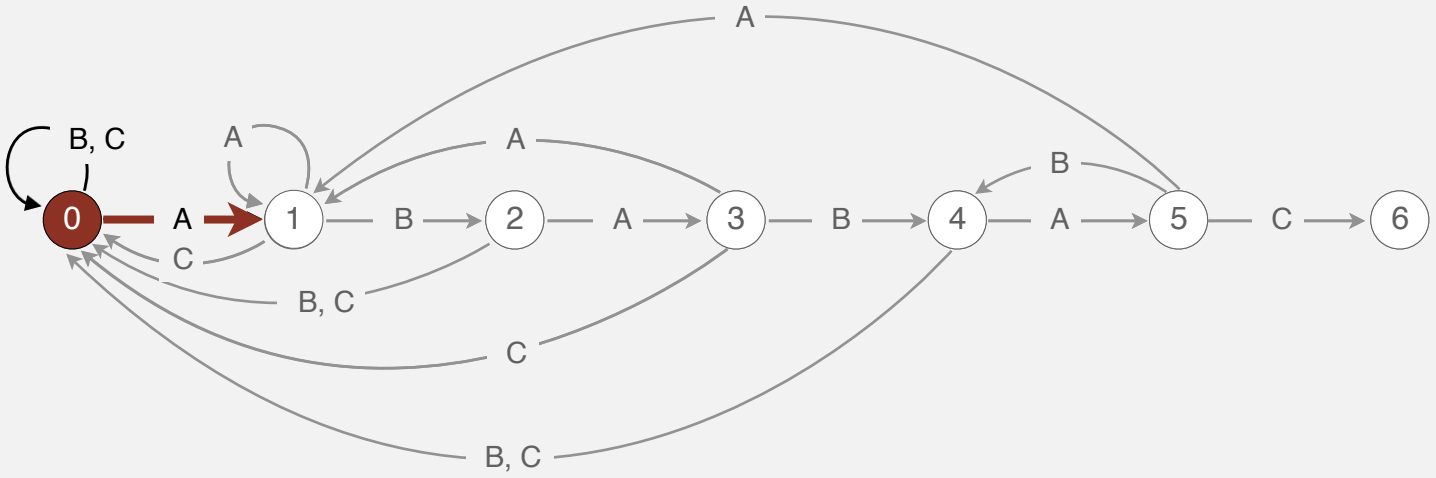
	0	1	2	3	4	5
	A	B	A	B	A	C
A	1	1	3	1	5	1
B	0	2	0	4	0	4
C	0	0	0	0	0	6



Knuth-Morris-Pratt demo: DFA simulation

A A B A C A A B A B A C A A

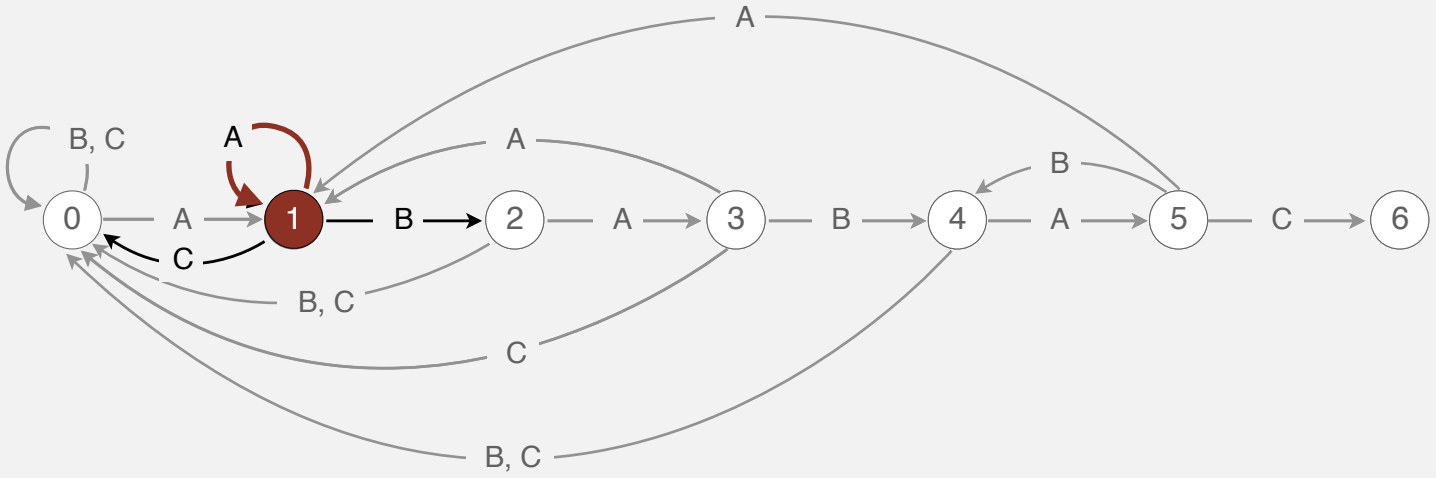
pat.charAt(j)	0	1	2	3	4	5
	A	B	A	B	A	C
dfa[][j] A	1	1	3	1	5	1
dfa[][j] B	0	2	0	4	0	4
dfa[][j] C	0	0	0	0	0	6



Knuth-Morris-Pratt demo: DFA simulation

A A B A C A A B A B A C A A

pat.charAt(j)	0	1	2	3	4	5
	A	B	A	B	A	C
dfa[][j] A	1	1	3	1	5	1
dfa[][j] B	0	2	0	4	0	4
dfa[][j] C	0	0	0	0	0	6



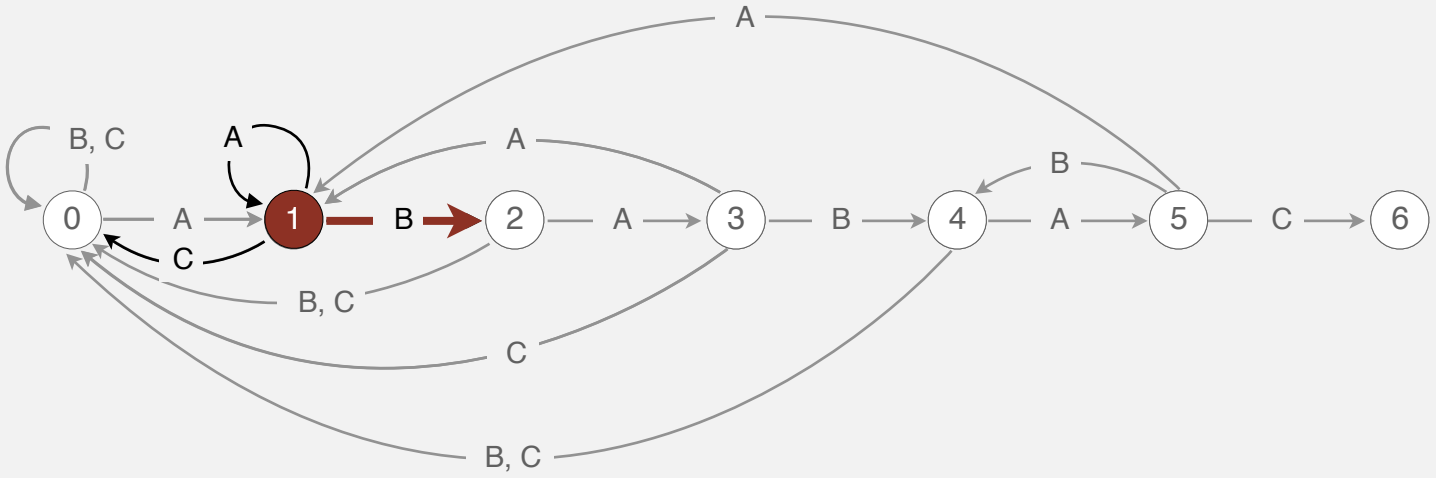
Knuth-Morris-Pratt demo: DFA simulation

A A B A C A A B A B A C A A

pat.charAt(j)

dfa[][j]

	0	1	2	3	4	5
	A	B	A	B	A	C
A	1	1	3	1	5	1
B	0	2	0	4	0	4
C	0	0	0	0	0	6



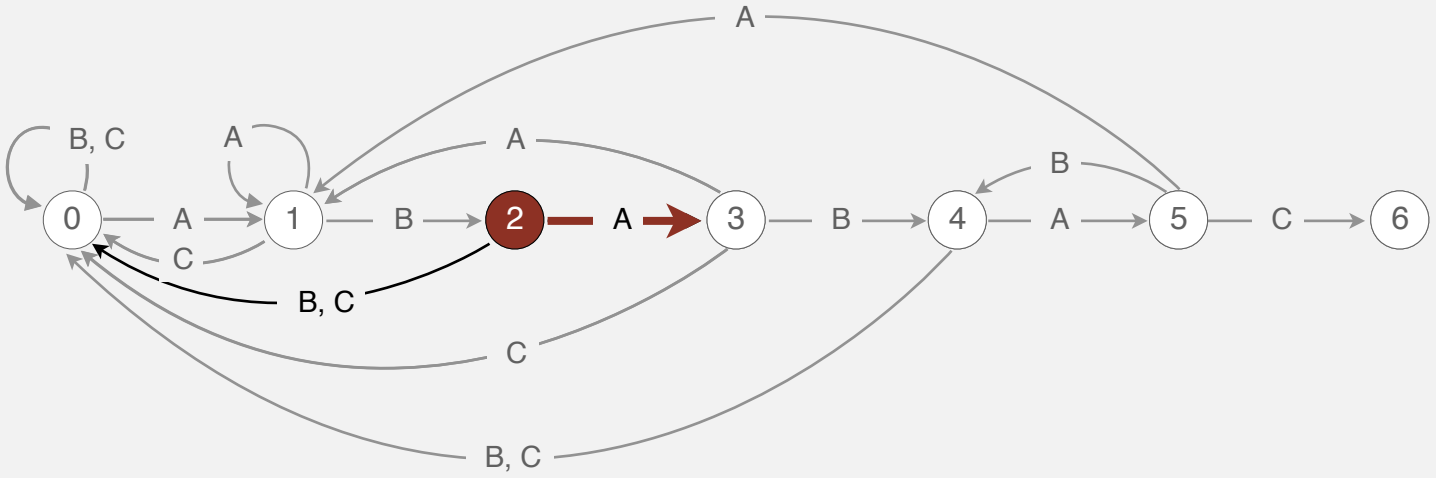
Knuth-Morris-Pratt demo: DFA simulation

A A B A C A A B A B A C A A

pat.charAt(j)

dfa[][j]

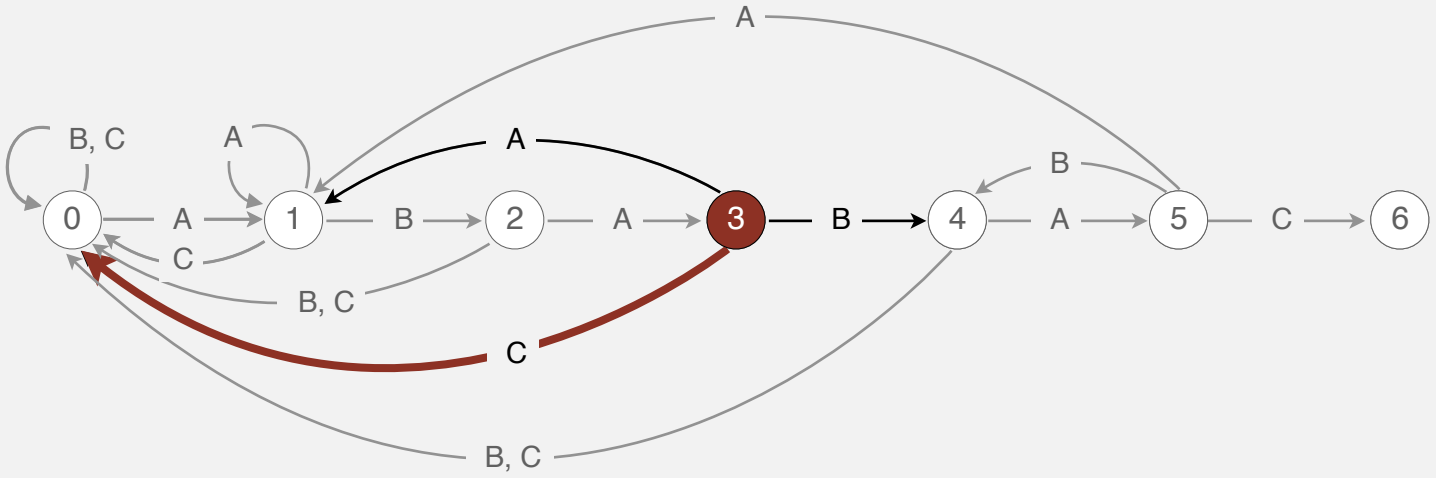
	0	1	2	3	4	5
	A	B	A	B	A	C
A	1	1	3	1	5	1
B	0	2	0	4	0	4
C	0	0	0	0	0	6



Knuth-Morris-Pratt demo: DFA simulation

A A B A C A A B A B A C A A

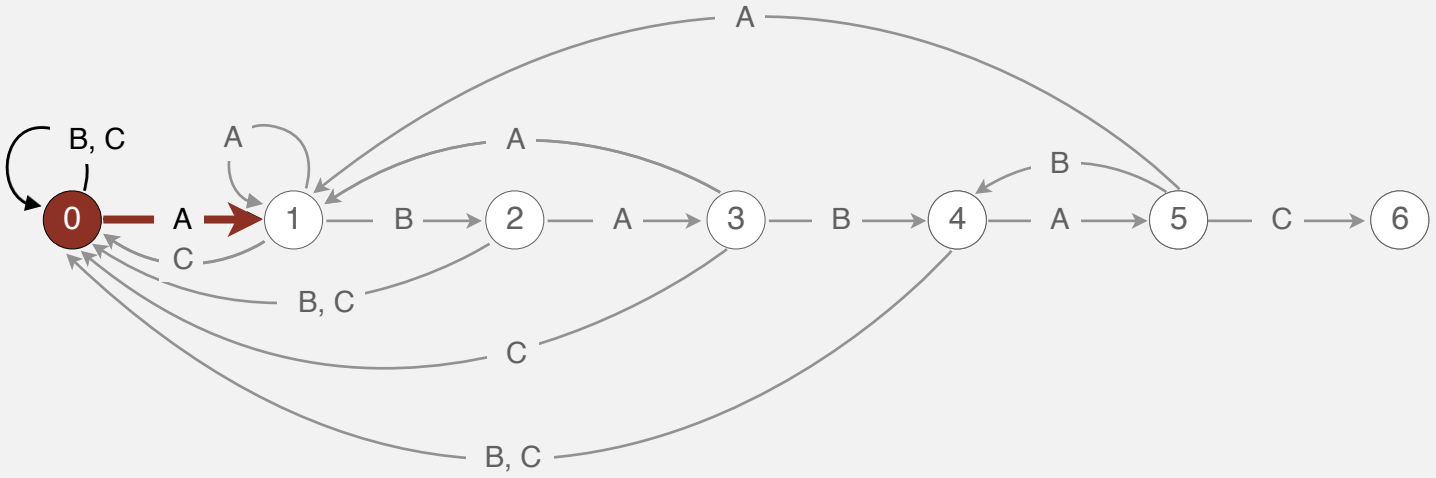
pat.charAt(j)	0	1	2	3	4	5
	A	B	A	B	A	C
dfa[][j] A	1	1	3	1	5	1
dfa[][j] B	0	2	0	4	0	4
dfa[][j] C	0	0	0	0	0	6



Knuth-Morris-Pratt demo: DFA simulation

A A B A C A A B A B A C A A

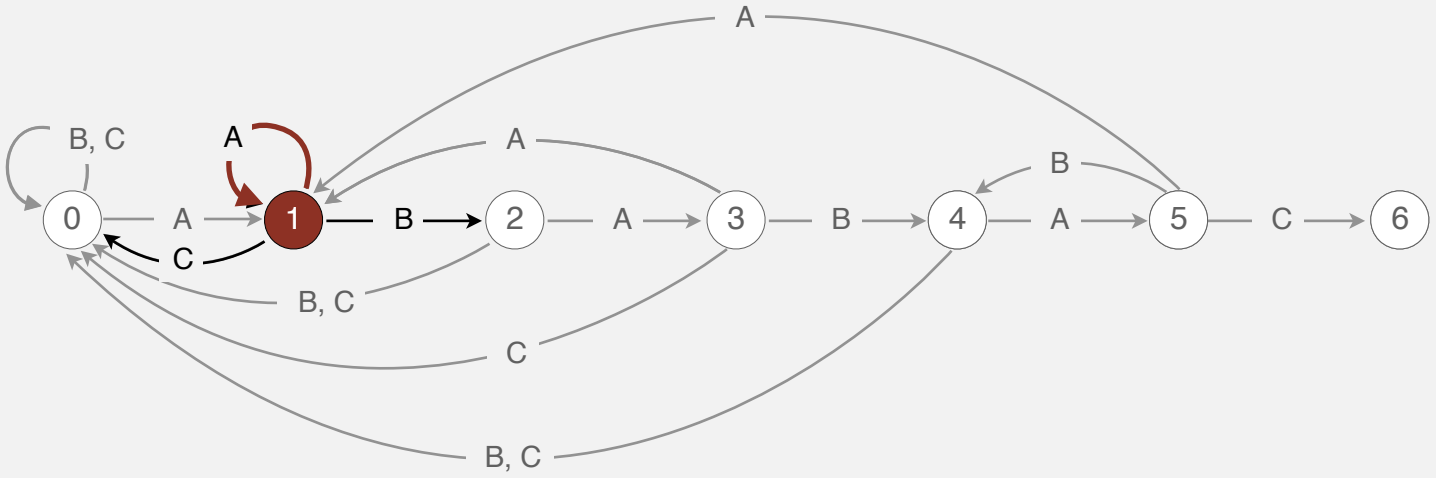
pat.charAt(j)	0	1	2	3	4	5
	A	B	A	B	A	C
dfa[][j] A	1	1	3	1	5	1
dfa[][j] B	0	2	0	4	0	4
dfa[][j] C	0	0	0	0	0	6



Knuth-Morris-Pratt demo: DFA simulation

A A B A C A A B A B A C A A

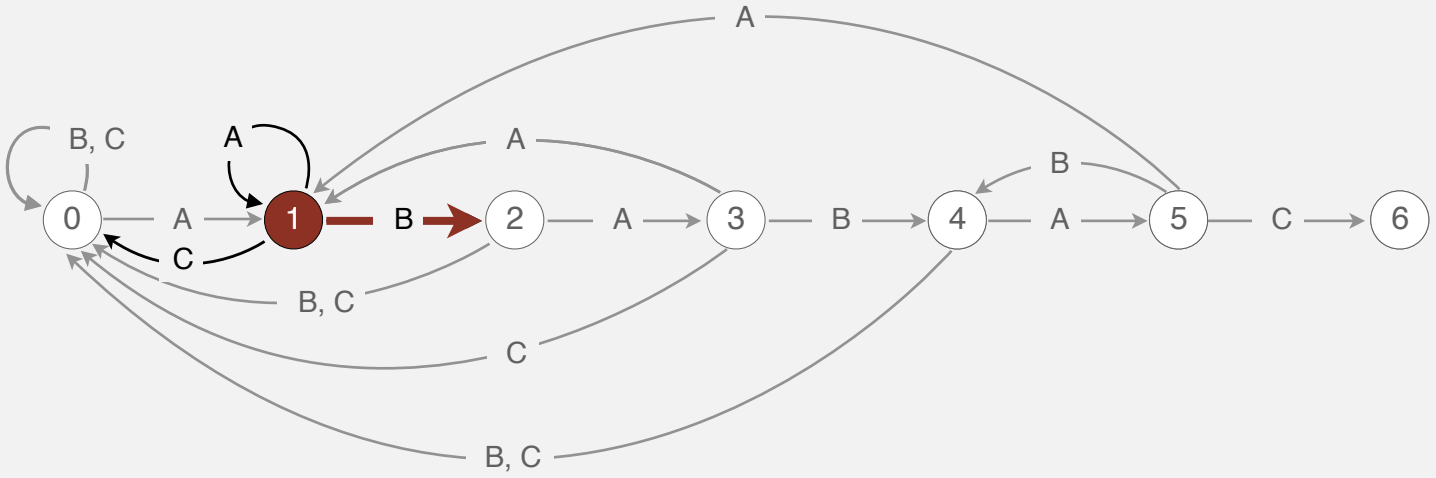
pat.charAt(j)	0	1	2	3	4	5
	A	B	A	B	A	C
dfa[][j] A	1	1	3	1	5	1
dfa[][j] B	0	2	0	4	0	4
dfa[][j] C	0	0	0	0	0	6



Knuth-Morris-Pratt demo: DFA simulation

A A B A C A A B A B A C A A

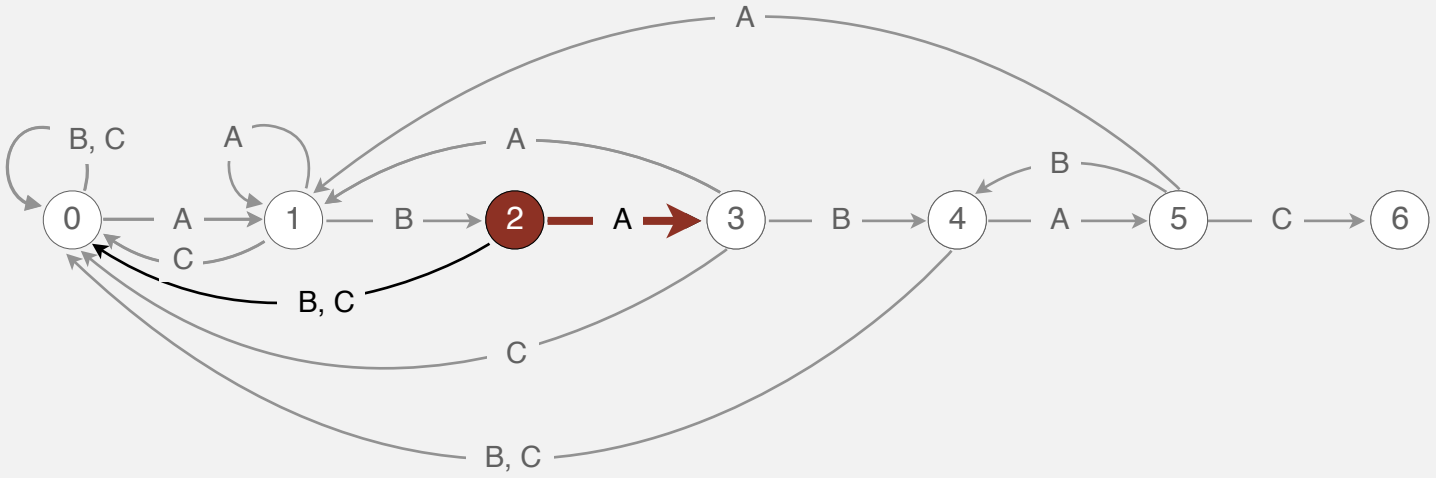
pat.charAt(j)	0	1	2	3	4	5
	A	B	A	B	A	C
dfa[][j] A	1	1	3	1	5	1
dfa[][j] B	0	2	0	4	0	4
dfa[][j] C	0	0	0	0	0	6



Knuth-Morris-Pratt demo: DFA simulation

A A B A C A A B A B A C A A

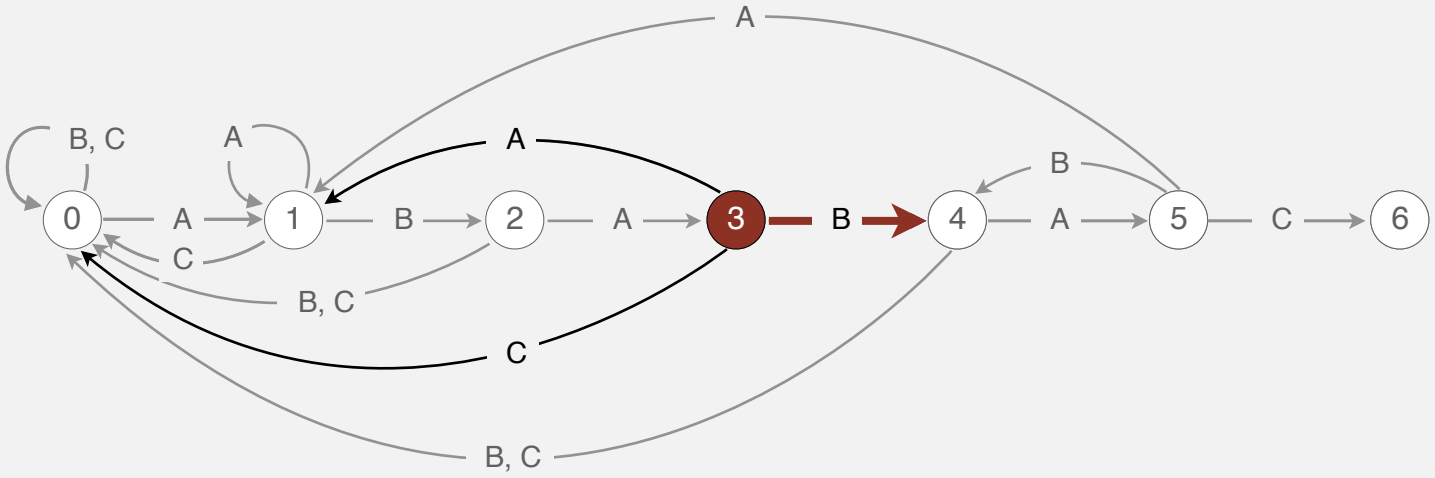
pat.charAt(j)	0	1	2	3	4	5
	A	B	A	B	A	C
dfa[][j] A	1	1	3	1	5	1
dfa[][j] B	0	2	0	4	0	4
dfa[][j] C	0	0	0	0	0	6



Knuth-Morris-Pratt demo: DFA simulation

A A B A C A A B A B A C A A

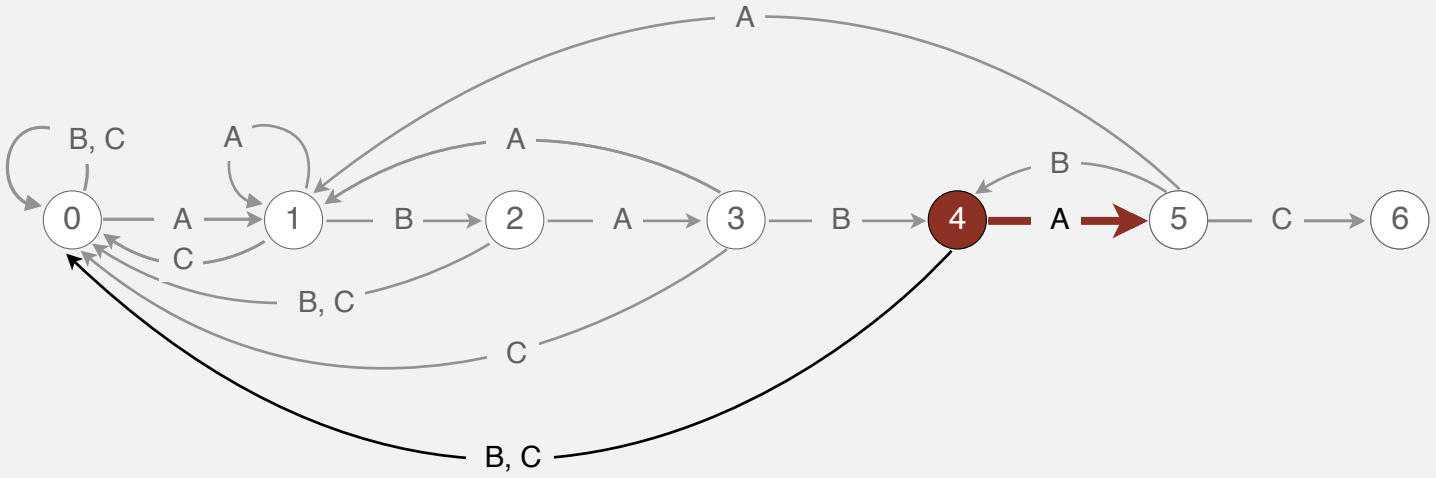
pat.charAt(j)	0	1	2	3	4	5
	A	B	A	B	A	C
dfa[][j] A	1	1	3	1	5	1
dfa[][j] B	0	2	0	4	0	4
dfa[][j] C	0	0	0	0	0	6



Knuth-Morris-Pratt demo: DFA simulation

A A B A C A A B A B A C A A

pat.charAt(j)	0	1	2	3	4	5
	A	B	A	B	A	C
dfa[][j] A	1	1	3	1	5	1
dfa[][j] B	0	2	0	4	0	4
dfa[][j] C	0	0	0	0	0	6

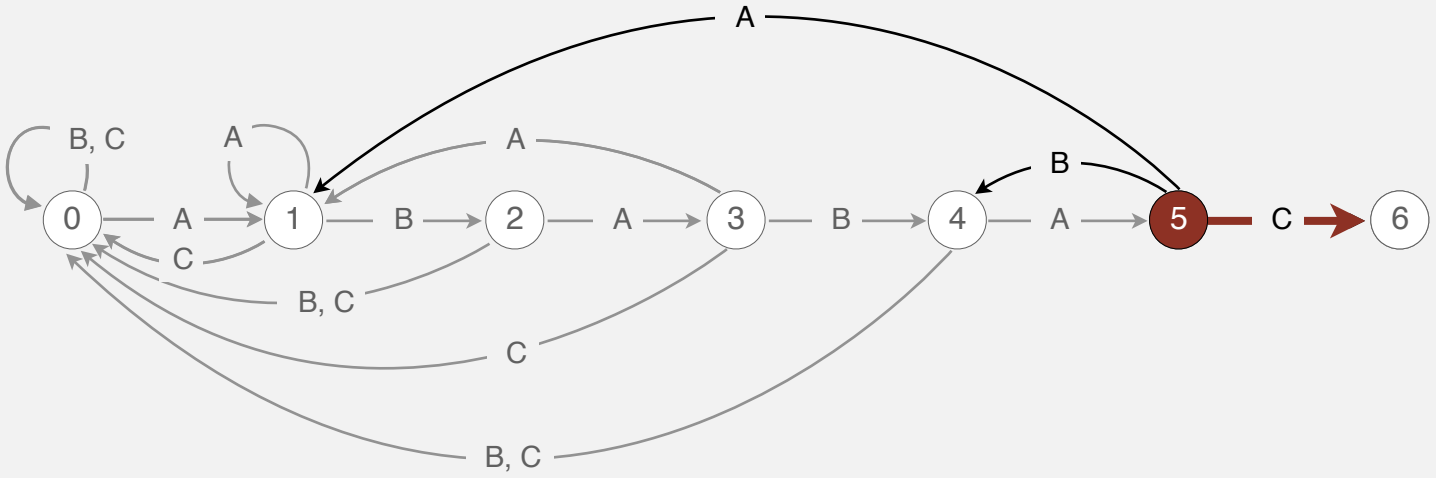


Knuth-Morris-Pratt demo: DFA simulation

A A B A C A A B A B A C A A

pat.charAt(j)	0	1	2	3	4	5
	A	B	A	B	A	C
A	1	1	3	1	5	1
B	0	2	0	4	0	4
C	0	0	0	0	0	6

dfa[][j]



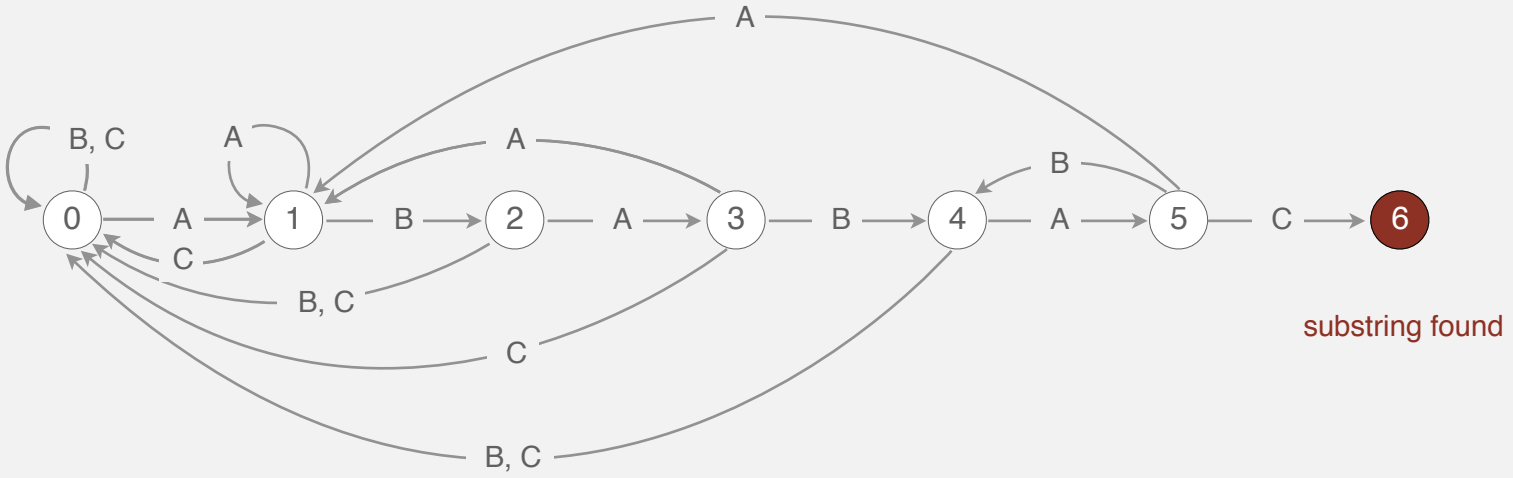
Knuth-Morris-Pratt demo: DFA simulation

A A B A C A A B A B A C A A

pat.charAt(j)

dfa[][j]

	0	1	2	3	4	5
	A	B	A	B	A	C
A	1	1	3	1	5	1
B	0	2	0	4	0	4
C	0	0	0	0	0	6



Knuth-Morris-Pratt demo: DFA simulation

A A B A C A A B A B A C A A

pat.charAt(j)

dfa[][j]

	0	1	2	3	4	5
	A	B	A	B	A	C
A	1	1	3	1	5	1
B	0	2	0	4	0	4
C	0	0	0	0	0	6