

44. Wildcard Matching

Hard  3777  165  Add to List  Share

Given an input string (`s`) and a pattern (`p`), implement wildcard pattern matching with support for `'?'` and `'*'` where:

- `'?'` Matches any single character.
- `'*'` Matches any sequence of characters (including the empty sequence).

The matching should cover the **entire** input string (not partial).

`s:` `a d c e b`

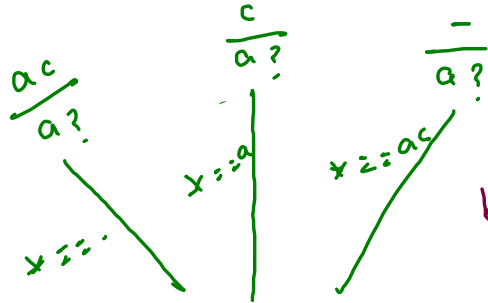
`p:` `a * b`

? → matches any single char

* → any seq. of char

s: b a a b a c

pat: b a * a ?



(i) ac, * a?
s, p

	b	a	a	a	b	a	c	-
b	T	F	F	F	F	F	F	F
a	F	T	T	T	F	F	F	F
*	T	T	T	T	T	T	F	F
a	F	F	F	F	F	T	F	F
?	F	F	F	F	F	F	T	F
-	F	F	F	F	F	F	F	T

$(p(i) == s(j) || p(i) == ?) ?$
 $dp[i][j] = dp[i+1][j+1];$

3
 $(p(i) == *) ?$

$dp[i][j] = dp[i+1][j];$

9
 $dp[i][j+1];$

smallest problem

	a	b	a	c	—
?	T	T	T	T	F
*	T	T	T	T	T
—	F	F	F	F	T

10. Regular Expression Matching

Hard 6877 954 Add to List Share

Given an input string `s` and a pattern `p`, implement regular expression matching with support for `'.'` and `'*'` where:

- `'.'` Matches any single character.
- `'*'` Matches zero or more of the preceding element.

The matching should cover the **entire** input string (not partial).

• \rightarrow matches any single character

* \rightarrow matches zero or more occ- of preceding char.

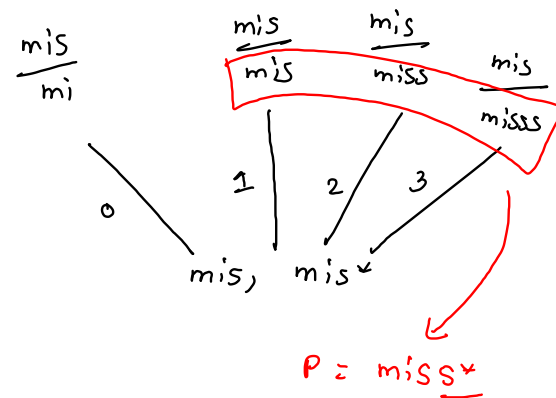
a^+ \rightarrow one or more occ `a, aa, aaa, aaaa`

a^* \rightarrow zero or more occ
`, a, aa, aaa, aaaa`


Input: s = "mississippi", p = "mis*is*p*."
 Output: false

OJO → smallest problem

	-	m	i	s	s	i	s	s	i	p	p	i
m	✓	α	α	α	α	α	α	α	α	α	α	α
i	α	✓	α	α	α	α	α	α	α	α	α	α
s	α	α	✓	α	α	α	α	α	α	α	α	α
*	α	α	✓	✓	✓	α	α	α	α	α	α	α
i	α	α	α	α	α	✓	α	α	α	α	α	α
s	α	α	α	α	α	α	✓	α	α	α	α	α
*	α	α	α	α	α	✓	✓	✓	α	α	α	α
p	α	α	α	α	α	α	α	α	α	α	α	α
.	α	α	α	α	α	α	α	α	α	α	α	α



	-	m	m	m	a	
-	✓	α	α	α	α	
m	α	✓	α	α	α	
*	✓	✓	✓	✓	α	
.	α	✓	✓	✓	✓	



Rod Cutting

$$n = 8$$

extra

	2	5	8	6	8	7	10	2
--	---	---	---	---	---	---	----	---

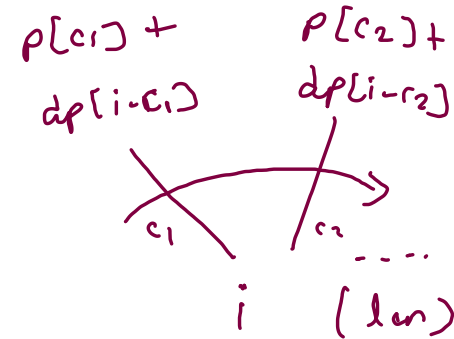
len

1 2 3 4 5 6 7 8

cut strategy

dp

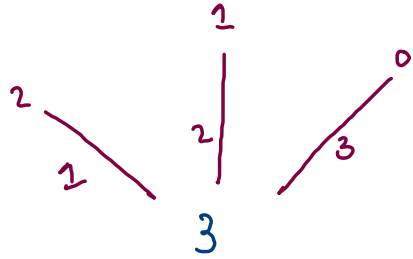
X	2	5	8	10	13	16		
	1	2	3	2-2	2-3	3-3		



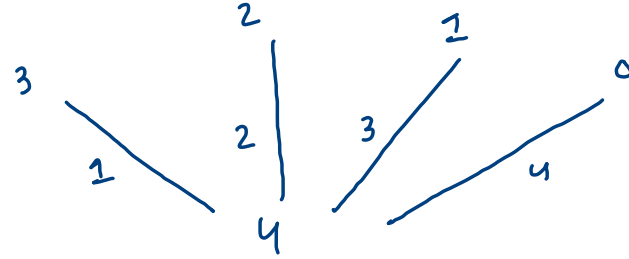
```
for(int i=1; i < dp.length;i++) {
    int max = 0;
    for(int c = 1; c <= i;c++) {
        int pr = p[c] + dp[i - c];
        max = Math.max(pr,max);
    }
    dp[i] = max;
}
```

1 [1]

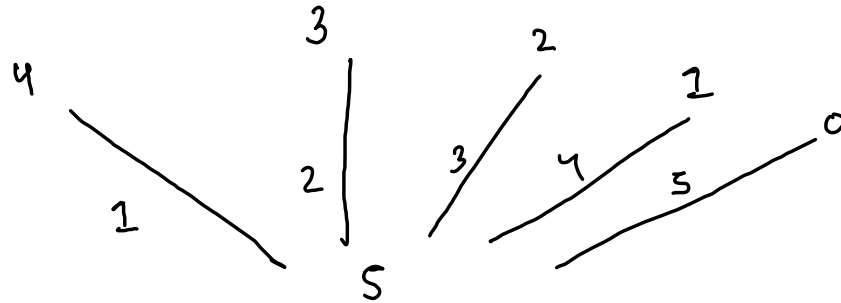
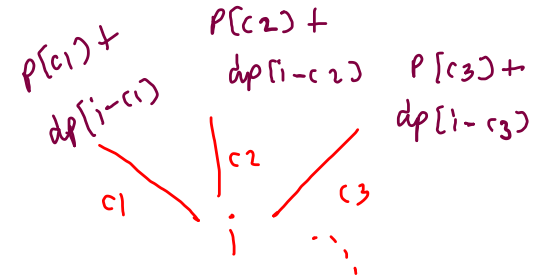
2 [11, 2]



[111, 12, 3]



[1111, 112, 13, 22, 4]



[11111, 1112, 113, 122, 14, 23, 5]

left - right strategy

extra

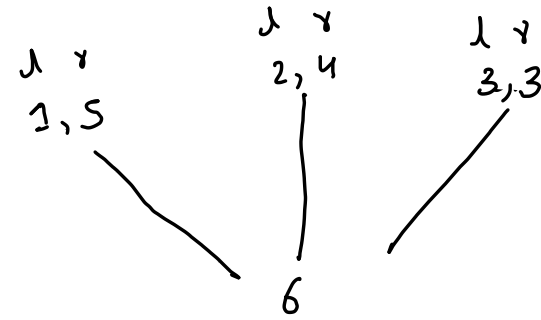
	2	5	8	6	8	7	10	2
--	---	---	---	---	---	---	----	---

len 1 2 3 4 5 6 7 8

dp

X	2	5	8	10	13	16		
	1	2	3	2-2	2-3	3-3		

$dp[1] + dp[8]$



1 [1]

2 [11, 2]

$\frac{1}{-} \times \frac{2}{-}$

|

3

[111, 12, 3]

1×3

|

4

2×2

|

[1111, 112, 13, 22, 4]

1×4

|

5

2×3

|

[11111, 1112, 113, 122, 14, 23, 5]