# 10. Regular Expression Matching\_Hard

Given an input string (s) and a pattern (p), implement regular expression matching with support for '.' and '\*'.

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

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

Note:

s could be empty and contains only lowercase letters a-z.

p could be empty and contains only lowercase letters a-z, and characters like . or \*.

Example 1:

Input:

s = "aa"

p = "a"

Output: false

Explanation: "a" does not match the entire string "aa".

Example 2:

Input:

s = "aa"

p = "a\*"

Output: true

Explanation: '\*' means zero or more of the precedeng element, 'a'. Therefore, by repeating 'a' once, it becomes "aa".

Example 3:

Input:

s = "ab"

p = ".\*"

Output: true

Explanation: ".\*" means "zero or more (\*) of any character (.)".

Example 4:

Input:

s = "aab"

p = "c\*a\*b"

Output: true

Explanation: c can be repeated 0 times, a can be repeated 1 time. Therefore it matches "aab".

Example 5:

Input:

s = "mississippi"

p = "mis\*is\*p\*."

Output: false

## 算法1：

//使用了递归的方法

思路：不懂，真的是很Hard。

class Solution {

public boolean recurse(String s, String p, int posS, int posP){

if(posP==p.length()&& posS==s.length()){

return true;

}

char nChar = (posP+1<p.length())? p.charAt(posP+1):' ';

//We need to check if it's finished or if the nextCharacter is a \*, if it's a \* we skip 2 letters ahead in p

//and check

if((posS>=s.length()||posP>=p.length())&&(nChar!='\*')){

return false;

}

else if((posS>=s.length()||posP>=p.length())&&(nChar=='\*')){

return recurse(s,p,posS,posP+2);

}

char sChar = s.charAt(posS);

char pChar = p.charAt(posP);

if(sChar==pChar){

if(recurse(s,p,posS+1,posP+1)){

return true;

}

}

if(nChar=='\*'){

//if 0 occurences

if(recurse(s,p,posS,posP+2)){

return true;

}

//if 1-all occurences

for(int i=posS;i<=s.length();++i){

//means we skip all characters with \*

if(i==s.length()){

if(recurse(s,p,i,posP+2)){

return true;

}

}

else{

//keep going until not a valid character

if((pChar=='.')||s.charAt(i)==pChar){

if(recurse(s,p,i+1,posP+2)){

return true;

}

}else{

break;

}

}

}

}

else if(pChar=='.'){

return recurse(s,p,posS+1,posP+1);

}

return false;

}

public boolean isMatch(String s, String p) {

return recurse(s,p,0,0);

}

}