P50

2的N次方的计算

递归方法:

将2^n=2^(n/2)\*2^(n/2)\*2^(n%2)

int temp=2^(n/2);

int 2^n=temp\*temp\*2^(n%2);

特殊情况：x=0,1,-1;

单独考虑n=-2147483648的情况

if(n==-2147483648){

if(x<1&&x>-1){

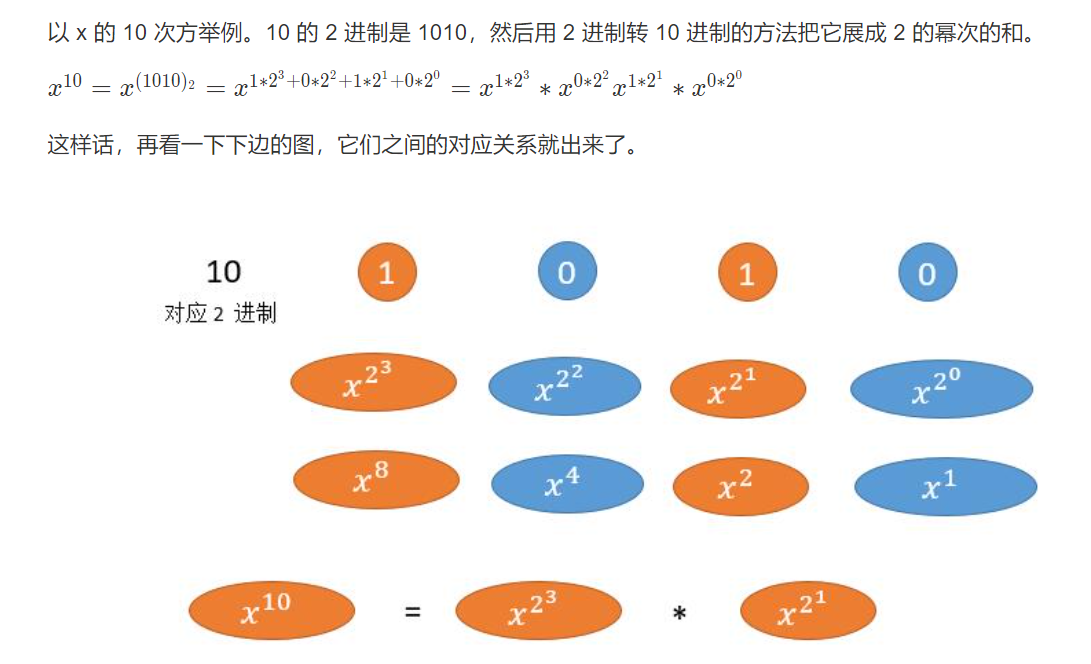
return Double.POSITIVE\_INFINITY;

}

else

return 0;

}



迭代：

private double getPower(double x,int n){

double ans=1;

while(n>0){

if ((n & 1) == 1) {

ans\*=x;

}

x=x\*x;

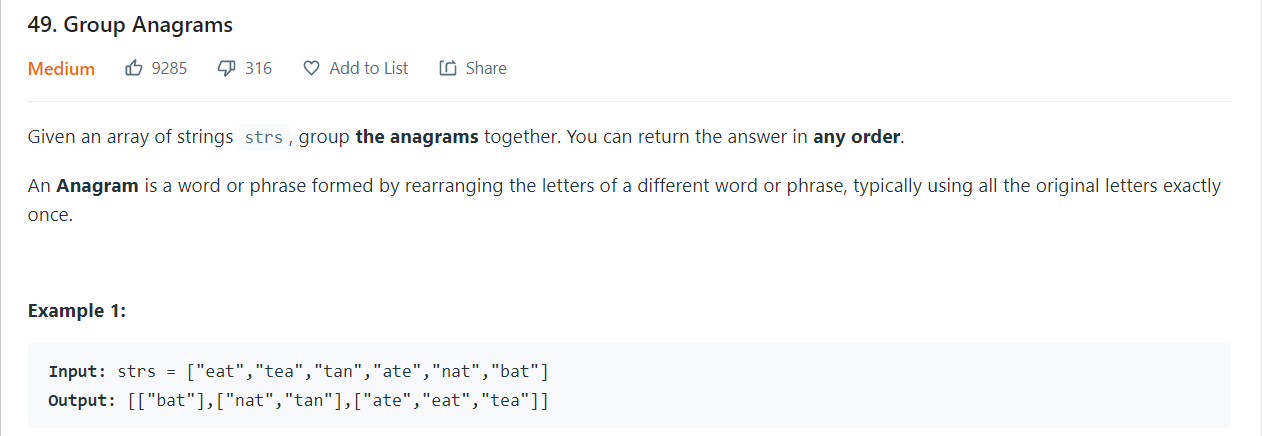
n=n>>1;

}

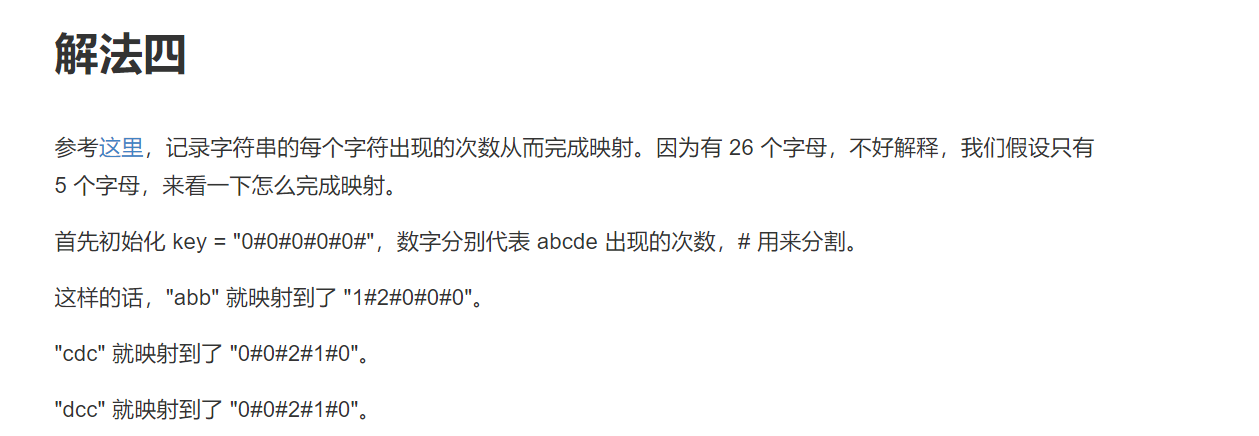
return ans;

}

P49



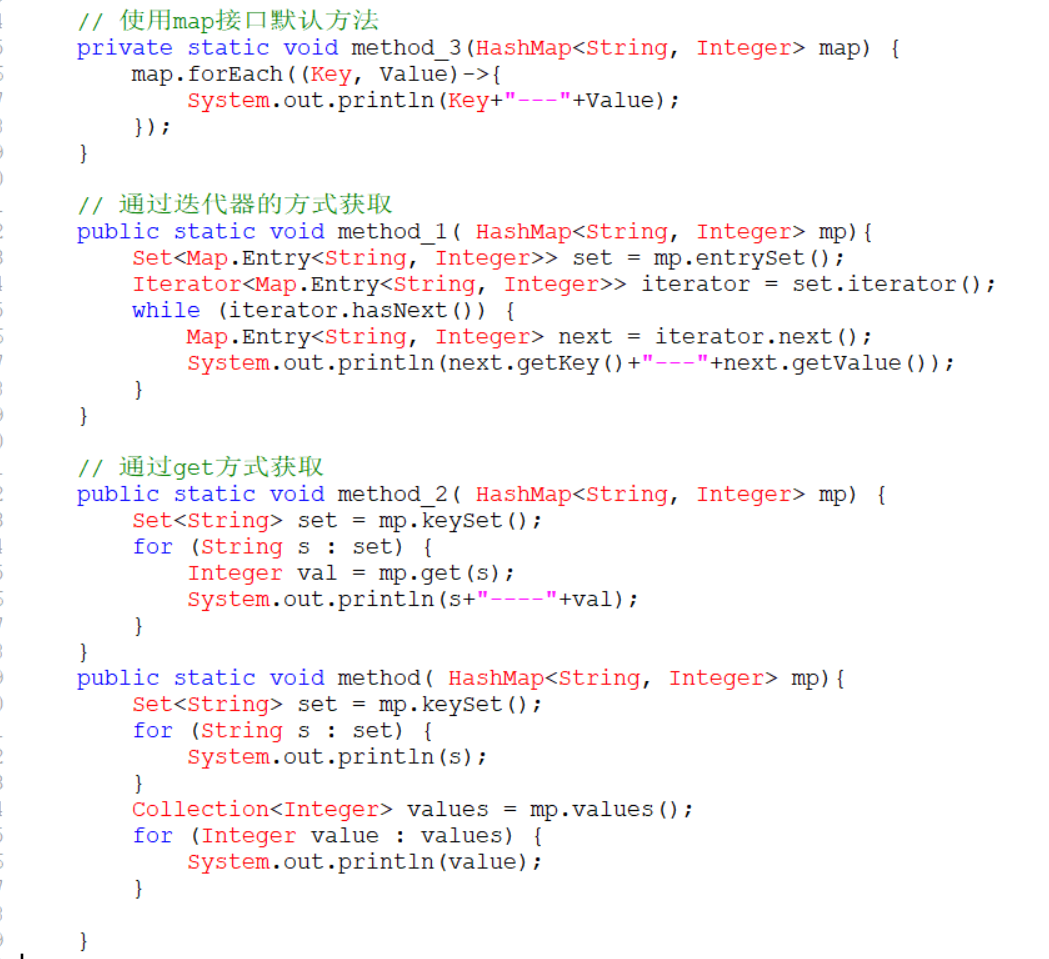
映射的方法。



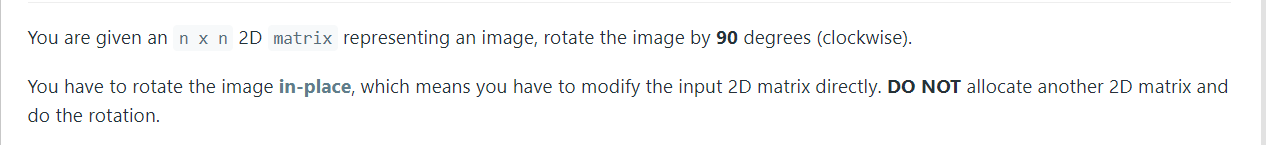
Key的确定。（1）字母排序，（2）字母出现的次数生成key

字符数组转成字符串（1） String.valueOf(charArray) （2）String s1=new String(charArray)

Map遍历key，value。



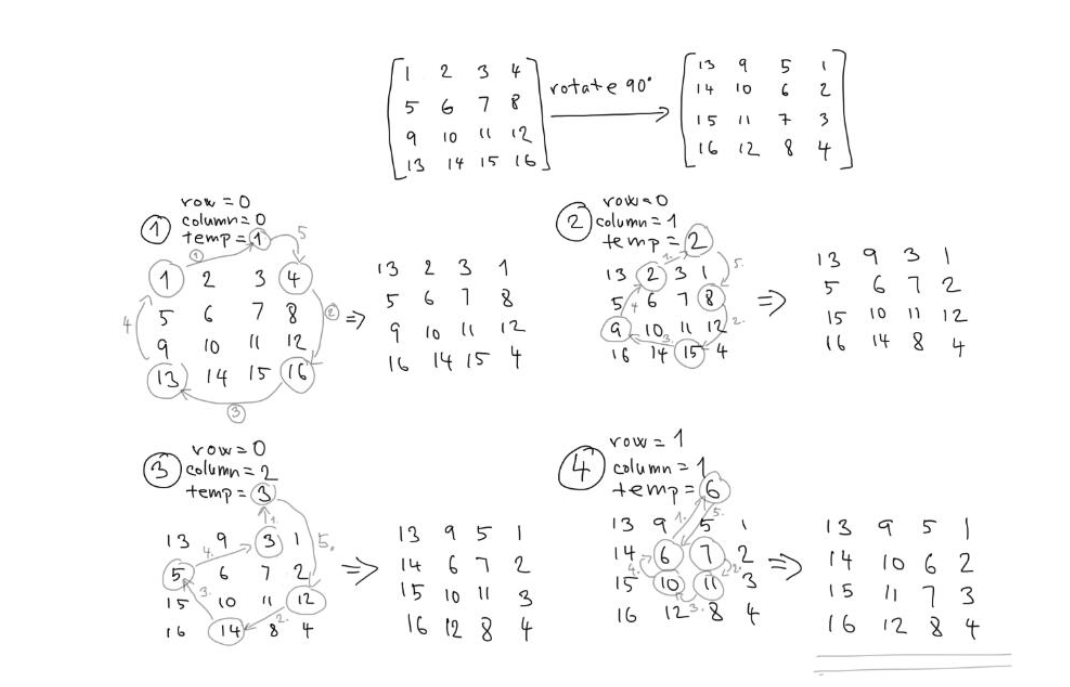
P48

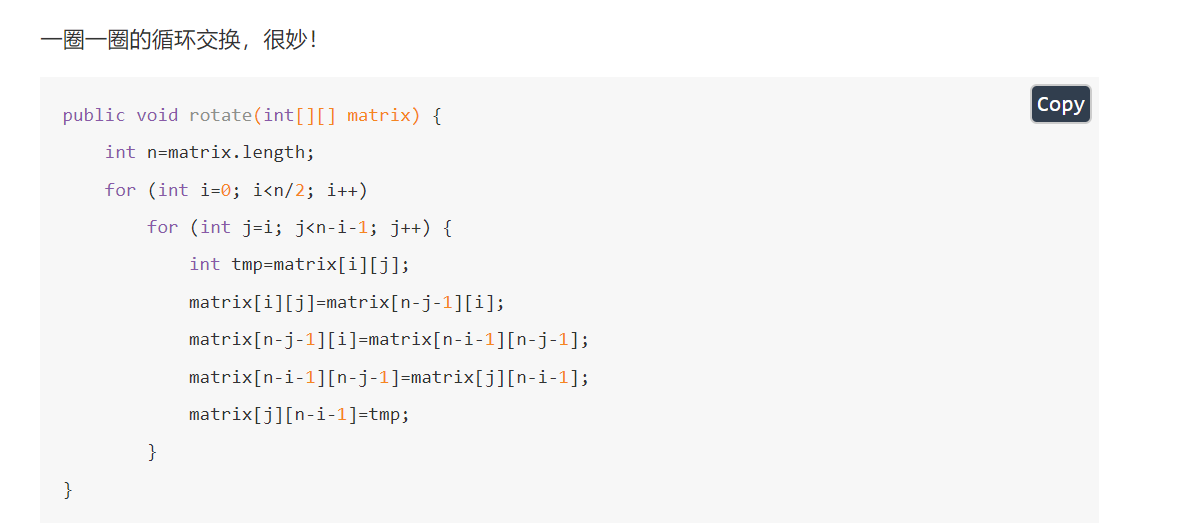


90度旋转一个数组：

方法：（1）先将数组对角线交换（2）对称列交换；

挨个旋转：





P47

Graphical user interface, text, application, email

Description automatically generated

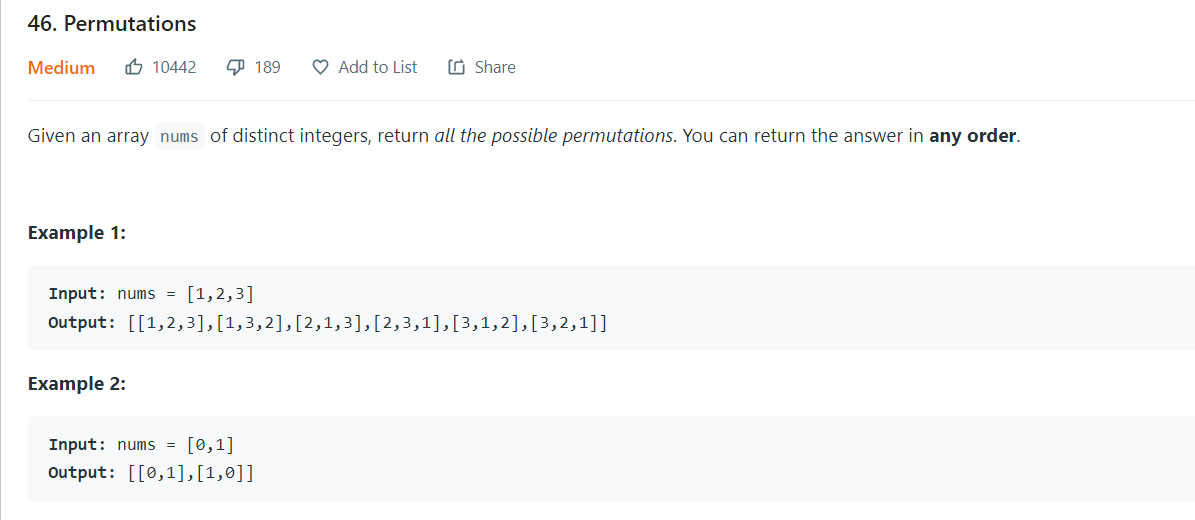
含有重复元素的DFS，使用Stack记录当前加入的元素的index，当当前元素已经存入时，continue。关键技术：（1）首先对数组排序，方便跳过重复元素。（2）跳过重复元素。也就是说，重复元素不可以作为新的开始。重复元素作为新的开始时，此时记录index的Stack为空，也就是说不包含前面的重复元素。

Text

Description automatically generated

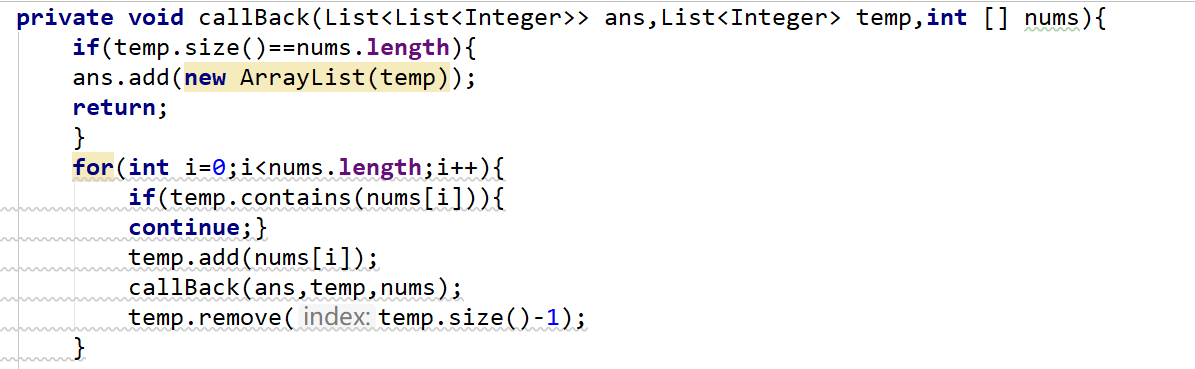
P46

全排列：

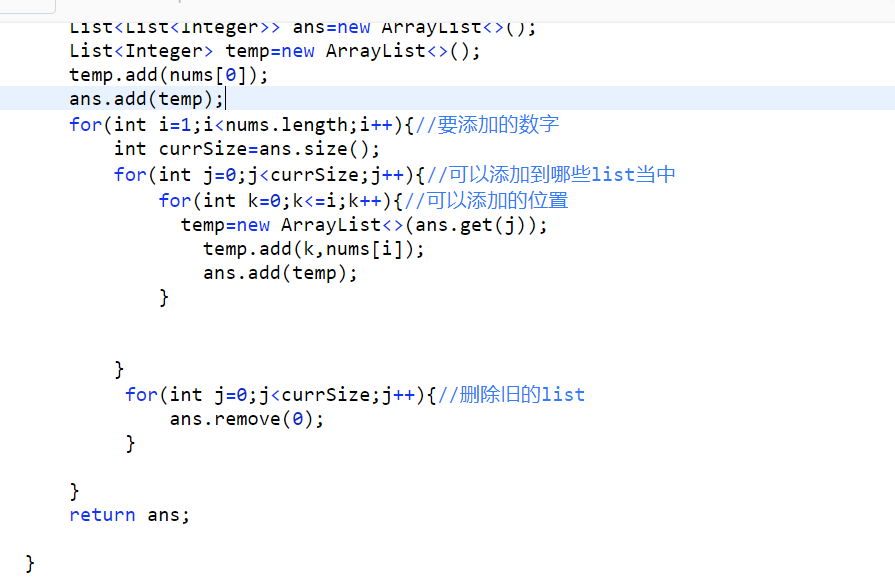


深度优先搜索。

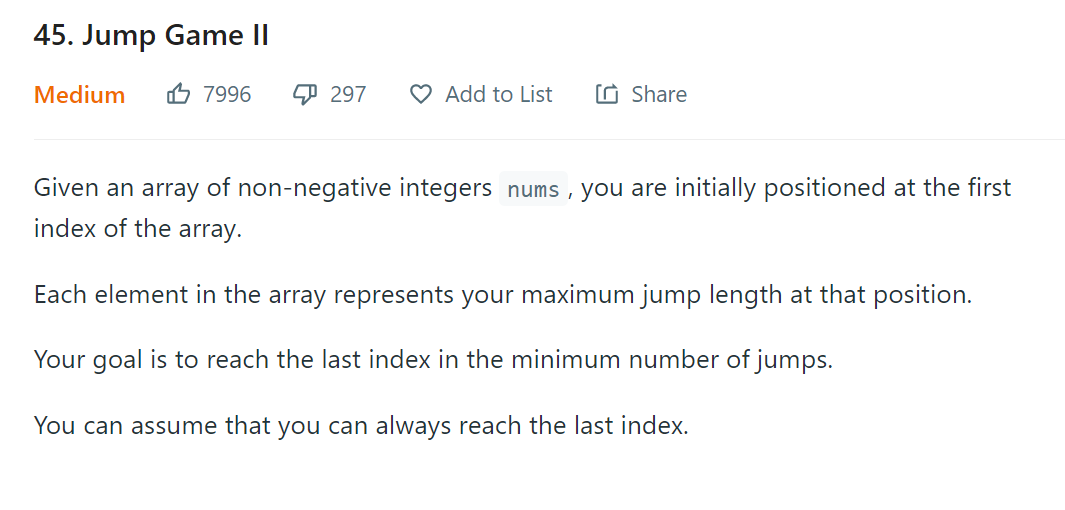
每一个数字作为开始，挨个遍历。



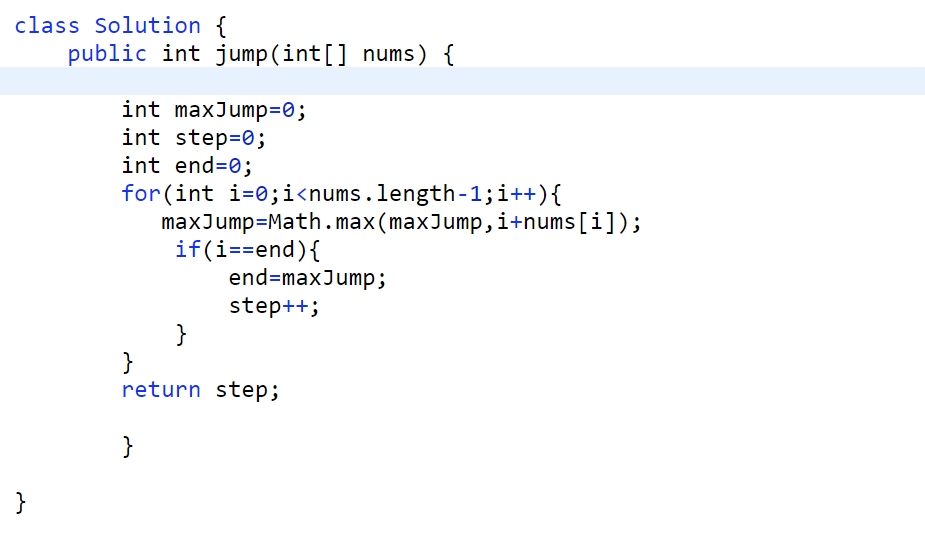
插入法。



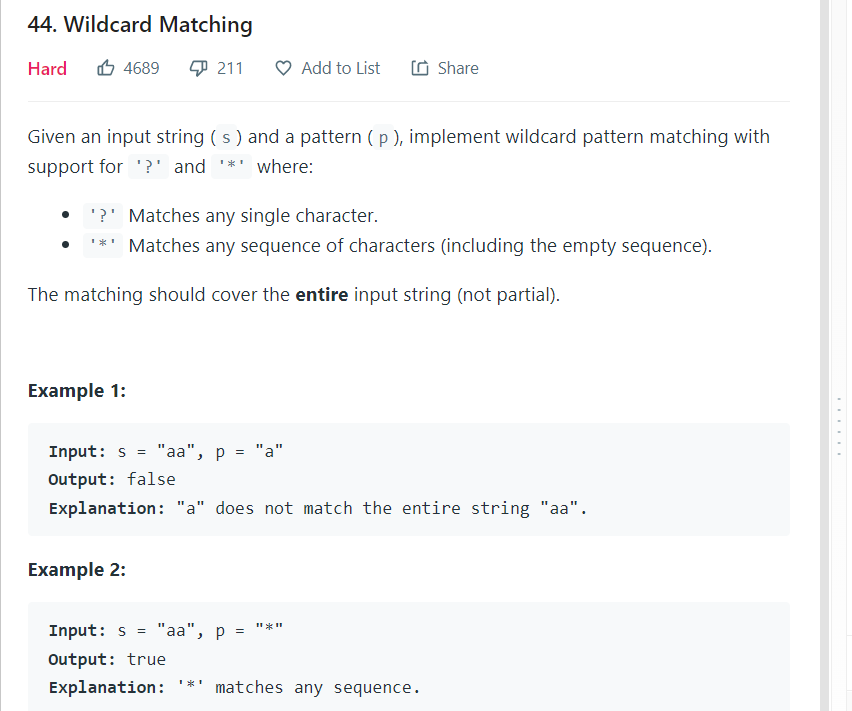
P45



求出每一步的最大跳跃距离。为下一步的边界。当遍历到该长度时，step+1。更新边界。



P44



问题分析：



迭代：当前字符相等或者是‘？’的时候移动到下一个字符。当存在’\*’时，存在不匹配的情况时，p回溯到‘\*’的位置，s向后移动。最后检查p是不是到达最后的位置，没有到达最后的位置时，判定为不匹配。

写代码要细心，随时检查可能出现的问题，越界。条件是否成立。

P43

