题目一：



思路：首先判断整体能否被3整除，如果不能，返回0。如果能被3整除，这从左往右一次读数字，当能整除时，数目加一，temp归零，否则向右移动，temp加上该位的数字

1. **class** Solution:
2. **def** find3(self,nums):
3. **if** nums<0:
4. **return** 0
5. NumToStr = list(str(nums))
6. maxN = 0
7. temp = 0
8. **for** i **in** range(len(NumToStr)):
9. temp = temp \* 10 + int(NumToStr[i])
10. **if** temp % 3 == 0:
11. maxN += 1
12. temp = 0
13. **if** temp % 3 == 0:
14. **return** maxN
15. **else**:
16. **return** 0

19. num = int(input())
20. a = Solution()
21. **print**(a.find3(num))

第二题



思路：暴力解法会超时，暴力解法就是findkth\_2依次加一，找到第k个数。题目中说道了位运算，所以可以考虑使用位运算来结果

加入x为5（二进制为101），如果要符合x+y=x|y，则说明只能在为0的位置进行运算。

当k为1(0b01)时，x|y=0b111，（红色说明是原来的1）

当k为2(0b10)时，x|y=0b1101

当k为3(0b11)是，x|y=0b1111

说明需要将k的二进制插入到x的二进制的为0的位置中。

1. **class** Solution:
2. **def** findkth(self,num,k):
3. numtostr = list(str(bin(num)))[2:]
4. ktostr = list(str(bin(k)))[2:]
5. **for** i **in** range(len(numtostr)-1,-1,-1):
6. **if** numtostr[i] == '0':
7. numtostr[i] = ktostr[-1]
8. ktostr.pop(-1)
9. **if** **not** ktostr:
10. **break**
11. **if** ktostr:
12. numtostr = ktostr + numtostr
13. s = int(''.join(numtostr), 2)
14. **return** s - num

17. **def** findkth\_2(self,num,k):
18. y = 1#返回的y值
19. n = 0#表示次数
20. **while** n <= k:
21. temp = num | y
22. **if** num + y == temp:
23. n += 1
24. **if** n == k:
25. **return** y
26. y += 1
27. **return** y

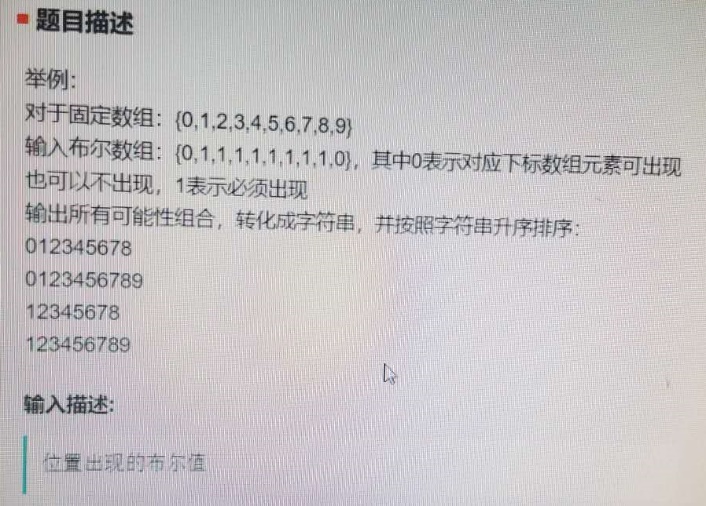
30. a = Solution()
31. # t = int(input())
32. # for i in range(t):
33. #     xandk = list(map(int,input().strip().split()))
34. #     print(a.findkth(xandk[0],xandk[1]))
35. num = 7
36. k = 125156416
37. **print**(a.findkth(num,k))

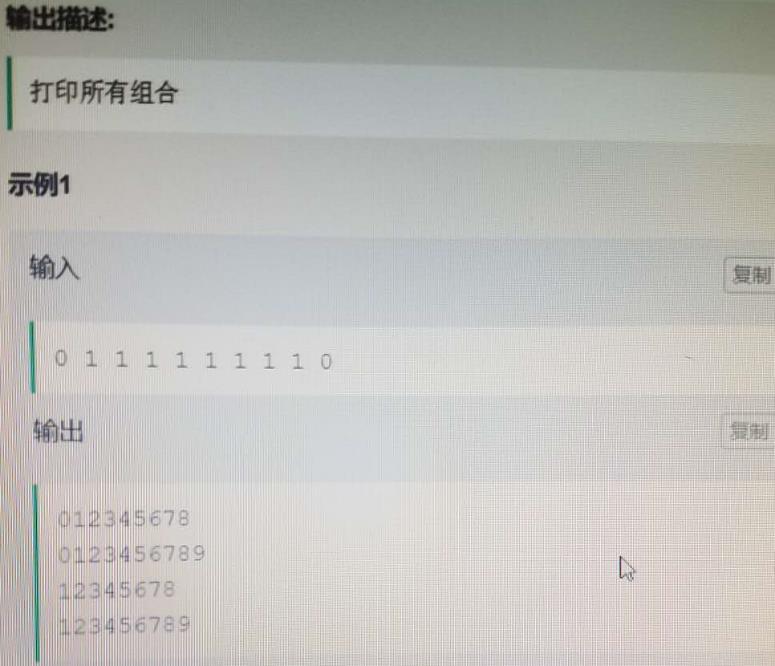
题目3：:输入字符串s1，将s1中的子串s2替换成s3。例如s1=‘this is a chick’，s2=‘is’,s3=’are’，最终输出的结果是’thare are a chick’。

思路：利用Python的find函数，找到出现s2的位置，然后进行替换即可。

1. s1 = 'this is a chick'
2. s2 = 'is'
3. s3 = 'are'
5. res = ''
6. x = s1.find(s2)
7. **while** x != -1:
8. res += s1[:x] + s3
9. s1 = s1[x + len(s2):]
10. x = s1.find(s2)
11. res += s1
12. **print**(res)

题目4：





思路：方法一是我使用了全排列的思路，但貌似复杂了。方法二是网上大佬给的，十分简洁。方法二的思路是，当此时boolset为1时，则让temp\_s = temp\_s+str(i)，当boolset为0时，就有两种情况，让temp\_s = temp\_s+str(i)或者temp\_s 不变。方法一是改变boolset，最后将其变回来。

1. **class** Solution:
2. **def** \_\_init\_\_(self):
3. self.res = []
5. **def** findall(self,nums,boolset):
6. self.findall\_core(nums,boolset,0)
7. **return** self.res
9. **def** findall\_core(self,num,boolset,index):
10. i = index
11. **while** i < len(boolset):
12. **if** boolset[i] != 0:
13. i += 1# 找到下一个为0的位置
14. **else**:
15. **break**
16. **if** i == len(boolset):
17. temp = []
18. **for** i **in** range(len(boolset)):
19. **if** boolset[i]==0:
20. # temp.append(0)
21. **continue**
22. **else**:
23. temp.append(num[i])
24. self.res.append(temp[:])
25. **return**
27. self.findall\_core(num, boolset, i + 1)  # 处理0
28. boolset[i] = 1# 变成1
29. self.findall\_core(num, boolset, i + 1)  # 处理1
30. boolset[i] = 0
32. **class** Solution2:
33. **def** \_\_init\_\_(self):
34. self.res = []
36. **def** findall(self,boolset):
37. self.findall\_core(boolset,0,'')
38. **return** self.res
40. **def** findall\_core(self,boolset,index,temp\_s):
41. **if** index == len(boolset):
42. self.res.append(temp\_s)
43. **return**
44. **if** boolset[index] == 1:
45. self.findall\_core(boolset, index + 1, temp\_s + str(index))
46. **else**:
47. self.findall\_core(boolset, index + 1, temp\_s)
48. self.findall\_core(boolset, index + 1, temp\_s + str(index))
50. nums = [0,1,2,3,4,5,6,7,8,9]
51. boolset = [0,1,1,1,1,1,1,1,1,0]
52. # a = Solution()
53. # b=a.findall(nums,boolset)
54. # b.sort()
55. # print(b)
57. c = Solution2()
58. d = c.findall(boolset)
59. **print**(d)

题目5：最大上升子序列和。即给出一个数组，如[5,1,3,4,9,7,6,8]，找到所有它的上升子序列，如[5,6,8],[1,3,4,7,8]，输出这些子序列的最大和。

思路：利用回溯法，找到所有上升子序列，然后利用sum相加，输出最大值。当网上的大佬提供了另外一种思路，动态规划，这种方法很简洁<https://www.nowcoder.com/discuss/100129>。

1. **class** Solution:
2. **def** \_\_init\_\_(self):
3. self.res = []
4. self.maxN = 0
5. self.length = 0
7. **def** findmaxN(self,lis):
8. self.length = len(lis)
9. temp = []
10. **for** i **in** range(len(lis)):
11. temp.append(lis[i])
12. self.findmaxN\_core(lis,i+1,temp,lis[i])
13. temp.pop()
14. **for** i **in** self.res:
15. temp = sum(i)
16. **if** temp>self.maxN:
17. self.maxN = temp
18. **print**(self.maxN)
20. **def** findmaxN\_core(self,lis,index,temp,cur\_max):
21. **if** index == self.length:
22. self.res.append(temp[:])
23. **return**
24. **for** i **in** range(index,self.length):
25. **if** lis[i]>cur\_max:
26. cur\_max = lis[i]
27. temp.append(lis[i])
28. self.findmaxN\_core(lis,i+1,temp,cur\_max)
29. temp.pop(-1)
30. cur\_max = temp[-1]
31. **if** i == self.length - 1:
32. self.res.append(temp[:])
33. **return**
35. # lis = list(map(int,input().strip().split()))
36. a = Solution()
37. lis = [5,1,3,4,9,7,6,8]
38. a.findmaxN(lis)

题目6：n 面筛子，m面有奖，有奖继续掷筛子，没奖结束。输入给定每面分数的数组 s, len(s) = n， 求期望。

思路：<http://www.cnblogs.com/hgc-bky/p/9553202.html>