**算法题**

有n个数，每个数出现三次，其中有一个只出现两次，如何用O(1)空间找出这个数？如果是随机删掉两个数（删掉的两个数可能相同也可能不同），找出被删掉的两个数？请写出代码。

*# 找到只出现两次的数*

**import** random

**def** selection\_sort(lst): *#选择排序，空间复杂度为O(1)*

**for** i **in** range(len(lst)**-**1):

min\_idx **=** i

**for** j **in** range(i**+**1, len(lst)):

**if** lst[min\_idx] **>** lst[j]:

min\_idx **=** j

lst[i], lst[min\_idx] **=** lst[min\_idx], lst[i]

**return** lst

**def** twiceNumber(lst):

nums**=**selection\_sort(lst)

i **=** 0

**while** i **+**2 **<** len(nums):

**if** nums[i] **==** nums[i**+**2]:

i **+=** 3

**else**:

**return** nums[i]

**return** nums[**-**1]

nums**=**[1,2,1,1,2,3,3,3]

print(twiceNumber(nums))

nums2**=**[1,1,1,2,2,3,3,3]

print(twiceNumber(nums2))

nums3**=**[1,2,1,1,2,3,2,3,3,5,5]

print(twiceNumber(nums3))

*# 找到删除的两个数*

**def** findDel(copy\_nums,nums):

ori\_nums**=**selection\_sort(copy\_nums) *#删除前的数组排序*

new\_nums**=**selection\_sort(nums) *#删除后的数组排序*

lst\_find**=**[]

i,j**=**0,0

**while** i**<**len(ori\_nums):

**if** j**>=**len(ori\_nums)**-**2: *#new\_nums越界*

lst\_find**.**append(ori\_nums[i])

**elif** ori\_nums[i]**!=**new\_nums[j]:

lst\_find**.**append(ori\_nums[i])

j**-=**1

i**+=**1

j**+=**1

**return** lst\_find

nums**=**[1,2,1,1,2,3,3,3]

*# 随机删除两个*

copy\_nums**=**list**.**copy(nums)

lst\_del**=**random**.**sample(nums, 2)

**for** x **in** lst\_del:

nums**.**remove(x)

findDel(copy\_nums,nums)

如何判断平面上两条线段是否相交？给定每条线段的两个端点的坐标。请写出代码。

**import** numpy **as** np

**def** intersect(lst\_points):

*# 由点斜式得出一般式(y2-y1)x+(x1-x2)y=x1y2-x2y1*

A**=**[] *#系数矩阵*

B**=**[] *#常数项矩阵*

**for** points **in** lst\_points:

x1,x2,y1,y2**=**points[0][0],points[1][0],points[0][1],points[1][1]

a1**=**y2**-**y1

a2**=**x1**-**x2

b**=**x1**\***y2**-**x2**\***y1

A**.**append([a1,a2])

B**.**append(b)

x,y **=** np**.**linalg**.**solve(A,B) *#求解方程组，得直线交点*

*# 在每条线段的两个点中间*

count**=**0 *#交点的x是否在线段端点的x范围内，满足一条则为1*

**for** points **in** lst\_points:

x1,x2,y1,y2**=**points[0][0],points[1][0],points[0][1],points[1][1]

**if** min(x1,x2)**<=**x **and** x**<=**max(x1,x2):

count**+=**1

**if** count**==**2:

print('有交点')

**else**:

print('没交点')

*# 不相交*

lst\_points1**=**[

[[1,1],[2,2]],

[[0,3],[1,2]]

]

*# 相交于中间*

lst\_points2**=**[

[[0,4],[2,0]],

[[0,2],[4,0]]

]

*# 相交于端点*

lst\_points3**=**[

[[0,2],[1,3]],

[[0,2],[1,1]]

]

intersect(lst\_points1)

intersect(lst\_points2)

intersect(lst\_points3)

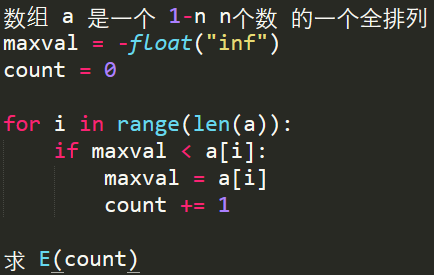
如何判断平面上两个多边形是否有重叠区域？多边形以逆时针方向的坐标序列给出，如(x0, y0), (x1, y1), …, (xn, yn)。 请文字描述算法。

* 如果两个多边形重叠，则至少有一个交点，因此就变成了第二题求交点的问题。对多边形端点进行遍历，每相邻的两个顶点构成一个线段，判断每一条线段是否与另一个多边形相交。

设计一个128位乘128位整数的算法，假设机器位宽是64位。

* 每一个128位的整数用两个64位的表示，如X∗Y=AB∗CD=(A0+0B)∗(C0+0D)X∗Y=AB∗CD=(A0+0B)∗(C0+0D)，其中X和Y是128位的，ABCD是64位的，0表示64位0。 再化简得A0∗C0+A0∗0D+0B∗C0+0B∗0DA0∗C0+A0∗0D+0B∗C0+0B∗0D。其中第一项大于264264，需要再拆分，如200位则用4个64位的存储；第四项可以直接计算；第2项和第3项保留低32位，再逻辑左移32位。

计算下面代码 count 的期望值



**global** lst\_allrange

**def** allRange(lst, s**=**''): *#获取全排列*

**if** len(lst) **==** 1:

lst\_allrange**.**append(int(s**+**str(lst[0])))

**return**

**for** i,x **in** enumerate(lst):

allRange(lst[:i] **+** lst[i**+**1:], s**+**str(x))

**def** getexp(n): *#计算期望*

lst\_count**=**[]

**for** lst **in** lst\_allrange:

a**=**[int(x) **for** x **in** str(lst)]

maxval**=-**float("inf")

count**=**0

**for** i **in** range(len(a)):

**if** maxval**<**a[i]:

maxval**=**a[i]

count**+=**1

lst\_count**.**append(count)

**return** sum(lst\_count)**/**len(lst\_count)

**for** n **in** range(3,10):

lst\_allrange**=**[] *# 用于递归传参*

allRange(list(range(1,n**+**1)))

print(n,':',getexp(n))

* 根据结果，找规律可得，期望为1+12+13+14+...+1n1+12+13+14+...+1n，调和数列

**基础题**

选择ReLU激活函数，有一定概率会屏蔽部分张量的值，这部分值在计算梯度时出直接丢弃，在反向传播时会影响模型的参数修正吗？在多次迭代训练中，是否一定存在模型参数无法被修正的情况?

* 在x<0时，ReLU的值为0。当出现异常输入x时，损失函数会变得很大，因此在反向传播中可能会有很大的梯度g，就可能会使得b变成一个很小的负数。而当输入为正常值时，ReLU的输入很可能是负数，这样会导致该神经元的参数一直不更新。
* 不一定。神经元死亡后是可以复活的，因为其他的神经元参数的更新会导致下一次的输入发生变化。

Conv(in\_channels=32, out\_channels=16, kernel\_size=3, stride=2, padding=1, bias=True)，计算可学习的参数个数

* out\_channels∗(in\_channels∗kernel\_size2+1)=16∗(32∗9+1)=4624

**AI设计题**

* 设计一个模型，对文章进行分类，输入是文章的文本（字符串），输出是分类的ID，请写出模型代码和loss的设计。文章长度在1万字以内。可以用Tensorflow或PyTorch。（提示：由于文章篇幅比较长，需考虑序列长度所造成的记忆消失或显存不足等问题）

**class** CNN:

**def** \_\_init\_\_(self, filter\_sizes, num\_filters, embedded\_size,

dict\_size, maxlen, label\_num, learning\_rate):

self**.**droput\_rate **=** 0.5

**def** cnn (input\_emb, filter\_sizes, num\_filters):

pooled\_outputs **=** []

**for** i, filter\_size **in** enumerate(filter\_sizes):

filter\_shape **=** [filter\_size, embedded\_size, 1, num\_filters]

W **=** tf**.**Variable(tf**.**truncated\_normal(filter\_shape, stddev**=**0.1), name**=**"W")

b **=** tf**.**Variable(tf**.**constant(0.1, shape**=**[num\_filters]), name**=**"b")

conv **=** tf**.**nn**.**conv2d(

input\_emb,

W,

strides**=**[1, 1, 1, 1],

padding**=**"VALID",

name**=**"conv")

h **=** tf**.**nn**.**relu(tf**.**nn**.**bias\_add(conv, b), name**=**"relu")

word\_num **=** input\_emb**.**shape**.**as\_list()[1]

pooled **=** tf**.**nn**.**max\_pool(

h,

ksize**=**[1, word\_num **-** filter\_size **+** 1, 1, 1],

strides**=**[1, 1, 1, 1],

padding**=**'VALID',

name**=**"pool")

pooled\_outputs**.**append(pooled)

num\_filters\_total **=** num\_filters **\*** len(filter\_sizes)

h\_pool **=** tf**.**concat(pooled\_outputs,3)

h\_pool\_flat **=** tf**.**reshape(h\_pool, [**-**1, num\_filters\_total])

cnn\_fea **=** tf**.**nn**.**dropout(h\_pool\_flat, keep\_prob **=** self**.**droput\_rate)

**return** cnn\_fea

self**.**X **=** tf**.**placeholder(tf**.**int32, [**None**, maxlen], name**=**'input\_x')

self**.**Y **=** tf**.**placeholder(tf**.**int64, [**None**])

self**.**encoder\_embeddings **=** tf**.**Variable(tf**.**random\_uniform([dict\_size, embedded\_size], **-**1, 1), trainable**=False**)

encoder\_embedded **=** tf**.**nn**.**embedding\_lookup(self**.**encoder\_embeddings, self**.**X)

*# 由于conv2d需要一个四维的输入数据，因此需要手动添加一个维度。*

encoder\_embedded **=** tf**.**expand\_dims(encoder\_embedded, **-**1) *# shape(encoder\_embedded) = [None, user\_review\_num\*u\_n\_words, embedding\_size, 1]*

outputs **=** cnn(input\_emb **=** encoder\_embedded, filter\_sizes **=** filter\_sizes, num\_filters **=** num\_filters)

self**.**logits **=** keras**.**layers**.**Dense(label\_num, use\_bias**=True**)(outputs)

self**.**probability **=** tf**.**nn**.**softmax(self**.**logits, name**=**'probability')

self**.**cost **=** tf**.**nn**.**sparse\_softmax\_cross\_entropy\_with\_logits(labels **=** self**.**Y, logits **=** self**.**logits)

self**.**cost **=** tf**.**reduce\_mean(self**.**cost)

self**.**optimizer **=** tf**.**train**.**AdamOptimizer(learning\_rate **=** learning\_rate)**.**minimize(self**.**cost)

self**.**pre\_y **=** tf**.**argmax(self**.**logits, 1, name**=**'pre\_y')

correct\_pred **=** tf**.**equal(self**.**pre\_y, self**.**Y)

self**.**accuracy **=** tf**.**reduce\_mean(tf**.**cast(correct\_pred, tf**.**float32))

* 设计一个模型，输出一个段落的k个关键词。请用深度学习的方式实现。

**from** torch **import** nn

**class** RNN(nn**.**Module):

**def** \_\_init\_\_(self, input\_dim, hidden\_dim, layer\_dim, output\_dim): *#layer\_dim: RNN的层数*

self**.**hidden\_dim **=** hidden\_dim *## RNN神经元个数*

self**.**layer\_dim **=** layer\_dim *## RNN的层数*

self**.**rnn **=** nn**.**RNN(input\_dim, hidden\_dim, layer\_dim,

batch\_first**=True**, nonlinearity**=**'relu')

*# 全连阶层*

self**.**fc1 **=** nn**.**Linear(hidden\_dim, output\_dim)

**def** forward(self, x):

out, h\_n **=** self**.**rnn(x, **None**)

out **=** self**.**fc1(out[:, **-**1, :])

**return** out