# phone

Given two arrays of numbers in descending order, write a function that returns a single array sorted in the same order.

E.g.

array1: 4, 2, 1

array2: 7, 6, 5, 3

Result array should be: 7,6,5,4,3,2,1

def merge(array1,array2):

# egde case

if array1 is None and array2 is None:

return []

if len(array1) == 0 and len(array2) == 0:

return []

if array1 is None or len(array1) == 0:

return array2

if array2 is None or len(array2) == 0:

return array1

# main part

array = []

i = 0

len1 =len(array1)

j = 0

len2 =len(array2)

while i<len1 and j<len2:

if array1[i]>=array2[j]:

array.append(array[i])

i+=1

elif array1[i]<array2[j]:

j+=1

if i<len1:

array.extend(array1[i:])

elif j<len2:

array.extend(array2[j:])

return array

# array = [array1,array2,array3,...]

import heapq

def merge\_n(array):

# edge case:

# main case

heap = []

heapq.heap

for i in range(array):

Write a function that will return an array of integers that occur exactly once in a given array of integers.

[1,2,2,1] ->[]

[1,2,3] -> [1,2,3]

def arr(array):

# edge case

if array is None or len(array)==0:

return []

# main part

hash = {}->[]

result = []

for i in range(array):

if hash.has\_key(array[i]):

hash[array[i]].append(i)

else:

hash[array[i]] = [i]

for item in hash:

if len(item[1]) == 1:

result.append((item[0],item[1]))

result = sorted(result,key=lambda x:x[1])

return [x[0] for x in result]

if \_\_name\_\_ =="\_\_main\_\_":

array =

# on-site

person 1:

1: do you know dependency?

2: do you know cron job？

3：I give you a list of process, like 500 process which need to kick out. how you can kick them at one time

4: give you there strings: “abcdefg”,”ab”,”12”. In the first string, replace second string with third string. so the result is “ab12efg”.

person 2:

there is a graph, in the graph, each dependency point to other dependencies. Given any dependency and return an array with all the dependencies it point to to the end. if there is circle, throw exception.

person 3:

1: given a several of base nodes and a target node in coordinate system. return m nodes which distance to target node is the smallest.

A = [(x1,y1),(x2,y2),(x3,y3)...(xn,yn)]

import heapq

def ~(A,(x,y)):

q = []

q = heapq.heapify(q)

for i in range(0,n):

heapq.heappush(q, distance((A[i][0],A[i][1]),(x,y)))

result = []

for i in range(0,m):

result.append(heapq.heappop(q))

return result

def distance((x1,y1),(x,y)):

return sqrt((x1-x)\*\*2+(y1-y)\*\*2)

2: there are several houses in a line, and you need to color them in one of these colors. two connected house should not be have the same color. Given a matrix with the cost for each house painted in each color. after all the houses colored, return the minimum total cost.

C = [color1, color2, color3]

H = [house1, house2, house3….]

M = cost\_matrix =

|  |  |  |  |
| --- | --- | --- | --- |
|  | color1 | color2 | color3 |
| house1 |  |  |  |
| house2 |  |  |  |
| house3 |  |  |  |

这里n暂时看成是从(1-n)

f[x][y]看成是x个房子，第x个房子刷成y色的总开销

f[n] = min(f[n][0], f[n][1], f[n][2])

f[n][0] = M[n][0] +min(f[n-1][1], f[n-1][2])

f[n][1] = M[n][1] +min(f[n-1][2], f[n-1][0])

f[n][2] = M[n][2] +min(f[n-1][1], f[n-1][0])

def ~~(M):

edge case

f = [[0 for i n range(len(C))] for j in range(len(H))]

for i in range(len(C)):

f[0][i] = M[0][i]

for i in range(len(H)):

f[i-1][0] = M[i-1][0] +min(f[i-2][1], f[i-2][2])

f[i-1][1] = M[i-1][1] +min(f[i-2][2], f[i-2][0])

f[i-1][2] = M[i-1][2] +min(f[i-2][1], f[i-2][0])

f[n-1] = min(f[n-1][0], f[n-1][1], f[n-1][2])

return f[n-1]