1.0\_1sort(list a)

left=0

right=len(a)-1

while left<right do

while a[left]==0 and left<right

left=left+1

while a[right]==1 and left<right

right=right-1

if left<right then

exchange a[left] and a[right]

Final: The list a which only contains 0 and 1 is sorted

Because this algorithm is just making a scan for the total array. So the running time is just O(n).

2.

Code:

def initCount(a,n,k):

length=n

b=list()

output=list()

for index in range(0,k):

b.insert(index,0) #initial the counting array

for index in range(0,length):

output.insert(index,0) #initial the output array

for index in range(0,length):

i=a[index]

b[i]=b[i]+1 #count the number of every item in a[0] by counting array

for index in range(1,len(b)):

b[index]=b[index-1]+b[index] #operations in determining the position of item in output[]

print(b)

for index in range(length-1,-1,-1):

i=a[index]

b[i]=b[i]-1

output[b[i]]=i #insert the item in a[] in to output[] with right position to make it sorted

print(output)

a=[7,1,3,1,2,4,5,7,2,4,3]

initCount(a,len(a),len(a)\*2)

Every step of counting array:

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

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

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

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

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

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

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

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

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

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

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

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

[0, 2, 4, 2, 2, 1, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

[0, 2, 4, 6, 2, 1, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

[0, 2, 4, 6, 8, 1, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

[0, 2, 4, 6, 8, 9, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

[0, 2, 4, 6, 8, 9, 9, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

[0, 2, 4, 6, 8, 9, 9, 11, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

[0, 2, 4, 6, 8, 9, 9, 11, 11, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

[0, 2, 4, 6, 8, 9, 9, 11, 11, 11, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

[0, 2, 4, 6, 8, 9, 9, 11, 11, 11, 11, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

[0, 2, 4, 6, 8, 9, 9, 11, 11, 11, 11, 11, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

[0, 2, 4, 6, 8, 9, 9, 11, 11, 11, 11, 11, 11, 0, 0, 0, 0, 0, 0, 0, 0, 0]

[0, 2, 4, 6, 8, 9, 9, 11, 11, 11, 11, 11, 11, 11, 0, 0, 0, 0, 0, 0, 0, 0]

[0, 2, 4, 6, 8, 9, 9, 11, 11, 11, 11, 11, 11, 11, 11, 0, 0, 0, 0, 0, 0, 0]

[0, 2, 4, 6, 8, 9, 9, 11, 11, 11, 11, 11, 11, 11, 11, 11, 0, 0, 0, 0, 0, 0]

[0, 2, 4, 6, 8, 9, 9, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 0, 0, 0, 0, 0]

[0, 2, 4, 6, 8, 9, 9, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 0, 0, 0, 0]

[0, 2, 4, 6, 8, 9, 9, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 0, 0, 0]

[0, 2, 4, 6, 8, 9, 9, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 0, 0]

[0, 2, 4, 6, 8, 9, 9, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 0]

[0, 2, 4, 6, 8, 9, 9, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11]

Final result: [1, 1, 2, 2, 3, 3, 4, 4, 5, 7, 7]

3.For the array < .78, .13, .16, .62, .39, .20, .89, .53, .71, .42 >,We could set 9 buckets to sort the array. For the No.i item of the array a[i]. The integer output of a[i]/0.1 determines which bucket this item would be inserted. Then we could use insertion sort to sort the item for each buckets. Then we pop out the item in each bucket one by one into the array a[].The array is sorted.

4.Pseudocode:

median(list a)

length=len(a)

if length is odd

return a[length/2]

else

return (a[length/2]+a[length/2-1])/2

getMedian(list a, list b,number n)

if n<=0

return -1

else if n=1

return (a[0]+b[0])/2

else if n=2

return (max(a[1],b[1])+min(a[0]+b[0]))/2

else

m1=median(a,n)

m2=median(b,n)

if m1=m2

return m1

else if m1<m2

return getMedian ( a+n/2, b,n)

else

return getMedian(a,b+n/2,n)

print(getMedian(a,b,len(a)))

Code:

def median(a,length):

if length%2==0:

return (a[length//2]+a[length//2-1])/2

else:

return a[length//2]

def getMedian(a,b,length):

if length<=0:

return -1

elif length==1:

return (a[0]+b[0])/2

elif length==2:

return (max(a[0],b[0])+min(a[1],b[1]))/2

m1=median(a,length)

m2=median(b,length)

print(a)

print(b)

print(m1)

print(m2)

if(m1==m2):

return m2

elif m1<m2:

if length%2==0:

return getMedian(a[length//2:],b[:length//2],length//2)

else:

return getMedian(a[length//2:],b[:length//2+1],length//2+1)

elif m1>m2:

if length%2==0:

return getMedian(a[:length//2],b[length//2:],length//2)

else:

return getMedian(a[:length//2+1],b[length//2:],length//2+1)

a=[8,9,12,20,41]

b=[7,9,10,12,15]

print(getMedian(a,b,len(a)))

input: a=[8,9,12,20,41] b=[7,9,10,12,15]

step1: 12 10 [8, 9, 12, 20, 41] [7, 9, 10, 12, 15]

step2: 9 12 [8, 9, 12] [10, 12, 15]

output: 11.0

5.For the array <19,15,14,6,13,10,2,5,7,12>. It is not a heap because the 8th item 7 is larger than the 4th item 6.

So it is not a heap.

6.An array that is in reverse sorted order is a heap. Suppose the array has N items. Because for any item in this array whose index i<=N/2, we are sure that the child of this item whose index are 2\*i and 2\*i+1(if exists) is less than their parent for the array’s reverse order. So this array is a heap