**演算法1 worse case在**Θ**(nlgn)**以内

**Algorithm:**

notes: {**x**} is defined as the **least integer greater than x**

inversion(A)

merge\_sort(A,p,r,ans) Θ(nlgn)

if p<r

q= {(p+r)/2}

merge\_sort(A,p,q,ans)

merge\_sort(A,q+1,r,ans)

merge(A,p,q,r,ans)

merge(A,p,q,r,ans)

n1=q-p+1

n2=r-q

create array L[1,2,3…n1+1] and R[1,2,3…n2+1]

for i=1 to n1

L[i]=A[p+i-1]

for i=1 to n2

R[i]=A[q+j]

i=1;j=1;L[n1+1]=∞；R[n2+1]=∞;

count=length(L)-1

for k=p to r

if L[i]<=R[j]

A[k]=L[i]

i=i+1

count=count -1 <Θ(n)

else

A[k]=R[j]

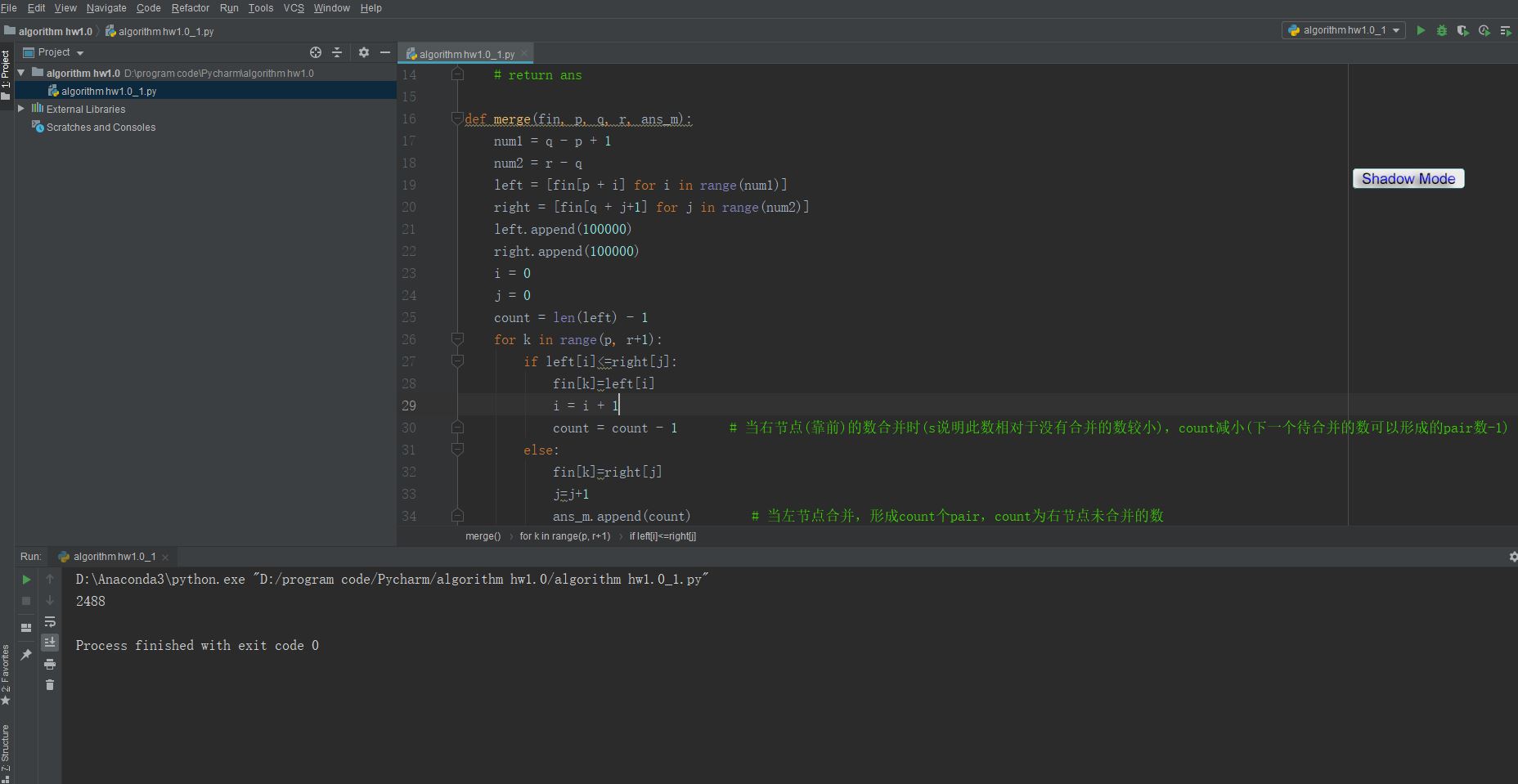
j=j+1

add count at the end of ans Θ(n)

**analyse:**

time complexity:

the worse case of merge sort is Θ(nlgn),(process of proof is in the algorithm 3th,page34-37) and only add some count less than Θ(n) ,so the algorithm’s time complexity is **Θ(nlgn)**

**screen shot:**

**演算法2，worse case在Ω(nlgn)**以内，这里代码比较复杂暂时没有实现

**Algorithm:**

Inversion(A,n)

Create B as [1,2,3,4…,n]

Quick\_sort(A,1,n,B)

for i=1 to n

If B[i]>i

ans=ans+B[i]-i

Return ans

Quick\_sort(A,p,r,B)

while p>r

q=Partition\_pro(A,p,r,B)

if q - p < r - q:

Quick\_sort(A,p,q-1,B)

p = q + 1

else:

Quick\_sort(A,q+1,r,B)

r = q - 1

Partition\_pro(A,p,r,B)

let k=median\_of\_median(A,p,r)

exchange A[k] with A[r]

exchange B[k] with B[r]

x=A[r]

i=p-1

for j=p to r-1

if A[j]<=x

i=i+1

exchange A[i] with A[j]

exchange B[i] with B[j]

exchange A[i+1] with A[r]

exchange B[i+1] with B[r]

return i+1

**analyse:**

time complexity:

Partition\_pro’s worse case T[n]= Θ(n), same to the quick\_sort , because there is only one for loop with n times.

so the Quick\_sort’s worse case is **Ω(nlgn)**, using median of median to choose the keyvalue

space complexity:

In the above code, if left part becomes smaller, then we make recursive call for left part. Else for the right part. In worst case (for space), when both parts are of equal sizes in all recursive calls, we use O(Log n) extra space.

**reference：**

Tail Call Elimination

<https://www.geeksforgeeks.org/quicksort-tail-call-optimization-reducing-worst-case-space-log-n/>

Quicksort with median of medians

<https://www.researchgate.net/publication/306258056_Quicksort_with_median_of_medians_is_considered_practical>