Tính độ phức tạp thời gian của các thuật toán sau:

VD: bài 11 là O(n)

list\_grade = [1,2,3,4,5]  
set\_grade = *set*(list\_grade)  
average = *sum*(list\_grade)/*len*(list\_grade)  
*print*(average)  
list\_below\_average = []  
*# This part is to find the elements that are below average and then  
# append them to a new list called list\_below\_average  
for* i *in* list\_grade:  
 *if* i < average:  
 list\_below\_average.append(i)  
*# Iterate all the elements in the list\_below\_average and then  
# remove them from the list\_grade  
for* i *in* list\_below\_average:  
 list\_grade.remove(i)  
  
set\_grade = *set*(list\_grade)  
*print*(set\_grade)

Big O notation

a.

def abc(items):

for item in items:

print(item)

🡪 assume len(items)=n, n belongs Z

The first line takes n unit of time

The second line is a simple statement

But it is tin a forloop that runs n times,

So it should be 1xn

Thus, we have f(n)=n+n=2n

🡪 BigO: O(n)

b. def abc2(items):

for item in items:

print(item)

for item in items:

print(item)

🡪 n

c.

def one(n):

a = 0

for i in range(0,n):

for j in range(n,0,-1):

a = i + j

🡪 n^2

c.

def two(A):

n = len(A) 1

p = random.randint(10,100) logn

count = 0 1

while count <= p: p

print(A[count]) 1xp

count += 1 p

for i in range(n-1,p,-1): n-1-p

print(A[i]) n-1-p

-> f(N)= 1+logn+1+p+p+p+n-1-p+n-1-p=2n+p+logn

🡪O(n+p+logn)

O(n+p) if randint=n

c.

def four (A, B):

n = len(A) n

m= len (B) m

for i in range(0,n): n

for j in range(0,m): mxn

print(k+j) 1xnxm

🡪 mxn

d.

// precondition: A contains only positive numbers

def find\_biggest(the\_list):

…………??

def five(A):

n = len(A)

B = []

max = 0

for i in range(n-1,0,-1):

find\_biggest(A)

B[i] = A[max]

A[max] = -1

return B

🡪 n^2

Giả sử find\_biggest là thuật toán tìm ra element lớn nhất trong 1 list. Hãy làm thử và calculate xem thuật toán đó mất bao nhiêu thời gian?