Problem Set 2(Written Part)

1. 

So,  for n>0

Because , so is big-Theta of .

1. a) def main():

result=0

for i in range (n):

result+=a[i]\*degree(x,i)

def degree(x,i):

if i==0:

return 1

else:

return x\*degree(x,i-1)

The degree function takes big-Oh (n) times because it basically make every x times together. And the main function also takes big-Oh (n) times for there is one for-loop. So the algorithm takes big-Oh(n^2) times.

b) def main():

result=0

for i in range (n):

result+=a[i]\*degree(x,i)

def degree(x,i):

if i==0:

return 1

else:

d=degree(x,i//2)

if i%2==0:

return d\*d

else:

return d\*d\*x

Suppose k is the time regarding ,  .

So the degree function takes bog-Oh (log(n)) times.

The main function takes big-Oh (n) times, so the whole algorithm takes (nlog(n)) times.

c) When i=1, the operations are 2 times;

when i=2, the operations are 4 times;

when i=3, the operations are 6 times.

As we can conclude, the arithmetic operations for i=n is 2n. Prove by induction:

When i=1, as proved above.

Suppose when i=k, the operations is 2k;

When i=k+1, 

Because takes 2k time,  takes 2(k+1) times.

So the number of arithmetic operations is big-Oh (n).

1. , for 

So  is big-Oh of .

1. def arrange(S,start,end,k):

if start<=end:

if S[start]<=k:

arrange(S,start+1,end,k)

else:

S.append(S[start])

S.remove(S[start])

arrange(S,start,end-1,k)

else:

return S

The runtime of this algorithm is big-Oh (n), because it goes over every element of this sequence for one time.

1. n=0

def log(number):

global n

division=number//2

n+=1

if division>=2:

log(division)

else:

print(n)