**1-)**

**a-)**

First for loop executes n-1 times

Second for loop executes n-1 times

Third for loop executes less than n times so time complexity is O(n)

b-)

first if has O(1) comp.

second if has O(2) comp.

temp.add has O(1) comp.

Third if is executed just for first function and it has O(n) complexity

First for has maximum O(n) but it reduces for each after function call because current is increased.

T(n) = T(n-current)+4+n

For n = 0 T(0) = T(current)+4 Correct since current is an integer value .

Lets assume for value of n it’s true.

T(n) = T(n-current) + 4 + n = T(n-current) + n

For n=n+1 T(n+1) = T(n-current+1)+4+n+1 = T(n-current) +n so it’s correct.

Master Theorem: T(n) = T(n-current)+4+n so a=1 b=1 f(n) = 4+n = n so c =1

So c = logba T(n) = Θ(nlogn)

2-)

array = [ 1,2,4,6,9...];

target\_value = x;

for(int i=0, j=size-1 ; i < size || i < j ; ){

if(array[i] + array[j] == x){

numbers are array[i] and array[j]

}else if(array[i]+array[j] < x){

İ++;

}else

j--;

}

For worst case for loop will executes n-1 time because ‘i’ and ‘j’ represents indexes of numbers and i cant be equal or bigger than j and j cant be equal or less than i .So complexity is O(n-1) = O(n).

3-)First for loop 2n times

Second for loop has 2n + (2n-1) + (2n-2) + ... + 1 = 2n! times

Third for loop has log32n + log3(2n-1) + log3(2n-2) + ... + log31 = log32n! times

So complexity 2n\*2n!\*(log32n!) = n\*n!\*(log3n!) O(n\*n!\* (log3n!))

4-)

O(1) if n == 1

T(n) = ( (n/2 – 1) \* (n/2 – 1) ) + T(n/2) + T(n/2) + T(n/2) + T(n/2)

T(n) = ( (n2 – 4n + 1) / 4 ) + 4T(n/2) if n>0

Master Theorem

f(n) = (n2 – 4n + 1) / 4 a=4 b=2 c=2 since f(n) = Θ(n2)

Since c = logba = log24 T(n) = Θ(n2logn)