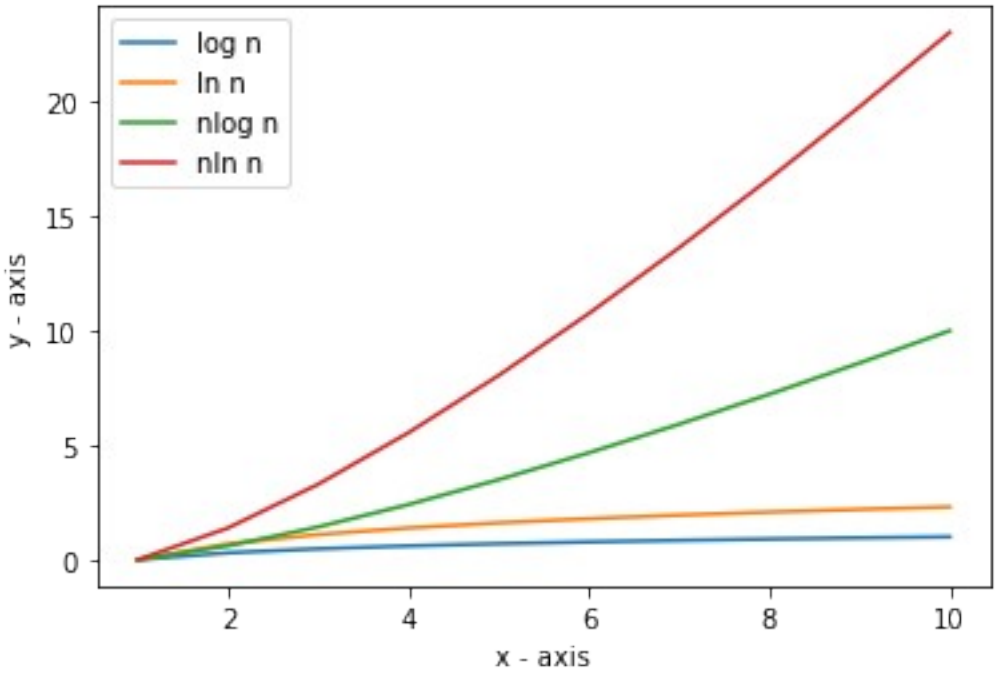
Graphs of functions

```
import math
x1=[]
y1=[]
for i in range(1,11):
    x1.append(i)
    a=math.log10(i)
    y1.append(a)

import matplotlib.pyplot as plt
plt.plot(x1, y1, label = "log n")


x2=[]
y2=[]
for i in range(1,11):
    x2.append(i)
    a=math.log(i)
    y2.append(a)

plt.plot(x2, y2, label = "ln n")

x3=[]
y3=[]
for i in range(1,11):
    x3.append(i)
    a=math.log10(i)
    b=i*a
    y3.append(b)

plt.plot(x3, y3, label = "nlog n")

x4=[]
y4=[]
for i in range(1,11):
    x4.append(i)
    a=math.log(i)
    b=i*a
    y4.append(b)

plt.plot(x4, y4, label = "nln n")



plt.xlabel('x - axis')
plt.ylabel('y - axis')
plt.legend()
```

```
print("nlogn and nlnn are are continuously increasing functions. logn
and lnn are increasing functions")
print("The shapes of the graphs of log n and ln n are similar and the
shapes of the graphs of nlogn and nlnn are similar")
```

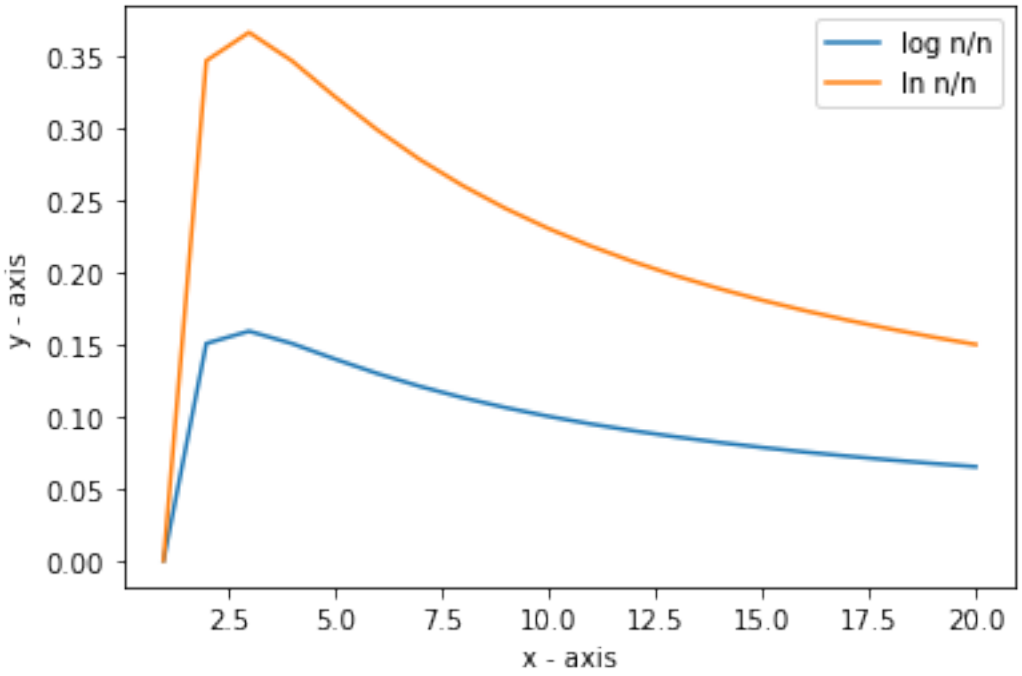
```
nlogn and nlnn are are continuously increasing functions. logn and lnn
are increasing functions
The shapes of the graphs of log n and ln n are similar and the shapes
of the graphs of nlogn and nlnn are similar
```




```
x5=[]
y5=[]
for i in range(1,21):
    x5.append(i)
    a=math.log10(i)
    b=a/i
    y5.append(b)

plt.plot(x5, y5, label = "log n/n")

x6=[]
y6=[]
for i in range(1,21):
    x6.append(i)
    a=math.log(i)
    b=a/i
    y6.append(b)
```

```
plt.plot(x6, y6, label = "ln n/n")


plt.xlabel('x - axis')
plt.ylabel('y - axis')
plt.legend()

print(" Both the functions (logn)/n and (ln n)/ n first increase,
reach a point of maxima and then decrease")
print("The shapes of the graphs of (logn)/n and (ln n)/ n are
similar")
```

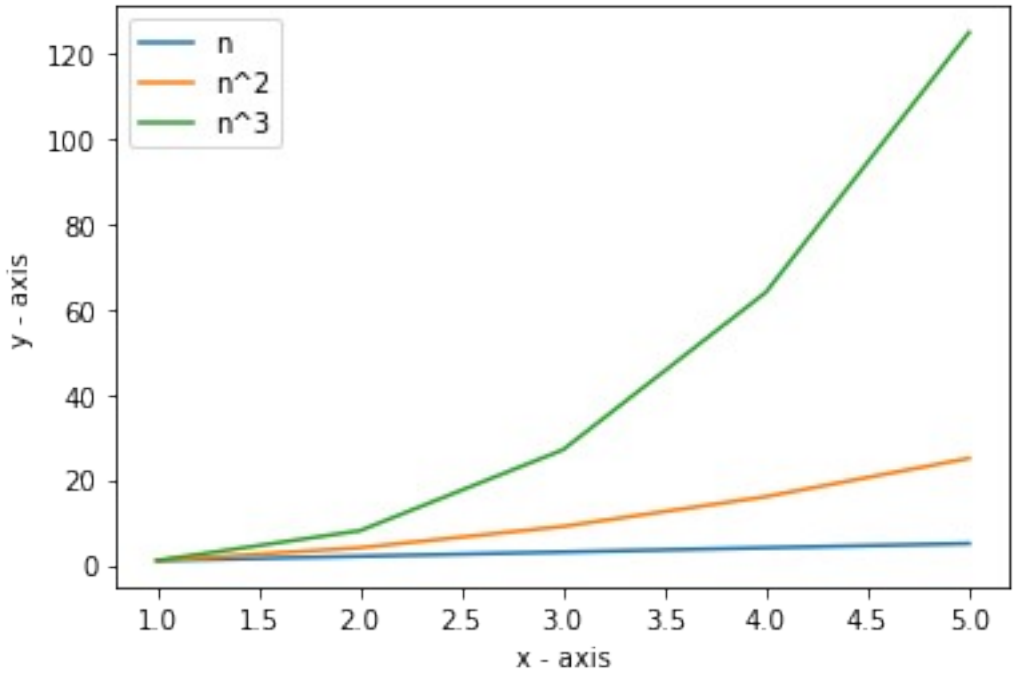
```
 Both the functions (logn)/n and (ln n)/ n first increase, reach a
point of maxima and then decrease
The shapes of the graphs of (logn)/n and (ln n)/ n are similar
```




```
x=[]
y=[]
for i in range(1,6):
    x.append(i)
    y.append(i)

plt.plot(x, y, label = "n")

x=[]
y=[]
for i in range(1,6):
    x.append(i)
```

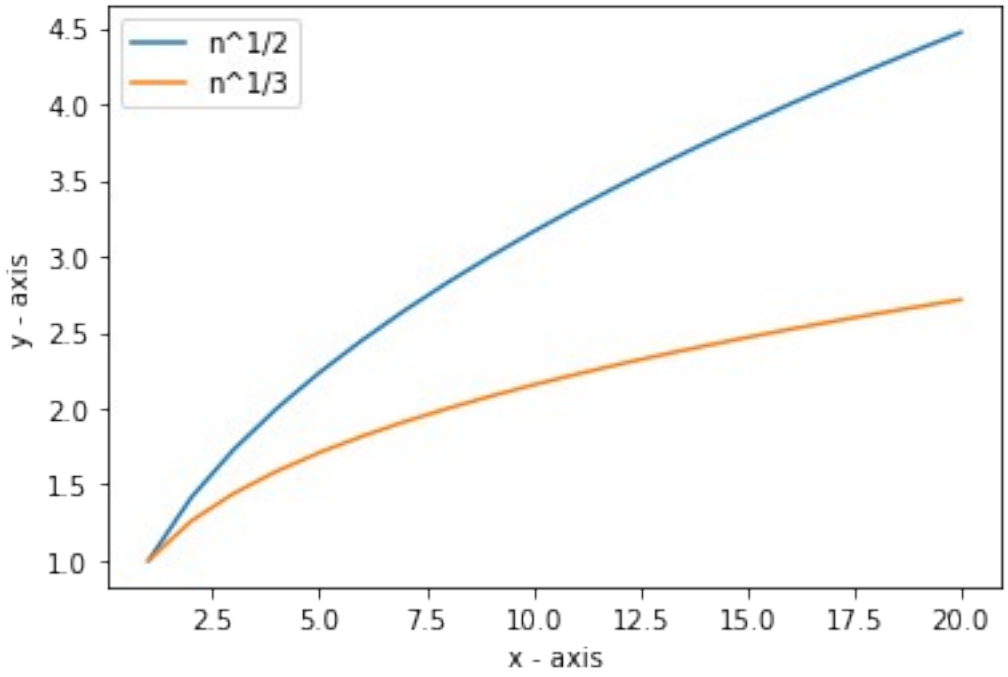
```
    y.append(i*i)

plt.plot(x, y, label = "n^2")

x=[]
y=[]
for i in range(1,6):
    x.append(i)
    y.append(i*i*i)

plt.plot(x, y, label = "n^3")




plt.xlabel('x - axis')
plt.ylabel('y - axis')
plt.legend()

print("the functions n,n^2 and n^3 are continuously functions")
print("The graph of n is a straight line. The graphs of n^2 and n^3
have a similar shape for all positive integers. For all integers >=2
n<n^2<n^3 . They are all equal at n=1")
```

```
the functions n,n^2 and n^3 are continuously functions
The graph of n is a straight line. The graphs of n^2 and n^3 have a
similar shape for all positive integers. For all integers >=2
n<n^2<n^3 . They are all equal at n=1
```



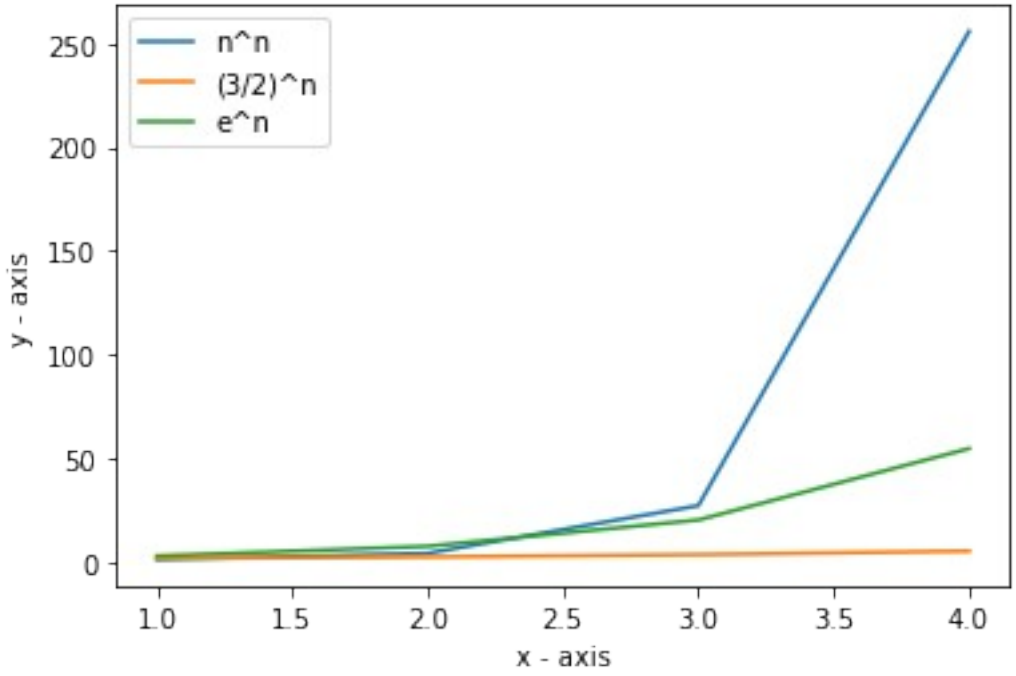

```
x=[]
y=[]
for i in range(1,21):
    x.append(i)
    y.append(i**0.5)

plt.plot(x, y, label = "n^1/2")

x=[]
y=[]
for i in range(1,21):
    x.append(i)
    y.append(i**(1/3))

plt.plot(x, y, label = "n^1/3")




plt.xlabel('x - axis')
plt.ylabel('y - axis')
plt.legend()

print("The functions n^(1/2) and n^1/3 are increasing functions. Their
graphs have a similar shape")
```

```
The functions n^(1/2) and n^1/3 are increasing functions. Their graphs
have a similar shape
```



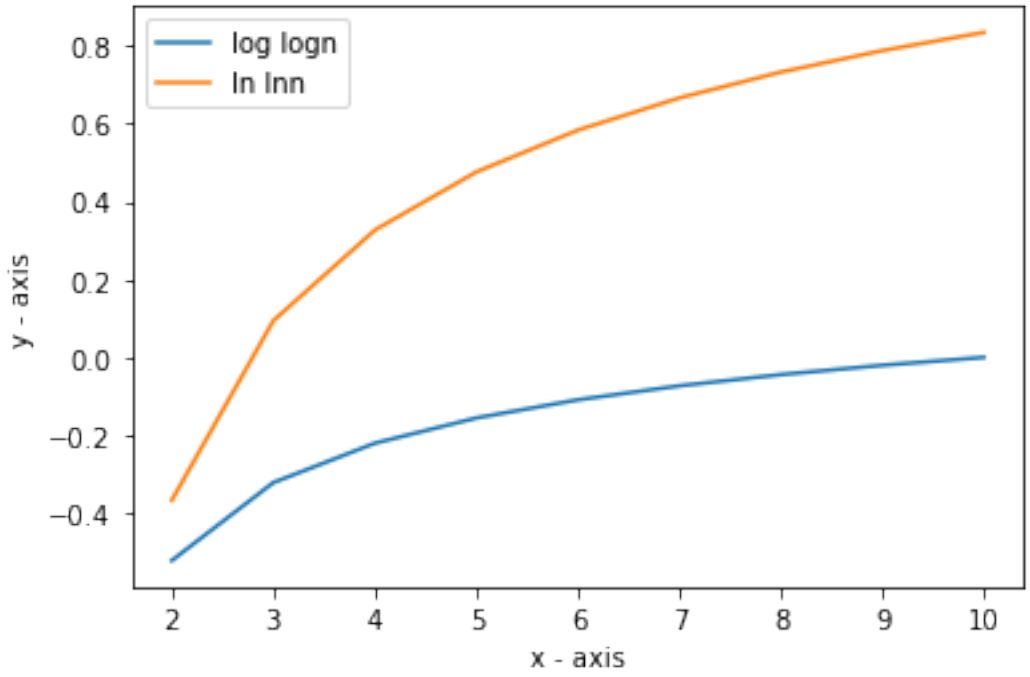

```
x=[]
y=[]
for i in range(1,5):
    x.append(i)
    y.append(i**i)

plt.plot(x, y, label = "n^n")

x=[]
y=[]
for i in range(1,5):
    x.append(i)
    y.append(1.5**i)

plt.plot(x, y, label = "(3/2)^n")

x=[]
y=[]
for i in range(1,5):
    x.append(i)
    y.append(math.exp(i))

plt.plot(x, y, label = "e^n")


plt.xlabel('x - axis')
plt.ylabel('y - axis')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x214ea320a30>
```




```
x=[]
y=[]
for i in range(2,11):
    x.append(i)
    y.append(math.log10(math.log10(i)))

plt.plot(x, y, label = "log logn")

x=[]
y=[]
for i in range(2,11):
    x.append(i)
    y.append(math.log(math.log(i)))

plt.plot(x, y, label = "ln lnn")



plt.xlabel('x - axis')
plt.ylabel('y - axis')
plt.legend()

print("The functions log logn and ln lnn are increasing functions and
their graphs have a similiar shape")
```

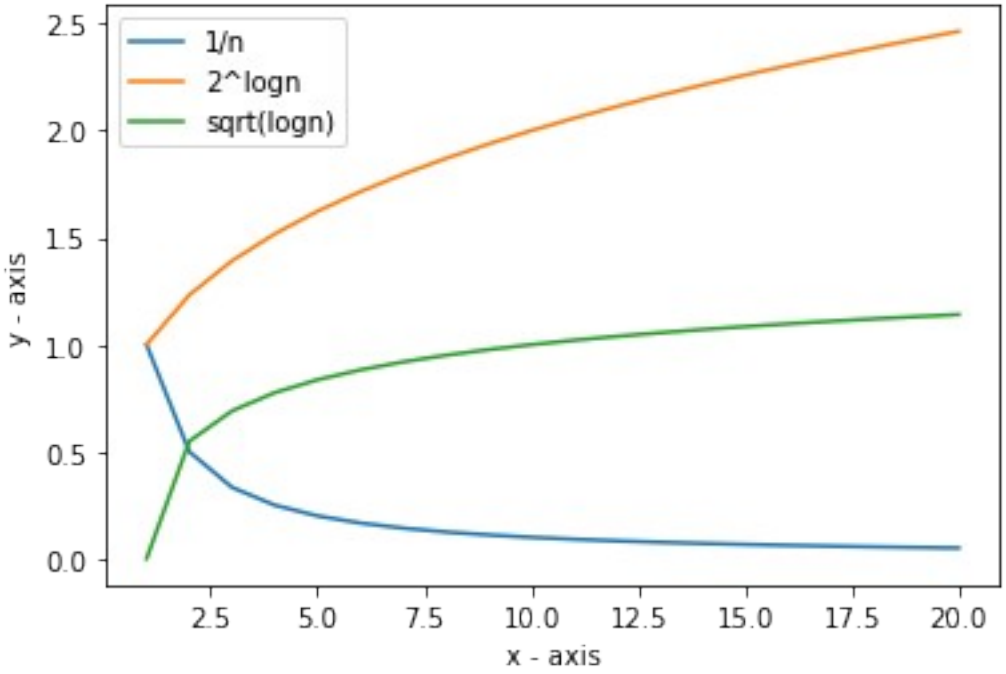
```
The functions log logn and ln lnn are increasing functions and their
graphs have a similiar shape
```




```
x=[]
y=[]
for i in range(1,21):
    x.append(i)
    y.append(1/i)

plt.plot(x, y, label = "1/n")


x=[]
y=[]
for i in range(1,21):
    x.append(i)
    y.append(2**math.log10(i))

plt.plot(x, y, label = "2^logn")

x=[]
y=[]
for i in range(1,21):
    x.append(i)
    y.append(math.log10(i)**0.5)

plt.plot(x, y, label = "sqrt(logn)")

plt.xlabel('x - axis')
plt.ylabel('y - axis')
plt.legend()
```

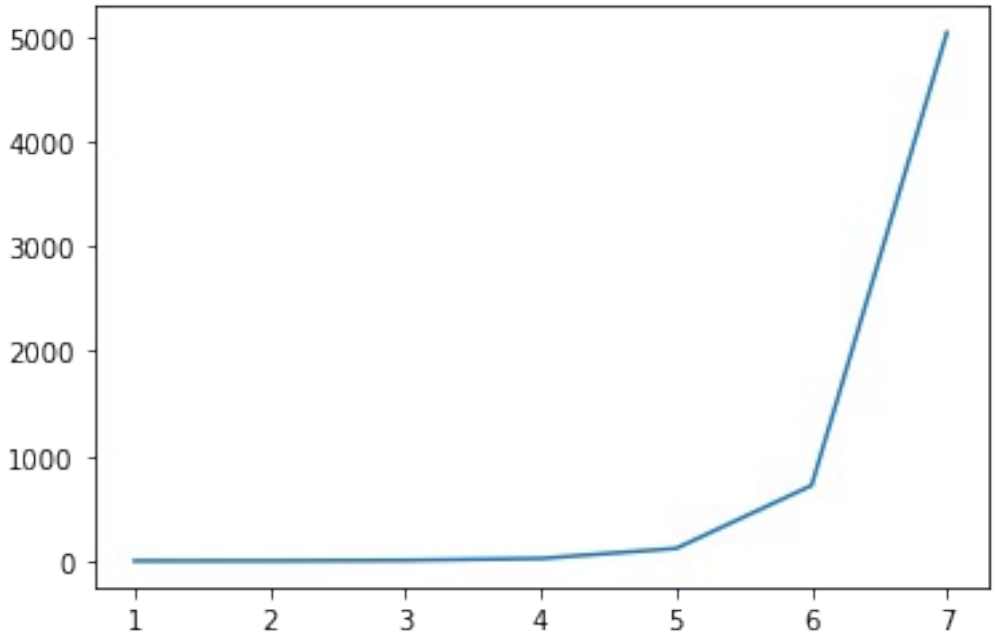
```
print("1/n is a decreasing function")
print("sqrt(logn) and 2^logn are increasing functions")
```

```
<matplotlib.legend.Legend at 0x214eb573400>
```




```
x=[]
y=[]
for i in range(1,8):
    x.append(i)
    y.append(math.factorial(i))

plt.plot(x, y, label = "n!")
print("n! is a strictly increasing function")
```

```
n! is a strictly increasing function
```



Binary search

```
def binarySearch(a,low,high,x):

    if high >= low:

        mid = (high + low) // 2

        if a[mid] == x:
            return mid

        elif a[mid] > x:
            return binarySearch(a, low, mid - 1, x)

        else:
            return binarySearch(a, mid + 1, high, x)
    else:
        return -1
```

Case 1: Object is present

```
a=[1,2,3,4,5,6,7,8,9,10]
x=8
index=binarySearch(a,0,len(a)-1,x)
```

```
if index!=-1:
    print("The object is present at index ",index)

else:
    print("The object is not present")
```

```
The object is present at index  7
```

Case 2: Object is not present

```
x=15
index=binarySearch(a,0,len(a)-1,x)
if index!=-1:
    print("The object is present at index ",index)

else:
    print("The object is not present")
```

```
The object is not present
```

Initilisation: The program is initialised assuming the array is sorted and the object is present in the loop. Maintainence: x must be found between a[low] and a[m−-1]. The n for the recursive call is n = m − 1 − low = ⌊(low+high)/2⌋ − 1 − low. (Note that ⌊x⌋ is the floor of x, which rounds it down toward negative infinity.) If low+high is odd, then n = (low + high − 1)/2 − 1 − low = (high − low)/2 − 1, which is definitely smaller than high−low. If low+high is even then n = (low + high)/2 − 1 − low = (high−low)/2, which is also smaller than k + 1 = high−low because high−low = k + 1 > 0. Termination: The loop is terminated if element is the mid of any sub array in the recursive steps