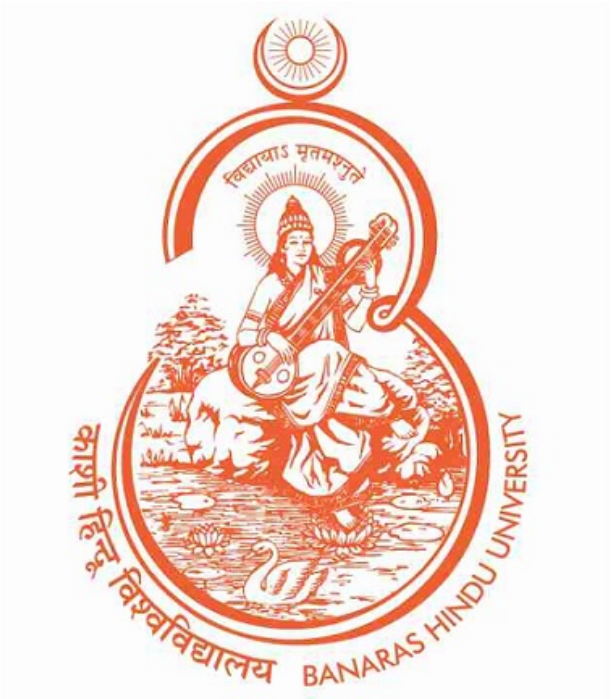


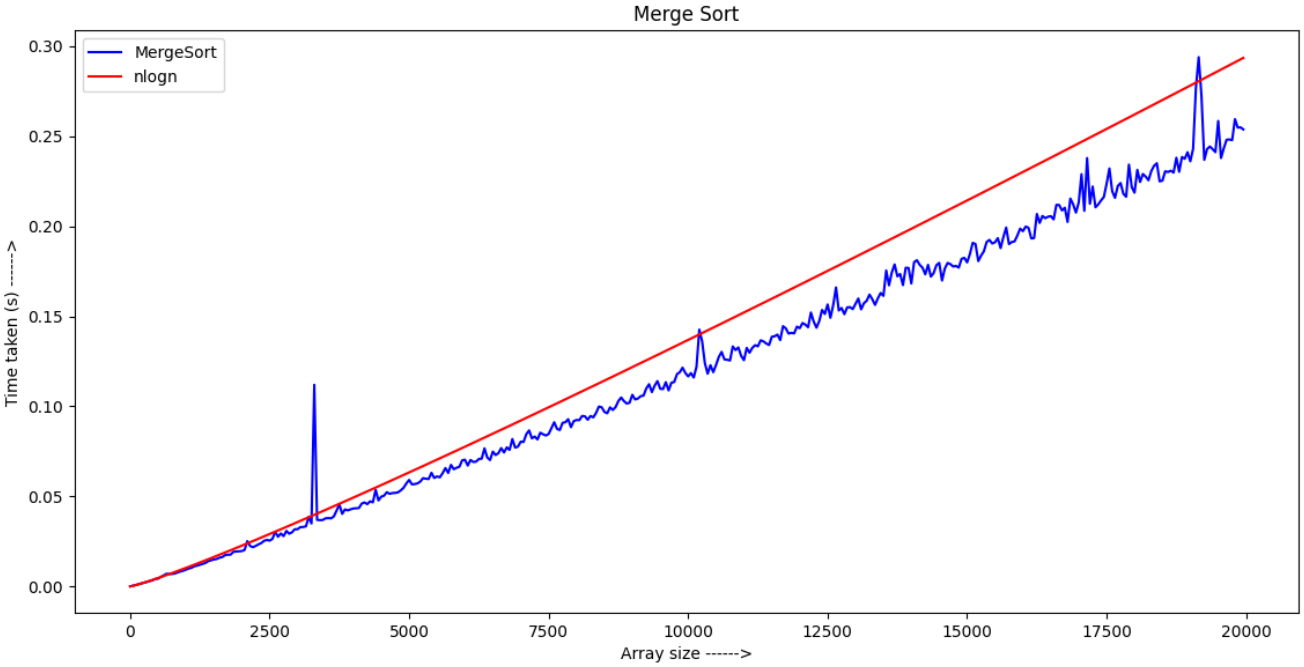
Name: Ashutosh Kumar
Msc Sem II (2021 – 23)
Roll No: 21419CMP008

1. Write a program to implement Merge Sort algorithm. Also plot the graph of the time complexity for different values of array size ‘n’. Compare this with the plot of nlogn and give your comments in 2-3 lines.

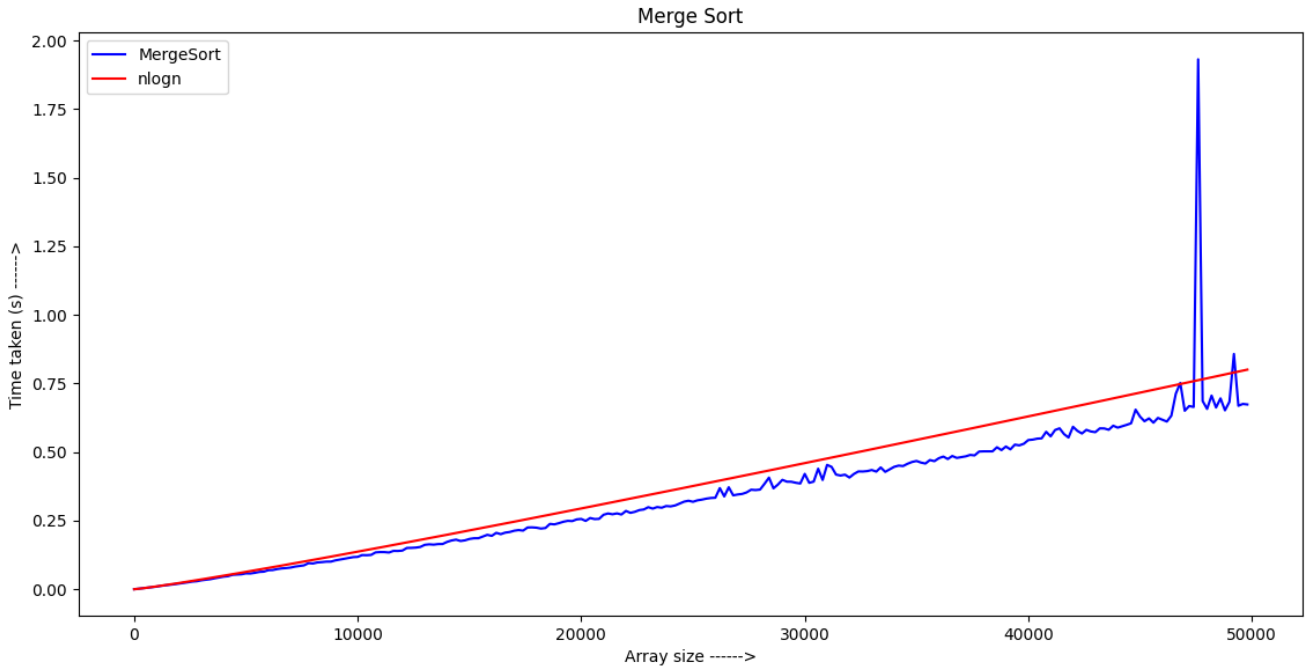
```
# Python program for time complexity graph of mergesort

from random import randint
from math import log2
import timeit
import matplotlib.pyplot as plt

def merge(arr, l, m, r):#merge function to merge the arrays
	n1 = m - l + 1
	n2 = r - m
	L = [0] * (n1)
	R = [0] * (n2)

	for i in range(0, n1):
		L[i] = arr[l + i]
	for j in range(0, n2):
		R[j] = arr[m + 1 + j]
	i = 0
	j = 0
	k = l
	while i < n1 and j < n2:
		if L[i] <= R[j]:
			arr[k] = L[i]
			i += 1
		else:
			arr[k] = R[j]
			j += 1
		k += 1
	while i < n1:
		arr[k] = L[i]
		i += 1
		k += 1
	while j < n2:
		arr[k] = R[j]
		j += 1
		k += 1
def mergeSort(arr, l, r):#mergesort function to perform mergesort
	if l < r:
		m = l+(r-l)//2
		mergeSort(arr, l, m)
		mergeSort(arr, m+1, r)
		merge(arr, l, m, r)
arr = []
x = []
y = []
n = []
for i in range(1, 50000, 200):
   for j in range(i):
```

```
        temp = randint(0, 100000)
        arr.append(temp)#appends a random value to the end of array
    x.append(i)
    start = timeit.default_timer()#for calculating time
    mergeSort(arr, 0, i-1)
    end = timeit.default_timer()#for calculating time
    y.append(end - start)
    n.append(i*log2(i)*0.00000103)
plt.plot(x, y, label = 'MergeSort', color = 'blue')
plt.plot(x, n, label = 'nlogn', color = 'red')
plt.legend()
plt.xlabel('Array size ------>')
plt.ylabel('Time taken (s) ------>')
plt.title('Merge Sort')
plt.show()#shows graph
```

When array size is 20000:



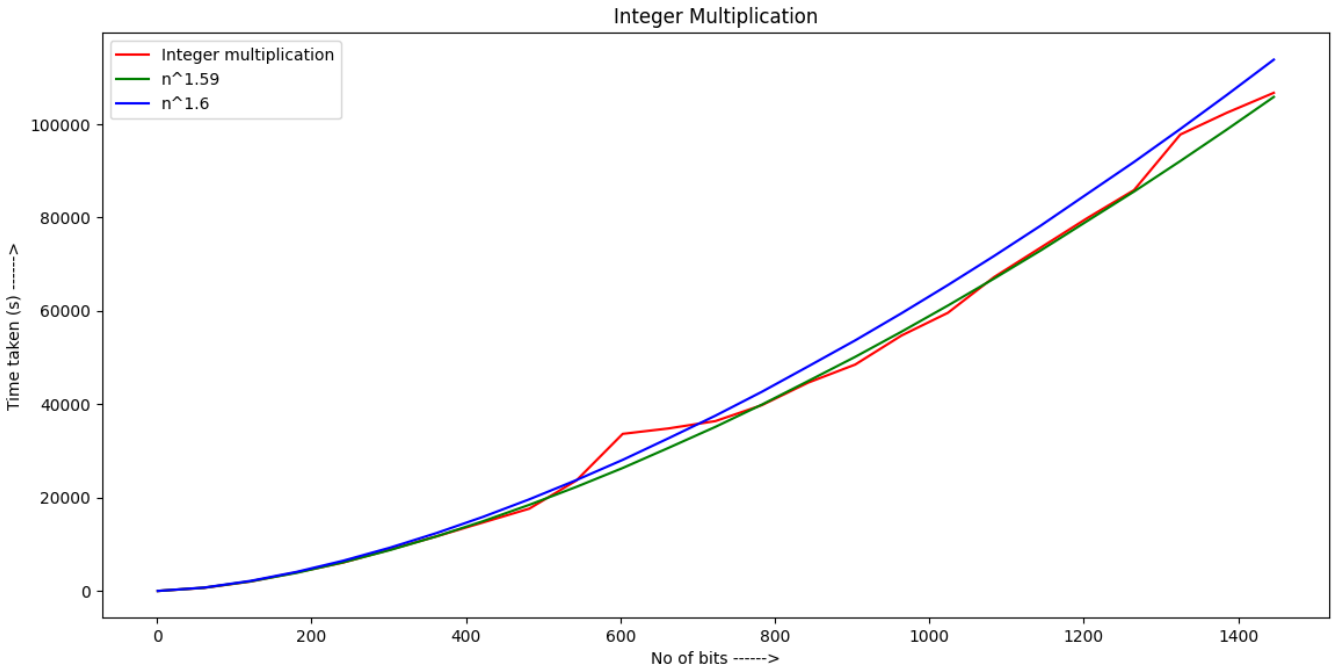
When array size is 50000:

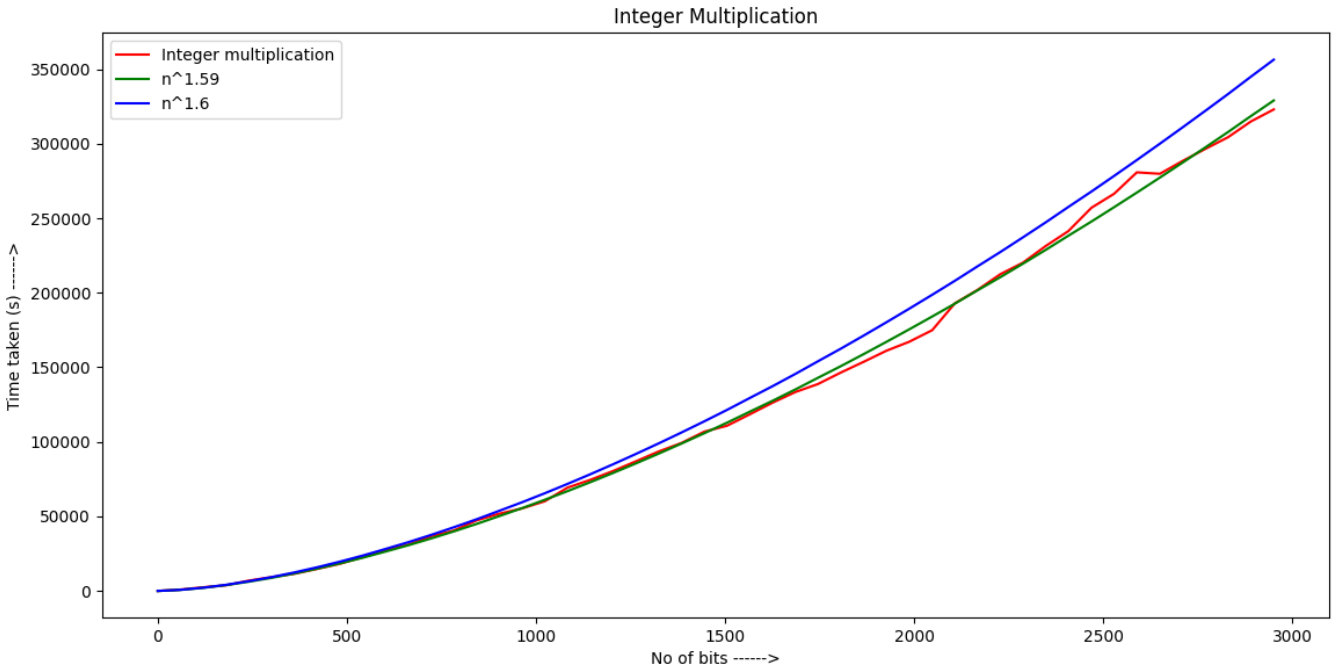


2. Write a program to perform Integer Multiplication using Divide and Conquer technique. Plot the graph showing time taken for running the program for different sizes of input integer numbers. Compare the curve with the curve of n^1.5 and n^1.6 and give your comments.

```
# Python program for time complexity graph of integer multiplication using divide and conquer

import time
import random

import matplotlib.pyplot as plt

def addzero(nString, zeros, left = True):#for adding zeroes
    for i in range(zeros):
        if left:#adds zeroes to left
            nString = '0' + nString
        else:#adds zeroes to right
            nString = nString + '0'
    return nString

def multiply(x ,y):#for integer multiplication
    x = str(x)
    y = str(y)
    if len(x) == 1 and len(y) == 1:
        return int(x) * int(y)
    if len(x) < len(y):
        x = addzero(x, len(y) - len(x))
    elif len(y) < len(x):
        y = addzero(y, len(x) - len(y))
    n = len(x)
    j = n//2
    if (n % 2) != 0:
        j += 1
    Baddzero = n - j
    Aaddzero = Baddzero * 2
    a = int(x[:j])
    b = int(x[j:])
    c = int(y[:j])
    d = int(y[j:])
    ac = multiply(a, c)
    bd = multiply(b, d)
    k = multiply(a + b, c + d)
    A = int(addzero(str(ac), Aaddzero, False))
    B = int(addzero(str(k - ac - bd), Baddzero, False))
    return A + B + bd

n = []
et = []
n_z=[]
n_y = []
```

```
for i in range(100,2000,200):
    x = random.randint(i, 2**i)
    y = random.randint(i, 2**i)

    start = time.time_ns()#for calculating time
    z = multiply(x, y)
    end = time.time_ns()#for calculating time

    n.append(len(str(x)))
    et.append((end - start) * 0.00005975)

    max_len=len(str(max(x,y)))
    n_y.append((max_len**(1.6)))
    n_z.append((max_len**(1.59)))

plt.plot(n, et, label="Integer multiplication",color="r")#plots graph for integer multiplication
plt.plot(n, n_z, label="n^1.59",color="g")#plots graph for n^1.59
plt.plot(n, n_y, label="n^1.6",color="b")#plots graph for n^1.6
plt.xlabel('No of bits ------>')
plt.ylabel('Time taken (s) ------>')
plt.title('Integer Multiplication')
plt.legend()
plt.show()#shows graph
```




Integer Multiplication
Integer multiplication
n^1.59
n^1.6
Time taken (s) ------>
No of bits ------>
0
50000
100000
150000
200000
250000
300000
350000
0
500
1000
1500
2000
2500
3000