Samip Shah
samipjanish.shah@mavs.uta.edu

1. Implement the insertion sort and merge sort algorithms with any programming language you choose and run them with input number lists. Generate the list elements with a random function and increase the list size incrementally until you find the execution time of your merge sort program is consistently shorter. Draw the two curves in a figure (execution time vs input list size) about the two programs. Using the example to discuss why the asymptotic analysis is meaningful. Attached program codes in your submission.

Merge Sort

```
import random
import time
import timeit

start = time.time()

def mergeSort(arr):
    if len(arr) >1:
        mid = len(arr)
        L = arr[:mid]
        R = arr[mid:]

        mergeSort(L)
        mergeSort(R)

        i = j = k = 0

        while i < len(L) and j < len(R):
            if L[i] < R[j]:
                arr[k] = L[i]
                i+=1
            else:
                arr[k] = R[j]
                j+=1
            k+=1

        while i < len(L):
            arr[k] = L[i]
            i+=1
            k+=1

        while j < len(R):
            arr[k] = R[j]
            j+=1
            k+=1

def printList(arr):
```

```
    for i in range(len(arr)):
        print(arr[i],end=" ")
    print()

if __name__ == '__main__':
    res = random.sample(range(1, 10000), 20)

    print ("Given array is", end="\n")
    printList(res)
    mergeSort(res)
    print("Sorted array is: ", end="\n")
    printList(res)

end = time.time()
print(end - start)
```

Insertion Sort

```
import random
import time

start = time.time()

def insertionSort(arr):

    for i in range(1, len(arr)):

        key = arr[i]

        j = i-1
        while j >=0 and key < arr[j] :
                arr[j+1] = arr[j]
                j -= 1
        arr[j+1] = key

res = random.sample(range(1, 10000), 100)
print ("Given array is", end="\n")
print(res)
insertionSort(res)
print ("Sorted array is:")
print(res)

end = time.time()
print(end - start)
```

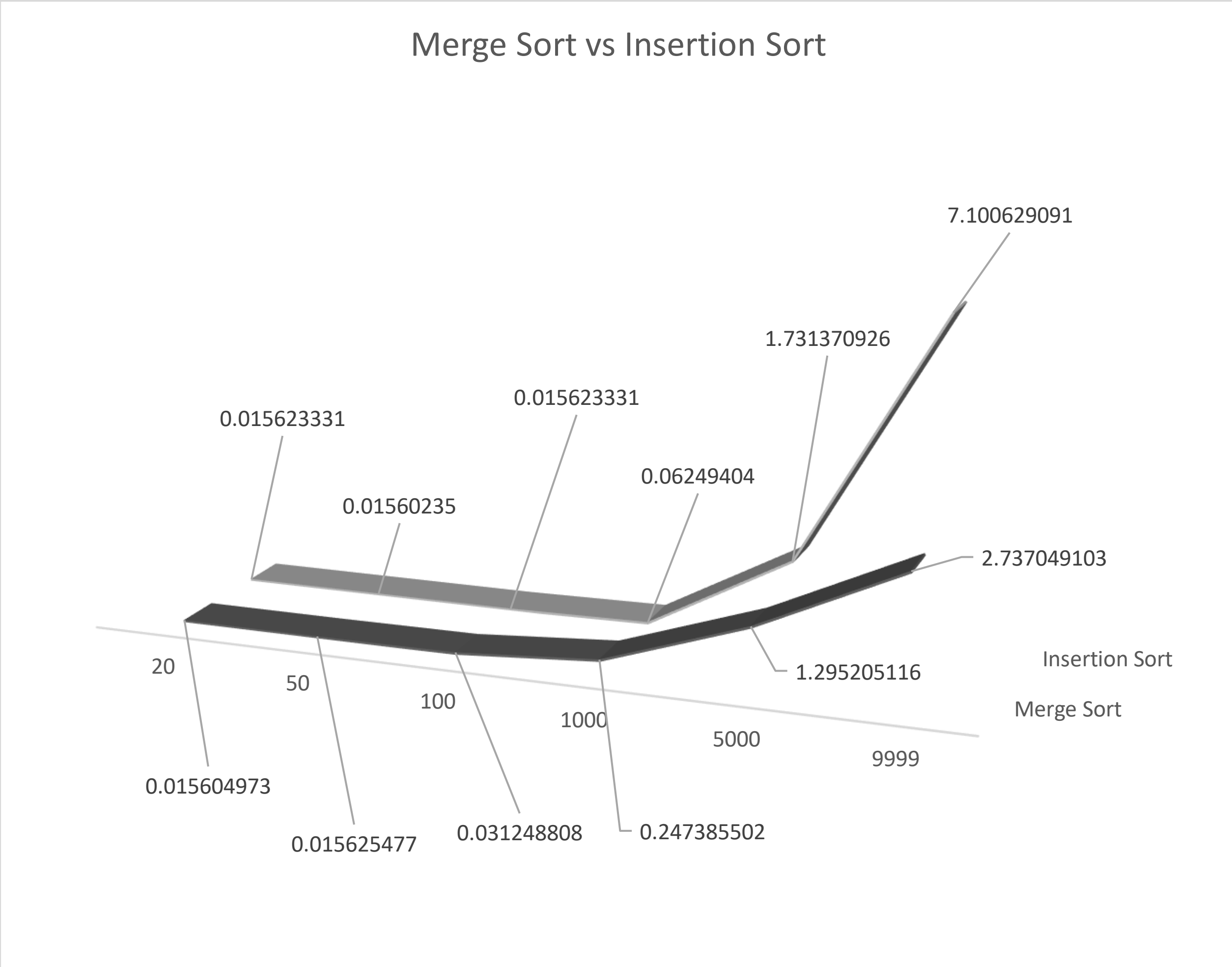

Figure 1

2. Insertion Sort on Small Arrays in Merge Sort

Although merge sort runs in $\Theta(n\lg n)$ worst-case time and insertion sort runs in $\Theta(n^2)$ worst-case time, the constant factors in insertion sort make it faster for small n. Thus, it makes sense to use insertion sort within merge sort when subproblems become sufficiently small. Consider a modification to merge sort in which n/k sublists of length k are sorted using insertion sort and then merged using the standard merging mechanism, where k is a value to be determined.

1. Show that the n/k sublists, each of length k, can be sorted by insertion sort in $\Theta(nk)$ worst-case time.

- For input of length k, insertion sort runs on $\Theta(k^2)$ worst-case time. So the worst-case time required to sort n/k sublists, each of length k, with insertion sort:
  $T(k)=n/k[(ak^2+bk+c)]=ank+bn+(cn/k)$
  Now, k is an integer significantly smaller than n. So, for large values of n, we can ignore the last terms of T(k). Thus, $T(k)=\Theta(nk)$.

2. Show that the sublists can be merged in Θ(nlg(n/k)) worst-case time.

- We have n elements divided into n/k sorted sublists each of length k. To merge these n/k sorted sublists and get a single sorted list of length n, we have to take 2 sublists together and continue to merge them. This will result in log(n/k). And in every step, we are going to compare n elements. So the whole process will take time: Θ(nlg(n/k)).

3. Given that the modified algorithm runs in Θ(nk+nlg(n/k)) worst-case time, what is the largest asymptotic (Θ-notation) value of k as a function of n for which the modified algorithm has the same asymptotic running time as standard merge sort?

- For the modified algorithm to have the same asymptotic running time as standard merge sort, Θ(nk+nlg(n/k))=Θ(nk+nlgn−nlgk) must be same as Θ(nlgn).

  To satisfy this condition, k cannot grow faster than lgn asymptotically (if k grows faster than lgn, because of the nk term, the algorithm will run at worse asymptotic time than Θ(nlgn)). But just this argument is not enough as we have to check for k=Θ(lgn), the requirement holds or not.

  If we assume, k=Θ(lgn),

  $$\begin{aligned}\Theta(nk+nlg(n/k)) &= \Theta(nk+nlgn-nlgk)\\ &= \Theta(nlgn+nlgn-nlg(lgn))\\ &= \Theta(2nlgn-nlg(lgn))\dagger=\Theta(nlgn)\end{aligned}$$

4. How should k be chosen in practice?

- For k to be used in practise we need to calculate exact running time with proper values for constant factors.