ADS Homework 3

Name - Tewodros Adane
Email - tadane@jacobs-university.de

Problem 4.1 – Merge Sort

a) The implementation in python

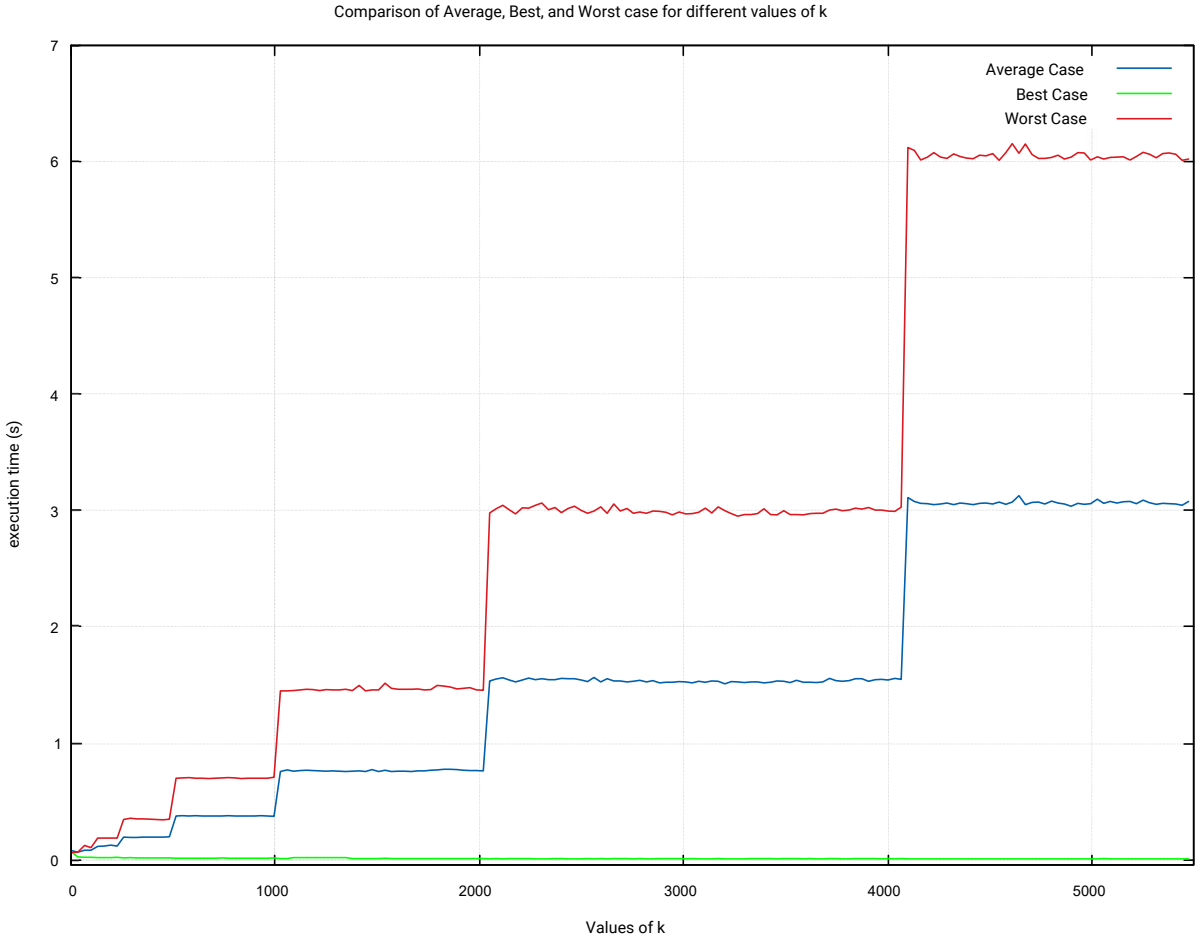
```python
def insertionSort(array):
    for i in range(1, len(array)):
        key = array[i]
        j = i-1
        while j >= 0 and key < array[j]:
                array[j + 1] = array[j]
                j -= 1
        array[j + 1] = key
    return array

def merge(left, right):
    i, j = 0, 0
    merged = []
    while True:
        if i == len(left):
            merged += right[j:]
            break
        if j == len(right):
            merged += left[i:]
            break

        if left[i] > right[j]:
            merged.append(right[j])
            j += 1
        else:
            merged.append(left[i])
            i += 1
    return merged


def mergeSort(array, k):
    if len(array) <= k:
        return insertionSort(array)
    mid = len(array)// 2
    left_half = mergeSort(array[0:mid], k)
    right_half = mergeSort(array[mid:], k)
    return merge(left_half, right_half)
```

b) The following graph depicts the change in execution time as the value of k changes for an array of size 8192.



c) **For the best case:** The best case is an array which is sorted in ascending order. From the graph above, we can see that this array will be sorted faster when **k** is larger. This is because when **k** is larger, the height of the merge sort tree will be shorter, and less time will be spent merging the arrays back. Also, since insertion sort is $O(n)$ in best case it will take significantly less time than the worst and average cases.

For the average case: The average case is an array which is generated using random number generator. The average case has an execution time that look like steps. This is due to the fact that the height of the tree is the same for specific ranges of **k**. For example, because I used an array of size 8192 when the value of **k** is between 2048 and 4096 the height of the tree will always be 3, Thus, after the insertion sort, which takes $O(n^2)$ for sorting the 8192 items, the merging step will take the same time for all **k** values in that range.

For the worst case: The worst case is an array which has all n/k subsequences sorted in a reverse order so that the insertion sort will take the longest time when sorting the subsequences. The graph even though it is higher than the average case, it has the same shape as the former. The reason for this is the same as the explanation for the average case.

d) The optimal choice for **k** is 30. This is because below 30 the average case performs worse than the best case and above 30 the average case even though it is better than the worst case the time it takes is multiple folds for the same array. This is because insertion sort is faster than merge sort for smaller values, therefore, when we give smaller values of **k** more of the work is done by the merge sort instead of the insertion sort.

Problem 4.2 – Recurrences

a) $T(n) = 36T\left(\frac{n}{6}\right) + 2n$

$a = 36, b = 6, \ \log_6 36 = 2$

$f(n) = O(n^{2-\varepsilon})$

$for\ \varepsilon = 1,$

$f(n) = O(n)$

$Therefore, T(n) \in \Theta(n^2)$

b) $T(n) = 5T\left(\frac{n}{3}\right) + 17n^{1.2}$

$a = 5, b = 3, \ \log_3 5 = 1.465$

$f(n) = O(n^{1.465-\varepsilon})$

$for\ \varepsilon = 0.265,$

$f(n) = O(n^{1.2})$

$Therefore, T(n) \in \Theta(n^{1.465})$

c) $T(n) = 12T\left(\frac{n}{2}\right) + n^2 \log n$

$a = 12, b = 2, \ \log_2 12 = 3.585$

$f(n) = O(n^{3.585-\varepsilon})$

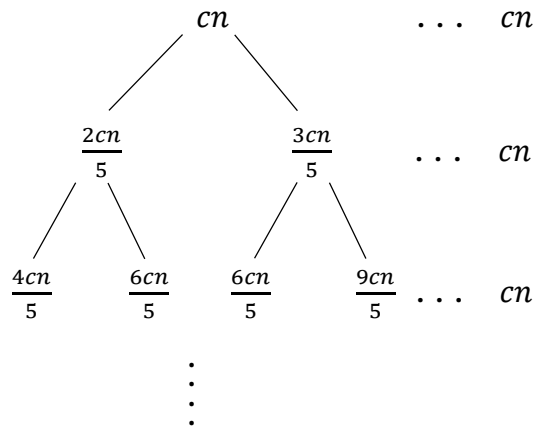
$for\ \varepsilon = 0.585,$

$f(n) = O(n^3)$

$Therefore, T(n) \in \Theta(n^{3.585})$

d)

e) Using recursion tree



The longest branch is going to be the right most branch, and the height of that branch is:

$$\log_{\frac{5}{3}} n$$

$$complexity = cost * height$$

$$= cn.\log_{\frac{5}{3}} n$$

$$= \Theta(\mathrm{n}\log_{\frac{5}{3}} n)$$