Hands On-4

Sathwik Goutham
Std id: 1002242313

Problem 0:

```
def fib(n):
    if n==0:
        return 0
    if n==1
        return 1
    return fib(n-1)+fib(n-2)
```

Recursive calls for fib(5)

```
• fib(5)
   calls fib(4) and fib(3)
   . call stack for fib(4)
        calls: fib(3) and fib(2)
          fib(3)
             calls fib(2) and fib(1)
               fib(2)
                 calls: fib(1) and fib(0)
                    fib(1) returns 1
                    fib(0) returns 0
               fib(2) returns 1
               fib(1) returns 1
          fib(3) returns 2
         fib(2) returns 1
       fib(4) returns 3
     fib(3) returns 2
  fib(5) returns 5
```

Problem 1

Time complexity of the algorithm:

time complexity of merge-two-arrays(arr1, arr2)

Let arr1 and arr2 have lengths m and n

Worst case $(m+n)$ iterations in the function.

$\therefore O(m+n)$

time complexity of merge-sorted-arrays(arrays)

merging k arrays from 1, 2, ..., k

$O(n_1+n_2)$, $O(n_1+n_2+n_3)$ ....

$\therefore O(N \log k)$

$\therefore$ Overall time complexity is $O(N \log k)$

To improve the implementation instead recursive function calls heap method can be used.

Problem 2

Time complexity for duplicates function:

The array is of length n

Worst case, the array is verified from 1 to n index.

$\therefore O(n)$ is the time complexity.

$\therefore$ Sorting array before removing duplicates simplifies the time complexity.