```python
def bucket_sort(lst):
    # measure biggest digit size of list
    n = 0
    for i in lst:
        i = str(abs(i))
        if len(i) > n:
            n = len(i)

    # Bucket sort
    for i in range(1, n+1):  # based on biggest digit size
        bucket_array = [[], [], [], [], [], [], [], [], [], []]
        bucket_array_neg = [[], [], [], [], [], [], [], [], [], []]
        copylst = []  # temp list to hold already sorted values

        for j in lst:
            s = str(abs(j))  # make the value a string
            if len(s) >= i:  # check is current i value matches the length of the string, so that only not sorted value get a true
                if j < 0:
                    bucket_array_neg[int(s[-i])].append(j)  # if the number is negative it goes into the negative bucket array
                else:
                    bucket_array[int(s[-i])].append(j)  # if the number is positive it goes into the positive bucket array
            else:
                copylst.append(j)  # if its false the value has already been sorted and it goes into the copylst

        for i in range(len(bucket_array)):  # add everything from the buckets to the copylst to make the next batch sorted
            copylst += bucket_array[i]
            copylst = bucket_array_neg[i] + copylst

        # put the values of copylst in lst, this way the memory adress will not point at the same space
        lst = list(copylst)

    return lst
```

Tijdscomplexiteit.

Er wordt eerst 1 variabel aangemaakt. Dus +1. En dan een for-loop op basis van de lengte van de lijst. In deze for-loop komen 2 variabelen en een boolean vergelijking voor. Dus: N * 3.

Daarna komt een belangrijke for-loop. Deze for-loop is gebaseerd op variabelen ‘n’, die waarde is zo hoog als het aantal characters van het grootste getal. Dus als de lijst ‘4323’ bevat is n=4. Dit betekent dus dat bij een random gesorteerde list van bijvoorbeeld 9000 waardens n=4 is, maar als de lengte van de lijst 10x zo groot is wordt n alleen met 1 verhoogt (n=5). Dit betekent dus dat de for-loop Log(N) is.

Hierna 1 variabelen en een for-loop gebaseerd op de lengte van de lijst. In deze lijst heb je een worst-case scenario van 3 boolean vergelijkingen en 2 variabelen mutaties. In totaal dus 3 + (N*5)

Daarna een for-loop gebaseerd op de lengte van de bucket-list, dat altijd 9 is, met 2 variabelen mutaties in de for-loop. Met daarbij nog 1 variabelen mutatie. Dus: 9*2+1 = 19

In totaal is dat Log(N) * (3 + (N * 5) + 19), Daar doe je nog de formule van de 1e paragraaf en dan krijg je 1 + (N*3) + Log(N) * (3 + (N * 5) + 19) = **1 + 3N + Log(N) * 5N + 22**

Met het verwijderen van de constante krijg je een Big-O van **O(NLog(N)**

Ruimtecomplexiteit

Eerst heb jee een n = 0 bovenaan staan dat 4 bytes is.

In de for-loop daarna wordt gebruik gemaakt van de input list, maar de variabelen i wordt aangemaakt dat weer 4 bytes inneemt. Daarna worden alleen de variabelen gebruikt die al waren aangemaakt.

Daarna komt een nieuwe for-loop en komt een nieuwe i van 4 bytes. in de for-loop worden er 3 lijsten aangemaakt die in de worst case dezelfde grootte krijgen als de input list. Daarna worden de variabelen j en s aangemaakt dat dus in totaal 4 + 4n + 4n + 4n + 4 + 4 = 12n + 12

Als laatst heb je nog de line lst = list(copylst) dat voor lst nieuwe ruimte aanmaakt om lst en copylst apart te houden. Hier wordt dus een nieuwe 4n bytes voor gemaakt.

Alles bij elkaar krijg je 4 + 4 + 12n + 12 + 4n = **12n + 20 bytes**

Dat heeft een Big-O van **O(N)**