Theoretical analysis of the time complexity of the array implementation – Step 7(inference from google and chatgpt)

Program is break down into components to analyse the time complexities:

1. Generating the Initial List
   - Generating `no_terms` random numbers takes O(no_terms) time.
2. Sorting the List
   - The `merge_sort` function implements the merge sort algorithm, which has a worst case time complexity of O(n log n), where 'n' is the number of elements in the list.
3. Deleting an Element from the List
   - Deleting an element from a list can take O(n) time in the worst case, where 'n' is the number of elements in the list.
4. Inserting an Element into the List
   - Inserting an element into a list can take O(n) time in the worst case, where 'n' is the number of elements in the list.
5. Loop Iterations
   - The loops in the code iterate over the list elements for various purposes. These loops are generally O(n), where 'n' is the number of elements in the list.
6. other constant operations

- Taking user inputs for `no_terms` and `end_range`

```
no_terms = int(input('enter no of terms:'))
end_range = int(input('enter the end range:'))
```

- Initializing variables

```
start_time = time.time()
init_list = []
count = 0
final_list = []
```

- Generating random numbers in the loop:

```
ele = random.randint(1, end_range)
```

- Checking and incrementing variables in the loop:

```
if ele > 50:
   count += 1
```

- Deleting an element from a list

```
del(init_list[4])
```

- Inserting an element into a list

```
final_list = init_list[:i] + [10] + init_list[i:]
```

  The insertion operation depends on the location of insertion (`i`), but it is typically an O(1) operation when inserting an element at a specific position

- Print statements

The program can be expressed as function : f(n)

f(n)= n+n log n+ n + n + n + 1 + 1+ 1 + 1 +1+1 +1

f(n)=4n+n log n + 7

f(n)=4n+n log n (omitting the constant time)

*n log n dominates over constant 4n so asymptotic f(n) can be expressed as

f(n)=0(n log(n))

The overall time complexity of the code can be approximated as:

O(no_terms * log(no_terms))

The dominant factor in terms of time complexity is the sorting step, other operations like deleting and inserting elements contribute linearly, but the sorting step dominates the overall complexity.

Analyzing the space complexity of the code
Breaking down the space usage in the code:

1. `init_list`: This list is used to store the initial random numbers. Its space complexity is O(no_terms) because it grows with the input size.
2. `final_list`: This list is used to store the final result after inserting an element. Its space complexity is also O(no_terms) because it can be at most the same size as `init_list`.
3. Temporary Variables: The code uses some temporary variables like `left_half`, `right_half`, and `result` within the `merge_sort` and `merge` functions. These variables have space complexity proportional to the input size for each recursive call. In the worst case, O(no_terms). The maximum stack depth is O(log(no_terms)) due to the merge-sort recursion. Therefore, the space complexity for these temporary variables is O(no_terms * log(no_terms)).

Overall, the space complexity of the code is dominated by the `init_list`, `final_list`, and the space used by temporary variables during the merge sort. Thus, the total space complexity is:
O(no_terms + no_terms + no_terms * log(no_terms))
Simplifying this expression:
O(no_terms * (1 + log(no_terms)))
In most practical cases, the term `1` is negligible compared to `log(no_terms)`, so the space complexity can be approximated as:
O(no_terms * log(no_terms))

Empirical analysis of time and space complexity – Step 8

Average time taken for n=10 is approximately 589.00 micro sec

```
C:\Users\Sathya\OneDrive\Desktop\time_complexity>python arr_opt.py
enter the vlaue for r:2
enter no of terms:10
enter the end range:100
initial array:
[29, 71, 55, 2, 38, 33, 76, 35, 38, 13]
no of elements greater than 50: 3
after sort:
[76, 71, 55, 38, 38, 35, 33, 29, 13, 2]
after delete
[76, 55, 38, 38, 35, 33, 29, 13, 2]
final array: [76, 55, 38, 38, 35, 33, 29, 13, 10, 2]
Space complexity of my_list: 136 bytes
Elapsed time: 622.60 microseconds
enter no of terms:10
enter the end range:50
initial array:
[41, 10, 4, 30, 10, 6, 34, 17, 19, 28]
no of elements greater than 50: 0
after sort:
[41, 34, 30, 28, 19, 17, 10, 10, 6, 4]
after delete
[41, 30, 28, 19, 17, 10, 10, 6, 4]
final array: [41, 30, 28, 19, 17, 10, 10, 10, 6, 4]
Space complexity of my_list: 136 bytes
Elapsed time: 555.40 microseconds
-------------------
Average time: 589.00 microseconds
```

Average time taken for n=50 is approximately 899.95 micro sec

```
C:\Windows\System32\cmd.e

C:\Users\Sathya\OneDrive\Desktop\time_complexity>python arr_opt.py
enter the vlaue for r:2
enter no of terms:50
enter the end range:100
initial array:
[60, 21, 48, 23, 93, 86, 86, 14, 2, 64, 74, 12, 84, 80, 48, 19, 70, 97, 55, 60, 57, 55, 1, 50, 37, 62, 27, 38, 97, 56, 6, 24, 22, 82, 24, 58, 89, 63, 31, 72, 2, 49, 19, 68, 86, 77, 18, 85, 9, 26]
no of elements greater than 50: 26
after sort:
[1, 2, 2, 6, 9, 12, 14, 18, 19, 19, 21, 22, 23, 24, 24, 26, 27, 31, 37, 38, 48, 48, 49, 50, 55, 55, 56, 57, 58, 60, 60, 62, 63, 64, 68, 70, 72, 74, 77, 80, 82, 84, 85, 86, 86, 86, 89, 93, 97, 97]
after delete
[1, 2, 2, 6, 12, 14, 18, 19, 19, 21, 22, 23, 24, 24, 26, 27, 31, 37, 38, 48, 48, 49, 50, 55, 55, 56, 57, 58, 60, 60, 62, 63, 64, 68, 70, 72, 74, 77, 80, 82, 84, 85, 86, 86, 86, 89, 93, 97, 97]
4
final array: [1, 2, 2, 6, 10, 12, 14, 18, 19, 19, 21, 22, 23, 24, 24, 26, 27, 31, 37, 38, 48, 48, 49, 50, 55, 55, 56, 57, 58, 60, 60, 62, 63, 64, 68, 70, 72, 74, 77, 80, 82, 84, 85, 86, 86, 86, 89, 93, 97, 97]
Space complexity of my_list: 456 bytes
Elapsed time: 983.70 microseconds
enter no of terms:50
enter the end range:50
initial array:
[46, 18, 6, 8, 36, 16, 32, 12, 20, 15, 12, 31, 29, 25, 45, 35, 40, 46, 50, 1, 23, 40, 44, 27, 46, 8, 46, 46, 26, 32, 28, 41, 49, 30, 35, 25, 24, 35, 15, 22, 49, 23, 2, 48, 27, 16, 37, 22, 35, 5]
no of elements greater than 50: 0
after sort:
[50, 49, 49, 48, 46, 46, 46, 46, 46, 45, 44, 41, 40, 40, 37, 36, 35, 35, 35, 35, 32, 32, 31, 30, 29, 28, 27, 27, 26, 25, 25, 24, 23, 23, 22, 22, 20, 18, 16, 16, 15, 15, 12, 12, 8, 8, 6, 5, 2, 1]
after delete
[50, 49, 48, 46, 46, 46, 46, 46, 45, 44, 41, 40, 40, 37, 36, 35, 35, 35, 35, 32, 32, 31, 30, 29, 28, 27, 27, 26, 25, 25, 24, 23, 23, 22, 22, 20, 18, 16, 16, 15, 15, 12, 12, 8, 8, 6, 5, 2, 1]
final array: [50, 49, 48, 46, 46, 46, 46, 46, 45, 44, 41, 40, 40, 37, 36, 35, 35, 35, 35, 32, 32, 31, 30, 29, 28, 27, 27, 26, 25, 25, 24, 23, 23, 22, 22, 20, 18, 16, 16, 15, 15, 12, 12, 10, 8, 8, 6, 5, 2, 1]
Space complexity of my_list: 456 bytes
Elapsed time: 816.20 microseconds
-------------------
Average time: 899.95 microseconds
```

Average time taken for n=100 is approximately 1310.40 micro sec

```
C:\Windows\System32\cmd.e

C:\Users\Sathya\OneDrive\Desktop\time_complexity>python arr_opt.py
enter the vlaue for r:2
enter no of terms:100
enter the end range:100
initial array:
[40, 94, 83, 23, 87, 34, 34, 92, 100, 20, 17, 60, 54, 7, 7, 67, 61, 33, 25, 13, 46, 35, 11, 4, 42, 20, 100, 73, 9, 65, 15, 1, 8, 71, 12, 22, 45, 53, 86, 92, 61, 43, 51, 50, 16, 85, 58, 39, 72, 74, 90, 36, 92,
73, 22, 72, 14, 70, 78, 33, 64, 24, 54, 85, 56, 32, 18, 13, 57, 84, 52, 99, 43, 14, 6, 44, 10, 6, 39, 53, 6, 20, 15, 7, 88, 10, 6, 30, 71, 57, 36, 7, 73, 89, 74, 22, 27, 87, 88, 62]
no of elements greater than 50: 46
after sort:
[1, 4, 6, 6, 6, 6, 7, 7, 7, 7, 8, 9, 10, 10, 11, 12, 13, 13, 14, 14, 15, 15, 16, 17, 18, 20, 20, 20, 22, 22, 22, 23, 24, 25, 27, 30, 32, 33, 33, 34, 34, 35, 36, 36, 39, 39, 40, 42, 43, 43, 44, 45, 46, 50, 51,
52, 53, 53, 54, 54, 56, 57, 57, 58, 60, 61, 61, 62, 64, 65, 67, 70, 71, 71, 72, 72, 73, 73, 73, 74, 74, 78, 83, 84, 85, 85, 86, 87, 87, 88, 88, 89, 90, 92, 92, 92, 94, 99, 100, 100]
after delete
[1, 4, 6, 6, 6, 7, 7, 7, 7, 8, 9, 10, 10, 11, 12, 13, 13, 14, 14, 15, 15, 16, 17, 18, 20, 20, 20, 22, 22, 22, 23, 24, 25, 27, 30, 32, 33, 33, 34, 34, 35, 36, 36, 39, 39, 40, 42, 43, 43, 44, 45, 46, 50, 51, 52,
 53, 53, 54, 54, 56, 57, 57, 58, 60, 61, 61, 62, 64, 65, 67, 70, 71, 71, 72, 72, 73, 73, 73, 74, 74, 78, 83, 84, 85, 85, 86, 87, 87, 88, 88, 89, 90, 92, 92, 92, 94, 99, 100, 100]
11
final array: [1, 4, 6, 6, 6, 7, 7, 7, 7, 8, 9, 10, 10, 10, 11, 12, 13, 13, 14, 14, 15, 15, 16, 17, 18, 20, 20, 20, 22, 22, 22, 23, 24, 25, 27, 30, 32, 33, 33, 34, 34, 35, 36, 36, 39, 39, 40, 42, 43, 43, 44, 45
, 46, 50, 51, 52, 53, 53, 54, 54, 56, 57, 57, 58, 60, 61, 61, 62, 64, 65, 67, 70, 71, 71, 72, 72, 73, 73, 73, 74, 74, 78, 83, 84, 85, 85, 86, 87, 87, 88, 88, 89, 90, 92, 92, 92, 94, 99, 100, 100]
Space complexity of my_list: 856 bytes
Elapsed time: 1339.00 microseconds
enter no of terms:100
enter the end range:50
initial array:
[14, 42, 8, 12, 28, 3, 24, 50, 41, 8, 21, 14, 18, 21, 13, 2, 11, 11, 26, 31, 3, 24, 20, 26, 22, 38, 6, 34, 1, 15, 13, 24, 17, 28, 24, 20, 10, 12, 25, 48, 29, 4, 49, 33, 14, 41, 12, 34, 20, 25, 1, 44, 11, 21, 4
2, 19, 9, 20, 45, 28, 39, 31, 31, 30, 7, 23, 32, 27, 35, 37, 12, 40, 9, 3, 38, 24, 50, 26, 21, 24, 37, 5, 15, 24, 22, 24, 41, 43, 11, 34, 39, 7, 15, 17, 17, 7, 29, 48, 42, 32]
no of elements greater than 50: 0
after sort:
[50, 50, 49, 48, 48, 45, 44, 43, 42, 42, 42, 41, 41, 41, 40, 39, 39, 38, 38, 37, 37, 35, 34, 34, 34, 33, 32, 32, 31, 31, 31, 30, 29, 29, 28, 28, 28, 27, 26, 26, 26, 25, 25, 24, 24, 24, 24, 24, 24, 24, 24, 23,
22, 22, 21, 21, 21, 21, 20, 20, 20, 20, 19, 18, 17, 17, 17, 15, 15, 15, 14, 14, 14, 13, 13, 12, 12, 12, 12, 11, 11, 11, 11, 10, 9, 9, 8, 8, 7, 7, 7, 6, 5, 4, 3, 3, 3, 2, 1, 1]
after delete
[50, 49, 48, 48, 45, 44, 43, 42, 42, 42, 41, 41, 41, 40, 39, 39, 38, 38, 37, 37, 35, 34, 34, 34, 33, 32, 32, 31, 31, 31, 30, 29, 29, 28, 28, 28, 27, 26, 26, 26, 25, 25, 24, 24, 24, 24, 24, 24, 24, 24, 23, 22,
22, 21, 21, 21, 21, 20, 20, 20, 20, 19, 18, 17, 17, 17, 15, 15, 15, 14, 14, 14, 13, 13, 12, 12, 12, 12, 11, 11, 11, 11, 10, 9, 9, 8, 8, 7, 7, 7, 6, 5, 4, 3, 3, 3, 2, 1, 1]
final array: [50, 49, 48, 48, 45, 44, 43, 42, 42, 42, 41, 41, 41, 40, 39, 39, 38, 38, 37, 37, 35, 34, 34, 34, 33, 32, 32, 31, 31, 31, 30, 29, 29, 28, 28, 28, 27, 26, 26, 26, 25, 25, 24, 24, 24, 24, 24, 24, 24,
 24, 23, 22, 22, 21, 21, 21, 21, 20, 20, 20, 20, 19, 18, 17, 17, 17, 15, 15, 15, 14, 14, 14, 13, 13, 12, 12, 12, 12, 11, 11, 11, 11, 10, 10, 9, 9, 8, 8, 7, 7, 7, 6, 5, 4, 3, 3, 3, 2, 1, 1]
Space complexity of my_list: 856 bytes
Elapsed time: 1281.80 microseconds
------------------
Average time: 1310.40 microseconds
```

Step 9: Recursive testing for higher order of N for the comparison between theoretical and empirical measurements.

```
C:\Windows\System32\cmd.e

C:\Users\Sathya\OneDrive\Desktop\time_complexity>python arr_opt.py
enter the vlaue for r:4
enter the end range for m:700
value of n=1000
Elapsed time: 5751.10 microseconds
Elapsed time: 5651.70 microseconds
Elapsed time: 5643.80 microseconds
Elapsed time: 5625.00 microseconds
-------------------
Average time: 5667.90 microseconds
```

```
C:\Windows\System32\cmd.e

C:\Users\Sathya\OneDrive\Desktop\time_complexity>python arr_opt.py
enter the vlaue for r:4
enter the end range for m:500
value of n=500
Elapsed time: 2802.30 microseconds
Elapsed time: 2839.20 microseconds
Elapsed time: 2745.80 microseconds
Elapsed time: 2725.40 microseconds
-------------------
Average time: 2778.18 microseconds
```

C:\Windows\System32\cmd.e

```
C:\Users\Sathya\OneDrive\Desktop\time_complexity>python arr_opt.py
enter the vlaue for r:4
enter the end range for m:8000
value of n=10000
Elapsed time: 74889.90 microseconds
Elapsed time: 73995.70 microseconds
Elapsed time: 73893.30 microseconds
Elapsed time: 73946.60 microseconds
-------------------
Average time: 74181.38 microseconds

C:\Users\Sathya\OneDrive\Desktop\time_complexity>
```

C:\Windows\System32\cmd.e

```
C:\Users\Sathya\OneDrive\Desktop\time_complexity>python arr_opt.py
enter the vlaue for r:4
enter the end range for m:3000
value of n=5000
Elapsed time: 35699.70 microseconds
Elapsed time: 62534.90 microseconds
Elapsed time: 36110.90 microseconds
Elapsed time: 35653.80 microseconds
-------------------
Average time: 42499.83 microseconds
```

C:\Windows\System32\cmd.e

```
C:\Users\Sathya\OneDrive\Desktop\time_complexity>python arr_opt.py
enter the vlaue for r:4
enter the end range for m:50000
value of n=100000
Elapsed time: 363266.70 microseconds
Elapsed time: 883110.60 microseconds
Elapsed time: 889320.00 microseconds
Elapsed time: 901508.20 microseconds
-------------------
Average time: 759301.38 microseconds

C:\Users\Sathya\OneDrive\Desktop\time_complexity>
```

C:\Windows\System32\cmd.e

```
C:\Users\Sathya\OneDrive\Desktop\time_complexity>python arr_opt.py
enter the vlaue for r:4
enter the end range for m:30000
value of n=50000
Elapsed time: 423940.70 microseconds
Elapsed time: 422837.90 microseconds
Elapsed time: 423467.90 microseconds
Elapsed time: 427697.90 microseconds
-------------------
Average time: 424486.10 microseconds
```

```
C:\Users\Sathya\OneDrive\Desktop\time_complexity>python arr_opt.py
enter the vlaue for r:4
enter the end range for m:50000
value of n=1000000
Elapsed time: 10760756.00 microseconds
Elapsed time: 10776526.40 microseconds
Elapsed time: 10726186.60 microseconds
Elapsed time: 10770444.20 microseconds
-------------------
Average time: 10758478.30 microseconds

C:\Users\Sathya\OneDrive\Desktop\time_complexity>
```

```
C:\Users\Sathya\OneDrive\Desktop\time_complexity>python arr_opt.py
enter the vlaue for r:4
enter the end range for m:50000
value of n=10000000
Elapsed time: 130962973.10 microseconds
Elapsed time: 128933400.20 microseconds
Elapsed time: 126861258.40 microseconds
Elapsed time: 129872634.10 microseconds
-------------------
Average time: 129157566.45 microseconds
```