**Report**

**Question 1**

T(n)=7T(n/2) + cn2

Substitute T(n/2) = 7T(n/4) + c(n/2)2

T(n) = 7\*(7T(n/4) + c(n/2)2) + cn2 = 49T(n/4) + 7c(n/2)2 + cn2

Substitute T(n/4) = 7T(n/8) + c(n/4)2

T(n) = 49(7T(n/8) + c(n/4)2) + 7c(n/2)2 + cn2 = 73T(n/8) + 72c(n/4)2 + 7c(n/2)2 + cn2

… …

So we can see the pattern T(n/a) = 7T(n/2a) + c(n/a)2

To the very end, also we have T(1) = 0:

T(n) = cn2 + 7/4\*cn2 + (7/4)2\*cn2 + (7/4)3\*cn2 + …….+（7/4）log2n-1 + nlog27

= 2 + nlog27

≤2 + nlog27

ꞓ O(nlog27)

So T(n) ꞓ O(nlog27),

The equation proved.

**Question 2**

**A:**

For distribution sort, the time complexity is C(n) = 2O(n) + 2O(n\_max) ∈ O(n) , the space complexity is O(n) + O(n\_max). However, for merge sort the time complexity is O(nlog(n)) and the space complexity is O(n + log(n)).

From the description we know that n is large, and the array contains a lot of repeated values, so if we consider the time complexity, the distribution sort performs better, cause when n is large O(n) < O(nlog(n)).

However, merge sort does better in space complexity cause in this condition, we don’t know the candidate values, so n\_max = n, and 2O(n) is larger than n + log(n)

So, if consider more about space, merge sort is recommend, if consider more about time, distribution sort is recommend.

**B:**

In this condition, if we know the array only contains multiples of n, then n\_max = log(n), so the space complexity of distribution and merge sort is same which is O(n + log(n)).

And the time complexity of both algorithms is same as above, O(n) for distribution sort and O(nlog(n)) for merge sort.

So, distribution sort is recommended.