1. 1 , $n^{(1/\log(n))}$, $\log(\log(n))$, $\log_{10}n$, $\sqrt{\log(n)}$, $\log(n)$, $\log(n^2)$, $\log^2 n$,
$(\sqrt{2})^{\log(n)}$,$n^{\log(\log(n))}$,$(\log n)^{\log n}$, $n^{2/3}$, n, $2^{\log n}$, $n^{3/2}$, $n\log n$, $4^{\log n}$, $n + n^2/10^{20}$
2^n, 2^{n+1}, $n.2^n$, 2^{2n}, $2^{(n^2)}$, $n!$,

2. To compute the sorting in O(n) time, radix sort can be used if the numbers are converted to base n. As Radix sort takes O(dn) time . If we can make d constant , radix sort can be done in O(n) time.
   As the given numbers are in the range $[0, n^2 - 1]$, we can represent each number by atmost 2 digits where each digit can vary from 0 to n-1.
   The largest number would be n-1 n-1 where n-1 would be treated as a single digit.$((n-1)*n^1 + (n-1)*n^0) = n^2 - 1$
   To convert the numbers to base n, the time complexity would be negligible as we just have to divide the number atmost twice and take the corresponding one or two remainders which will be treated as least and most significant digits of the number.

   Algo:
   1. Convert the numbers in the base n
   2. Store each digit in separate arrays
   3. Rearrange the array of last digit (using RadixSort from the slide, consecutively rearrange the main array & the baseN array)
   4. Rearrange the array of first digit (using RadixSort from the slide, consecutively rearrange the main array & the baseN array).